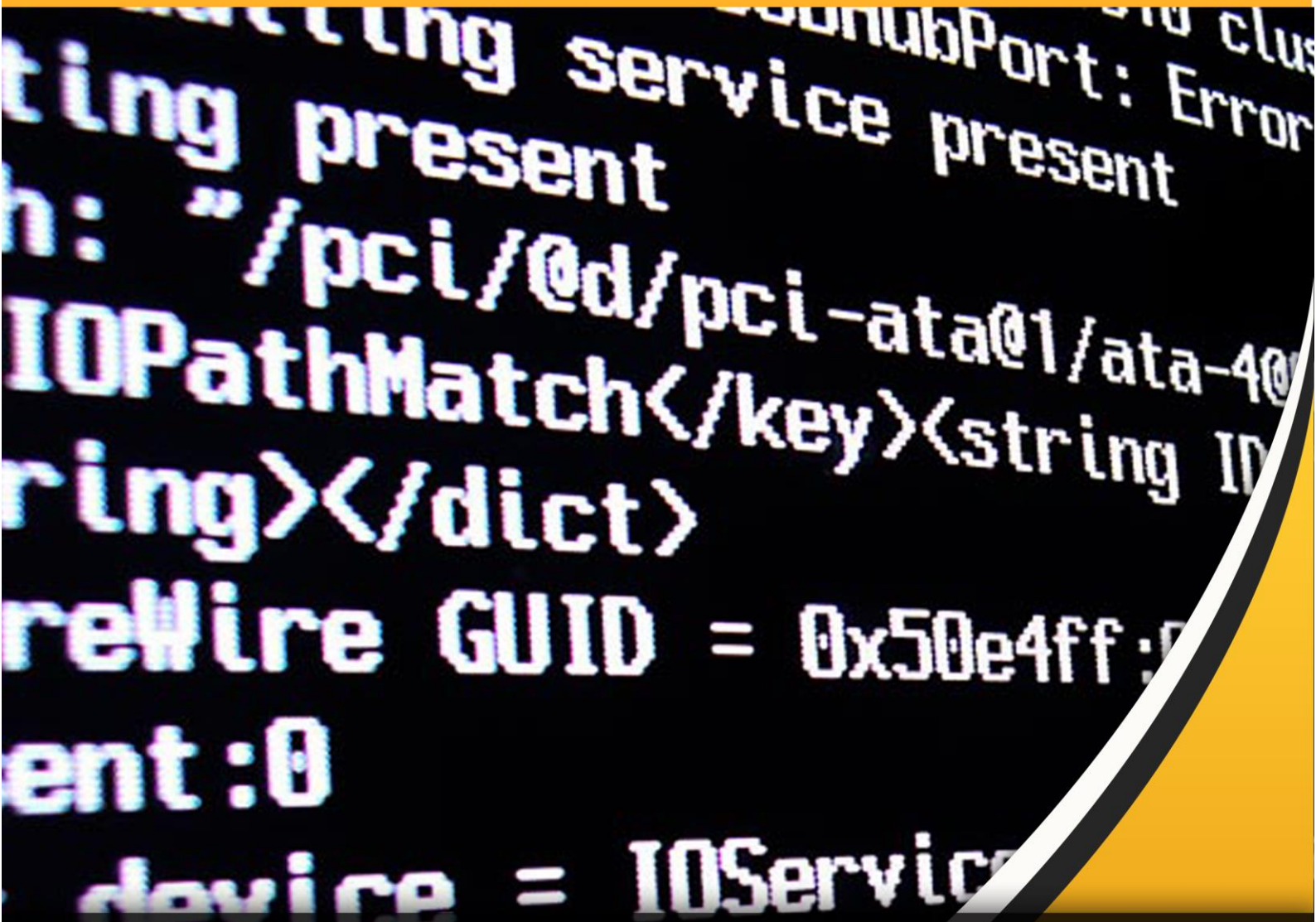
Kementerian Pendidikan Dan Kebudayaan
Republik Indonesia
2013



PEMOGRAMAN BERORIENTASI OBJEK

Semester

2

Untuk SMK / MAK Kelas XI

Penulis : **Eko Subiyantoro**
Editor Materi : **Joko Pitono**
Editor Bahasa :
Ilustrasi Sampul :
Desain & Ilustrasi Buku : **PPPPTK BOE Malang**

Pusat Pengembangan Pemberdayaan Pendidik dan Tenaga Kependidikan Bidang Otomotif dan Elektronika:

Jl. Teluk Mandar, Arjosari Tromol Pos 5, Malang 65102, Telp. (0341) 491239, (0341) 495849, Fax. (0341) 491342, Surel: vedcmalang@vedcmalang.or.id, Laman: www.vedcmalang.com

DISKLAIMER (*DISCLAIMER*)

Penerbit tidak menjamin kebenaran dan keakuratan isi/informasi yang tertulis di dalam buku tek ini.Kebenaran dan keakuratan isi/informasi merupakan tanggung jawab dan wewenang dari penulis.

Penerbit tidak bertanggung jawab dan tidak melayani terhadap semua komentar apapun yang ada didalam buku teks ini.Setiap komentar yang tercantum untuk tujuan perbaikan isi adalah tanggung jawab dari masing-masing penulis.

Setiap kutipan yang ada di dalam buku teks akan dicantumkan sumbernya dan penerbit tidak bertanggung jawab terhadap isi dari kutipan tersebut. Kebenaran keakuratan isi kutipan tetap menjadi tanggung jawab dan hak diberikan pada penulis dan pemilik asli.Penulis bertanggung jawab penuh terhadap setiap perawatan (perbaikan) dalam menyusun informasi dan bahan dalam buku teks ini.

Penerbit tidak bertanggung jawab atas kerugian, kerusakan atau ketidaknyamanan yang disebabkan sebagai akibat dari ketidakjelasan, ketidaktepatan atau kesalahan didalam menyusun makna kalimat didalam buku teks ini.

Kewenangan Penerbit hanya sebatas memindahkan atau menerbitkan mempublikasi, mencetak, memegang dan memproses data sesuai dengan undang-undang yang berkaitan dengan perlindungan data.

Katalog Dalam Terbitan (KDT)
Rekayasa Perangkat Lunak Edisi Pertama 2013
Kementerian Pendidikan & Kebudayaan
Direktorat Jenderal Peningkatan Mutu Pendidik & Tenaga Kependidikan,
th. 2013: Jakarta

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan yang Maha Esa atas tersusunnya buku teks ini, dengan harapan dapat digunakan sebagai buku teks untuk siswa Sekolah Menengah Kejuruan (SMK) Bidang Studi Keahlian Rekayasa Perangkat Lunak.

Penerapan kurikulum 2013 mengacu pada paradigma belajar kurikulum abad 21 menyebabkan terjadinya perubahan, yakni dari pengajaran (*teaching*) menjadi BELAJAR (*learning*), dari pembelajaran yang berpusat kepada guru (*teachers-centered*) menjadi pembelajaran yang berpusat kepada peserta didik (*student-centered*), dari pembelajaran pasif (*pasive learning*) ke cara belajar peserta didik aktif (*active learning-CBSA*) atau *Student Active Learning-SAL*.

Buku teks ″Pemrograman Berorientasi Obyek″ ini disusun berdasarkan tuntutan paradigma pengajaran dan pembelajaran kurikulum 2013 diselaraskan berdasarkan pendekatan model pembelajaran yang sesuai dengan kebutuhan belajar kurikulum abad 21, yaitu pendekatan model pembelajaran berbasis peningkatan keterampilan proses sains.

Penyajian buku teks untuk Mata Pelajaran ″Pemrograman Berorientasi Obyek ″ ini disusun dengan tujuan agar supaya peserta didik dapat melakukan proses pencarian pengetahuan berkenaan dengan materi pelajaran melalui berbagai aktivitas proses sains sebagaimana dilakukan oleh para ilmuwan dalam melakukan eksperimen ilmiah (penerapan scientifik), dengan demikian peserta didik diarahkan untuk menemukan sendiri berbagai fakta, membangun konsep, dan nilai-nilai baru secara mandiri.

Kementerian Pendidikan dan Kebudayaan, Direktorat Pembinaan Sekolah Menengah Kejuruan, dan Direktorat Jenderal Peningkatan Mutu Pendidik dan Tenaga Kependidikan menyampaikan terima kasih, sekaligus saran kritik demi kesempurnaan buku teks ini dan penghargaan kepada semua pihak yang telah berperan serta dalam membantu terselesaikannya buku teks siswa untuk Mata Pelajaran basis data kelas XI / Semester 1 Sekolah Menengah Kejuruan (SMK).

Jakarta, 12 Desember 2013

Menteri Pendidikan dan Kebudayaan

Prof. Dr. Mohammad Nuh, DEA

DAFTAR ISI

HALAMAN SAMPUL i
LAMAN FRANCISDISKLAIMER (*DISCLAIMER*) ii
KATA PENGANTAR iii
DAFTAR ISI iv
GLOSARIUM vii
PETA KEDUDUKAN BUKU ix
Peta Konsep : Pemrograman Berorientasi Obyek Kelas XI Semester 2 x
BAB I PENDAHULUAN 1
A. Deskripsi. 1
B. Prasyarat. 2
C. Petunjuk Penggunaan. 3
D. Tujuan Akhir. 3
E. Kompetensi Inti Dan Kompetensi Dasar 4
F. Cek Kemampuan Awal 5
BAB II KEGIATAN BELAJAR 6
1. Kegiatan Belajar 1 : Interface 6
A. Tujuan Pembelajaran 6
B. Uraian Materi 6
C. Rangkuman 15
D. Tugas 16
E. Tes Formatif 18
F. Lembar Jawaban Test Formatif (LJ) 19
G. Lembar Kerja Siswa 21

2. Kegiatan 2 :Class Built-in 22
A. Tujuan Pembelajaran 22
B. Uraian Materi 22
C. Rangkuman 32
D. Tugas 33
E. Tes Formatif 35
F. Lembar Jawaban Test Formatif (LJ) 36
G. Lembar Kerja Siswa 37
3. Kegiatan 3 : Exception Handling 38
A. Tujuan Pembelajaran 38
B. Uraian Materi 38
C. Rangkuman 52
D. Tugas 53
E. Tes Formatif 54
F. Lembar Jawaban Test Formatif (LJ) 56
G. Lembqar Kerja Siswa 58
4. Kegiatan 4 : String 59
A. Tujuan Pembelajaran 59
B. Uraian Materi 59
C. Rangkuman 72
D. Tugas 73
E. Tes Formatif 74
F. Lembar Jawaban Test Formatif (LJ) 75
G. Lembar Kerja Siswa 76

5. Kegiatan 5 : Array 77
A. Tujuan Pembelajaran 77
B. Uraian Materi 77
C. Rangkuman 90
D. Tugas 91
E. Tes Formatif 93
F. Lembar Jawaban Test Formatif (LJ) 94
G. Lembar Kerja Siswa 96
6. Kegiatan 6 : Sistem File 97
A. Tujuan Pembelajaran 97
B. Uraian Materi 97
C. Rangkuman 103
D. Tugas 105
E. Tes Formatif 106
F. Lembar Jawaban Test Formatif (LJ) 107
G. Lembar Kerja Siswa 109
DAFTAR PUSTAKA 110

GLOSARIUM

Abstract class adalah class yang mempunyai sedikitnya satu abstract method. Abstract class hanya bisa digunakan sebagai super class, dan dapat diturunkan dari class abstract lainnya.

Abstract method adalah method yang belum mempunyai implementasi.

Arrayadalah suatu kumpulan data pada suatu variabel. Array digunakan untuk membuat variabel bisa menampung beberapa data dengan tipe data yang sama alias satu tipe data.

Class StringBuffer adalah pasangan class String yang menyediakan banyak fungsi string yang umum.

Class Wrapper adalah representasi objek sederhana dari variabel- variable non-objek yang sederhana. Ada 10 tipe data Wrapper, yaitu Boolean, Byte, Character, Double, Float, Integer, Long, Number, Short, dan Void.

Collection merupakan istilah umum yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan suatu teknik tertentu pula.

Exception adalah sebuah event yang menjalankan alur proses normal pada program.

Finally merupakan keyword pada class exception handling yang menunjukan bahwa blockprogram tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada.

Interface merupakan sekumpulan dari method-method yang dibuat, namun belum ada operasi di dalam tubuh method tersebut.

Konstruktor merupakan method khusus yang dipakai oleh Java untuk membuat sebuah object didalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor.

List merupakan pengelompokan berdasarkan urutan seperti layaknya array, karena itu ia memiliki posisi awal dan juga posisi akhir.

Modifier digunakan untuk menentukan sifat dari suatu kelas dan menentukan preveledge (hak akses) dari kelas lain.

PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusial.

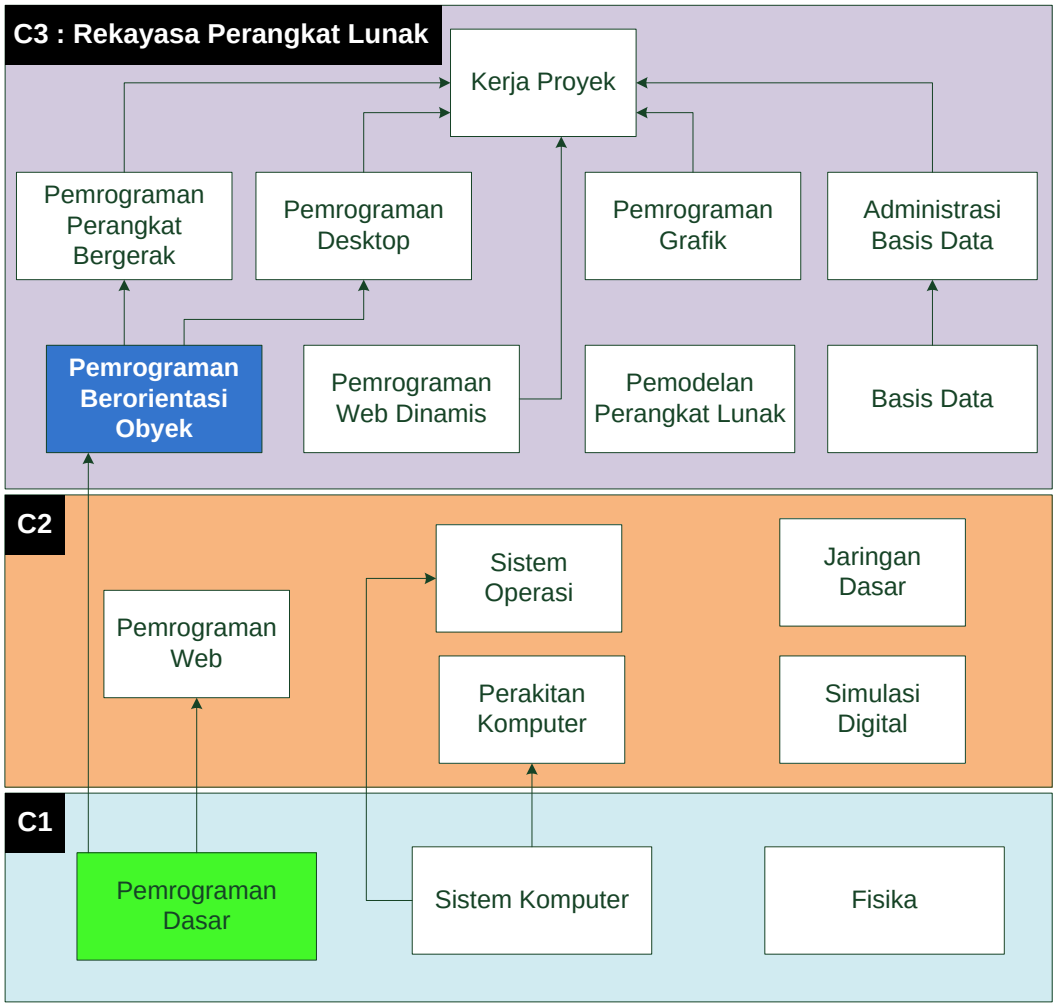
Queue merupakan model pengelompokan berdasarkan metode antrian suatu prioritas tertentu(contoh FIFO-First In First Out).

Set merupakan pengelompokan mengikuti model himpunan dimana setiap anggota-nya harus unik.

Throw digunakan untuk melemparkan suatu bug yang dibuat secara manual.

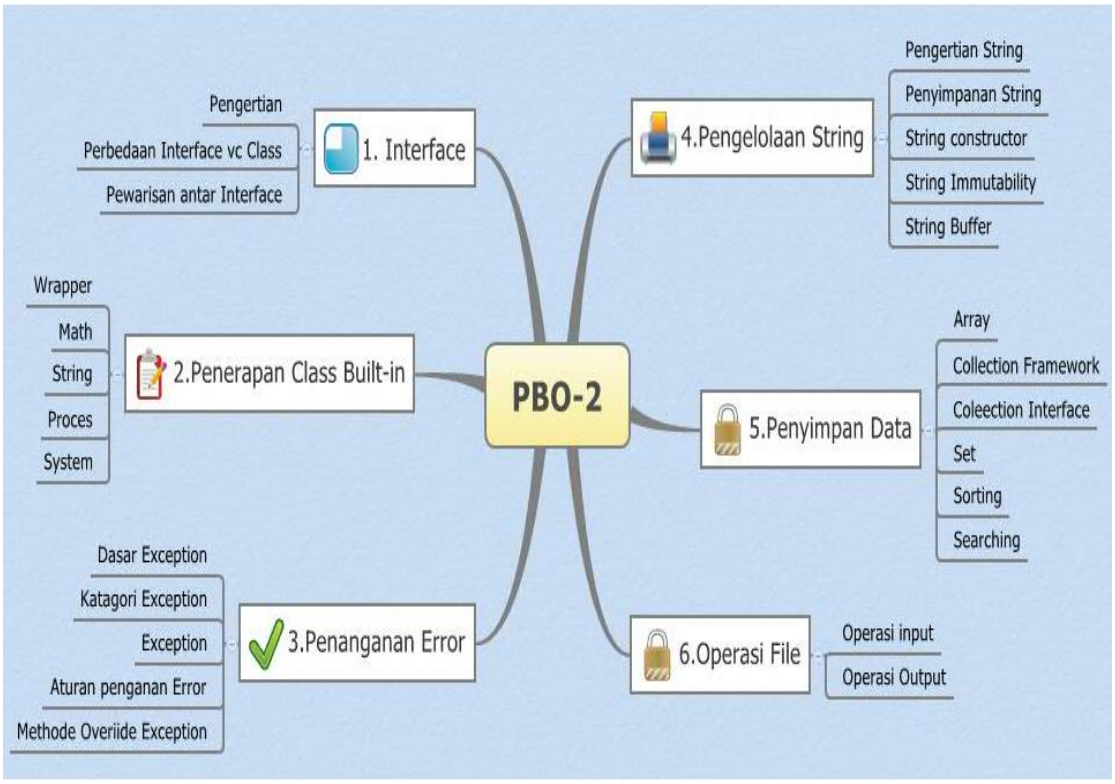
Trydigunakan dalam suatu blockprogram. Keyword ini digunakan untuk mencoba menjalankan blockprogram, kemudian mengenai dimana munculnya kesalahan yang ingin diproses.

PETA KEDUDUKAN BUKU



Keterangan	
C1	Kelompok mata pelajaran Dasar Bidang Keahlian Teknologi Informasi dan Komunikasi
C2	Kelompok mata pelajaran Dasar Program Keahlian Teknik Komputer dan Informatika
C3	Kelompok mata pelajaran Paket Keahlian Rekayasa Perangkat Lunak
	Mata pelajaran Pemrograman Berorientasi Obyek Semester 2
	Mata pelajaran prasyarat

Peta Konsep : Pemrograman Berorientasi Obyek Kelas XI Semester 2



Keterangan	
KD 3.8- 4.8	Interface
KD 3.9- 4.9	Penerapan Class Built-in
KD 3.10- 4.10	Penanganan Error
KD 3.11- 4.11	Pengelolaan String
KD 3.12- 4.12	Penyimpan Data
KD 3.13- 4.13	Operasi File

BAB I PENDAHULUAN

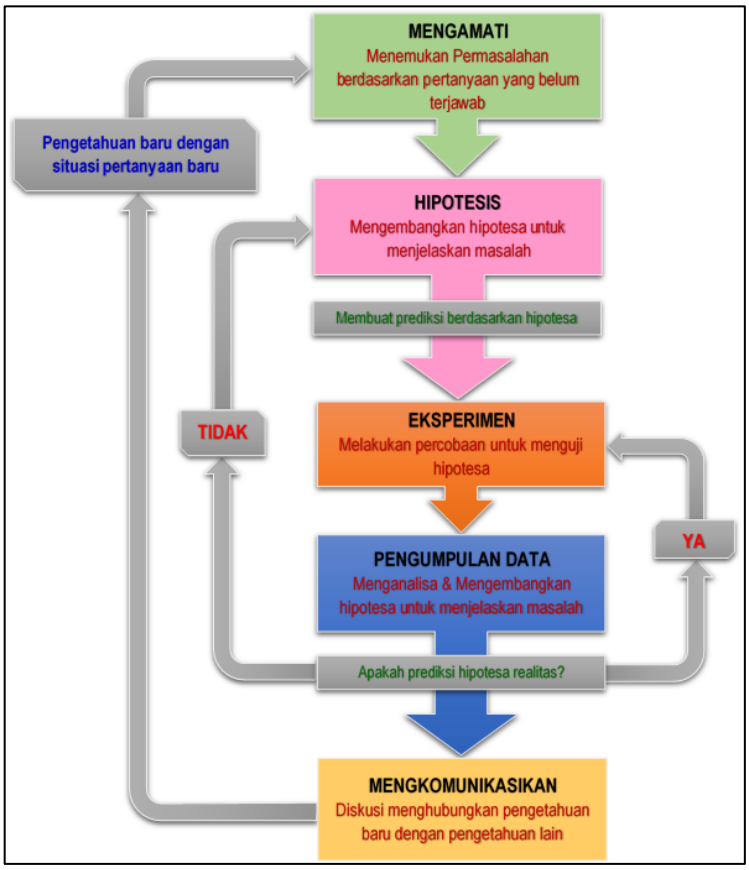
A. Deskripsi.

Pemrograman berorientasi objek (Inggris: *object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Ini adalah jenis pemrograman di mana programmer mendefinisikan tidak hanya tipe data dari sebuah struktur data, tetapi juga jenis operasi (fungsi) yang dapat diterapkan pada struktur data. Dengan cara ini, struktur data menjadi objek yang meliputi data dan fungsi. Selain itu, pemrogram dapat membuat hubungan antara satu benda dan lainnya. Sebagai contoh, objek dapat mewarisi karakteristik dari objek lain.

Salah satu keuntungan utama dari teknik pemrograman berorientasi obyek atas teknik pemrograman prosedural adalah bahwa memungkinkan programmer untuk membuat modul yang tidak perlu diubah ketika sebuah jenis baru objek ditambahkan.Seorang pemrogram hanya dapat membuat objek baru yang mewarisi banyak fitur dari objek yang sudah ada.Hal ini membuat program object-oriented lebih mudah untuk memodifikasi.

Pembelajaran pemrograman berorientasi obyek ini menggunakan metode pendekatan saintifik.Dalam pendekatan ini praktikum atau *e*ksperimen berbasis sains merupakan bidang pendekatan ilmiah dengan tujuan dan aturan khusus, dimana tujuan utamanya adalah untuk memberikan bekal ketrampilan yang kuat dengan disertai landasan teori yang realistis mengenai fenomena yang akan kita amati. Ketika suatu permasalahan yang hendak diamati memunculkan pertanyaan-pertanyaan yang tidak bisa terjawab, maka metode eksperimen ilmiah hendaknya dapat memberikan jawaban melalui proses yang logis. Proses-proses dalam pendekatan scientifik meliputi beberapa tahapan yaitu: mengamati, hipotesis atau menanya, mengasosiasikan atau eksperimen,

mengumpulkan atau analisa data dan mengkomunikasikan. Proses belajar pendekatan eksperimen pada hakekatnya merupakan proses berfikir ilmiah untuk membuktikan hipotesis dengan logika berfikir.



Gambar 1. Diagram Proses Metode Saintifik-Eksperimen Ilmiah

B. Prasyarat.

Untuk kelancaran pencapaian kompetensi dalam mata pelajaran pemrograman berorientasi obyek ini dibutuhkan beberapa peryaratan baik pengetahuan maupun ketrampilan dasar. Persyaratan tersebut antara lain ialah: Peserta didik telah menguasai mata pelajaran pemrograman dasar. Konsep dan algoritma pemrograman ini dibutuhkan untuk mendukung implementasi pemrograman berorientasi obyek.Disamping itu peserta didik mempunyai kompetensi dalam hal pemanfaatan teknologi informasi, seperti mengoperasikan hardware komputer dan mengoperasikan perangkat lunak aplikasi. Perangkat lunak aplikasi tersebut antar lain ialah pengolah data untuk menganalisis data

hasil eksperimen, pengolah kata untuk membuat laporan dan aplikasi presentasi untuk mengkomunikasikan dan mempresentasikan hasil laporan.

C. Petunjuk Penggunaan.

Buku pedoman siswa ini disusun berdasarkan kurikulum 2013 yang mempunyai ciri khas penggunaan metode scientifik. Buku ini terdiri dari dua bab yaitu bab satu pendahuluan dan bab dua pembelajaran. Dalam bab pendahuluan beberapa yang harus dipelajari peserta didik adalah diskripsi mata pelajaran yang berisi informasi umum, rasionalisasi dan penggunaan metode scientifik. Selanjutnya pengetahuan tentang persyaratan, tujuan yang diharapkan, kompetensi inti dan dasar yang akan dicapai serta test kemampuan awal.

Bab dua menuntun peserta didik untuk memahami deskripsi umum tentang topik yang akan dipelajari dan rincian kegiatan belajar sesuai dengan kompetensi dan tujuan yang akan dicapai. Setiap kegiatan belajar terdiri dari tujuan dan uraian materi topik pembelajaran, tugas serta test formatif. Uraian pembelajaran berisi tentang diskripsi pemahaman topik materi untuk memenuhi kompetensi pengetahuan.Uraian pembelajaran juga menjelaskan deskripsi unjuk kerja atau langkah-langkah logis untuk memenuhi kompetensi skill.

Tugas yang harus dikerjakan oleh peserta didik dapat berupa tugas praktek, eksperimen atau pendalaman materi pembelajaran. Setiap tugas yang dilakukan melalui beberapa tahapan saintifik yaitu : 1) melakukan pengamatan setiap tahapan unjuk kerja 2) melakukan praktek sesuai dengan unjuk kerja 3) mengumpulkan data yang dihasilkan setiap tahapan 4) menganalisa hasil data menggunakan analisa diskriptif 5) mengasosiasikan beberapa pengetahuan dalam uraian materi pembelajaran untuk membentuk suatu kesimpulan 6) mengkomunikasikan hasil dengan membuat laporan portofolio. Laporan tersebut merupakan tagihan yang akan dijadikan sebagai salah satu referensi penilaaian.

D. Tujuan Akhir.

Setelah mempelajari uraian materi dalam bab pembelajaran dan kegiatan belajar diharapkan peserta didik dapatmemiliki kompetensi sikap, pengetahuan dan ketrampilan yang berkaitan dengan materi:

- ✓ Interface
- ✓ Penerapan class built-in

- ✓ Penanganan Error
- ✓ Pengelolaan String
- ✓ Penyimpanan Data

E. Kompetensi Inti Dan Kompetensi Dasar

1. **Kompetensi Inti 1 :** Menghayati dan mengamalkan ajaran agama yang dianutnya.

   Kompetensi Dasar :

   1.1. Memahami nilai-nilai keimanan dengan menyadari hubungan keteraturan dan kompleksitas alam dan jagad raya terhadap kebesaran Tuhan yang menciptakannya

   1.2. Mendeskripsikan kebesaran Tuhan yang menciptakan berbagai sumber energi di alam

   1.3. Mengamalkan nilai-nilai keimanan sesuai dengan ajaran agama dalam kehidupan sehari-hari.

2. **Kompetensi Inti 2:** Menghayati dan Mengamalkan perilaku jujur, disiplin, tanggung jawab, peduli (gotong royong, kerjasama, toleran, damai), santun, responsif dan proaktif dan menunjukan sikap sebagai bagian dari solusi atas berbagai permasalahan dalam berinteraksi secara efektif dengan lingkungan sosial dan alam serta dalam menempatkan diri sebagai cerminan bangsa dalam menempatkan diri sebagai cerminan bangsa dalam pergaulan dunia.

   Kompetensi Dasar:

   2.1. Menunjukkan perilaku ilmiah (memiliki rasa ingin tahu; objektif; jujur; teliti; cermat; tekun; hati-hati; bertanggung jawab; terbuka; kritis; kreatif; inovatif dan peduli lingkungan) dalam aktivitas sehari-hari sebagai wujud implementasi sikap dalam melakukan percobaan dan berdiskusi

   2.2. Menghargai kerja individu dan kelompok dalam aktivitas sehari-hari sebagai wujud implementasi melaksanakan percobaan dan melaporkan hasil percobaan.

   Kompetensi Inti 3: Memahami, menerapkan dan menganalisis pengetahuan faktual, konseptual dan prosedural berdasarkan rasa ingin tahunya tentang ilmu pengetahuan, teknologi, seni, budaya, dan humaniora dalam wawasan kemanusiaan, kebangsaan, kenegaraan, dan peradaban terkait penyebab fenomena dan kejadian dalam bidang kerja yang spesifik untuk memecahkan masalah.

Kompetensi Dasar:

3.8. Memahami pembuatan interface

3.9. Menganalisis pemanfaatan class built-in

3.10. Memahami mekanisme penanganan kesalahan

3.11. Memahami string dan berbagai propertinya

3.12. Memahami data collection sebagai media penyimpanan data.

3.13. Menerapkan operasi file dan Input Output(IO)

3. **Kompetensi Inti 4:** Mengolah, menalar, dan menyaji dalam ranah konkret dan ranah abstrak terkait dengan pengembangan dari yang dipelajarinya di sekolah secara mandiri, dan mampu melaksanakan tugas spesifik dibawah pengawasan langsung.

Kompetensi Dasar:

4.8 Menyajikan hasil pembuatan aplikasi dengan interface

4.9 Menyajikan beberapa class built-in dan penerapannya dalam memecahkan masalah

4.10 Mengolah penanganan error dalam mendeteksi kesalahan program

4.11 Mengolah data String dan berbagai propertinya

4.12 Menyajikan data colection sebagai penyimpan data

4.13 Menyajikan operasi file dan operasi Input Output

F. Cek Kemampuan Awal

1. Jelaskan perbedaan perbedaan interface dengan class !
2. Jelaskan fungsi dan manfaat class-class built-in!
3. Jelaskan secara singkat cara penanganan error!
4. Jelaskan cara pengelolaan data String !
5. Jelaskan secara singkat konsep konsep data collection sebagai penyimpan data !
6. Jelaskan secara singkat operasi file dalam class !

BAB II KEGIATAN BELAJAR

1. Kegiatan Belajar 1 : Interface

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1 ini siswa diharapkan dapat :

1). Memahami pembuatan interface

2). Menyajikan hasil pembuatan aplikasi dengan interface

B. Uraian Materi

1) Pengantar Interface

Di kelas kita sudah belajar apa interface tersebut, untuk mengulang materi di kelas, semoga postingan saya tentang interface kali ini bisa memperjelas tentang konsep interface yang kita pelajari di kelas praktikum dan penjelasan tugas yang sudah dikumpulkan minggu lalu.

Kenapa kita butuh interface? Sebagai pengantar kita harus mengetahui apa yang disebut interface dan kegunaannya dalam pemrograman java khususnya pemrograman berorientasi objek, karena kita akan bermain banyak dengan objek tersebut.

Interface adalah jenis khusus dari blok yang hanya berisi method signatureatau constant. Interface mendefinisikan sebuahsignature dari sebuah kumpulan method tanpa tubuh. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-classtanpa memperhatikan lokasinya dalam hirarki classuntuk mengimplementasikan sifat-sifat yang umum. Dengan catatan bahwa interface juga menunjukkan polimorfisme, dikarenakan program dapat memanggil method interface dan versi yang tepat dari method yang akan dieksekusi tergantung dari tipe object yang melewati pemanggil method interface.

Untuk lebih mudah memahami, interface merupakan sekumpulan dari method-method yang dibuat, namun belum ada operasi di dalam tubuh method tersebut.Interface bisa diturunkan atau diwariskan kepada class yang ingin

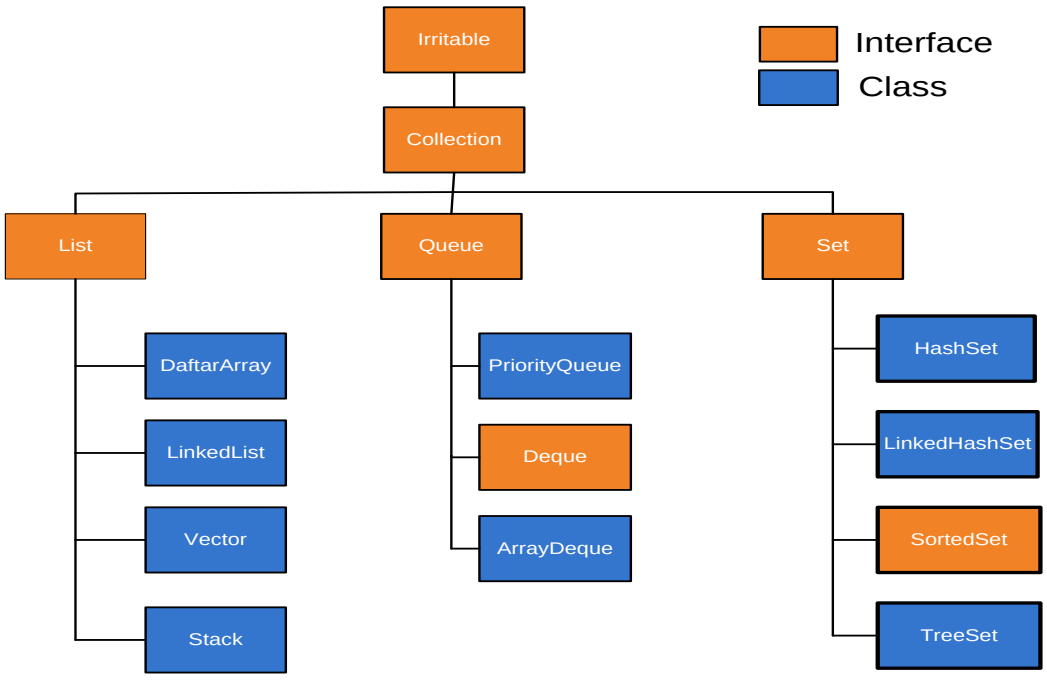
memakai method yang ada dalam masing-masing interface tersebut dengan keyword **extends** [interface yang didefinisikan]. Sebuah class dapat mengimplementasikan 1 interface yang sudah dibuat dengan keyword **implement.**

- ✓ **Ciri-ciri Interface**

Ciri-ciri dari interface adalah sebagai berikut :

- Method interface tidak punya tubuh, sebuah interface hanya dapat mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class lainnya, mereka didefinisikan secara independent.
- Tidak bisa membuat instance atau objek baru dari sebuah interface.
- Ciri umum lain adalah baik interface maupun class dapat mendefinisikan method. Bagaimanapun, sebuah interface tidak memiliki kode implementasi sedangkan class memiliki salah satunya.

- ✓ **Perbedaan Interface dan Class**



- ✓ **Pendeklarasian Interface**

Contoh pendeklarasian interface adalah sebagai berikut :

Listing Program

```
1 interface Operasi
2 {
3 public void Penjumlahan();
4 public void Pengurangan ();
5 }
```

- ✓ **Implementasi Interface**

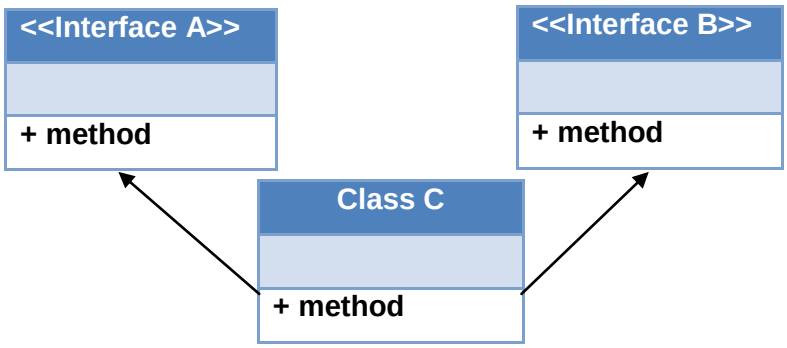
Cara menggunakan interface pada kelas lain, harus menggunakan keyword **implements.** Deklarasi implements interface sebagai berikut :

Listing Program

```
1 public class Calculator implements Operasi{
2 private double bil1, bil2;
3 …//override semua method pada interface
4 public static void main (String []args){
5 Calculator cal = new Calculator(3,4);
6       System.out.print       (“Hasil       penjumlahan
“+cal.getbil1()+” dan “cal.getbil2()”adalah “);
7 cal.Penjumlahan();
8 }
9 }
```

- ✓ **Multiple Interface**

Java tidak memperkenankan adanya multiple inheritance, tetapi java memperbolehkan multiple interface.Dibawah ini adalah ilustrasi multiple interface.



2) Abstract Class

Abstract class adalah class yang mempunyai sedikitnya satu abstract method.Abstract class hanya bisa digunakan sebagai super class, dan dapat diturunkan dari class abstract lainnya.Untuk mendeklarasikan sebuah abstract class digunakan keyword abstract, **[abstract] class [class_name].**

Sebuah abstract class pada dasarnya tidak jauh berbeda dengan class lainnya, yakni juga berisi method yang menggambarkan karakteristik dari kelas abstract tersebut.Perbedaannya yaitu sebuah abstract class bisa berisi method tanpa diimplementasikan, artinya sebuah method tanpa body.Method seperti ini disebut method abstract.

✓ **Implementasi Abstract Class**

Abstract class tidak bisa dibuat objectnya atau tidak dapat di instasiasi. Object hanya bisa dibuat dari non-abstract class (concrete class). Konsekuensinya suatu abstract class harus diturunkan dimana pada subclass tersebut berisi implementasi dari abstract method yang ada di superclass.

Sintaks dalam membuat abstract class adalah sebagai berikut :

Listing Program

```
1 public abstract class Hewan
2 {
3 …//definisi class
4 }
```

Sintaks dalam membuat method abstract class adalah sebagai berikut :

Listing Program

```
1 public abstract class Hewan{
2 void Bernafas(){
3 System.out.println("Bernafas");
4 }
5 }
```

- ✓ **Abstract Method**

Abstract method adalah method yang belum mempunyai implementasi.Kita dapat menyatakan suatu method abstract dengan membutuhkan keyword abstract pada deklarasi method tersebut.

Secara umum sintaks pendeklarasian abstract method adalah sebagai berikut :

Listing Program

```
1 abstract class Seniman{
2 public abstract void berkesenian();
3 public void tidur(){
4 System.out.println("Zzz…");
5 }
6 }
```

Listing Program

```
1 class Penyanyi extends Seniman{
2 public void berkesenian(){
3 System.out.println("Tralala-trilili…");
4 }
5 }
```

Listing Program

```
1 public class Explain{
2 public static void main(String args []){
3 Penyanyi Joshua = new Penyanyi();
4 Joshua.berkesenian();
5 }
6 }
```

✓ **Perbedaan Abstract Class dan Interface**

Abstract Class	Interface
1. Bisa berisi abstract dan non-abstract method.	1. Hanya boleh berisi abstract method.
2. Kita harus menuliskan sendiri modifiernya.	2. Kita tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method diinterfaceadalah *public* dan*abstract*.
3. Dapatmendeklarasikanconstantdan instance variable.	3. Hanyabisamendeklarasikanconstant. Secara implisit variable yang dideklarasikan di interface bersifatpublic, static dan final.
4. Method boleh bersifat static.	4. Method tidak boleh bersifat static.
5. Method boleh bersifat final.	5. Method tidak boleh bersifat final.
6. Suatu abstract class hanya bisa meng-*extend* satu abstract class lainnya.	6. Suatu interface bisa meng-*extend* satu atau lebih interface lainnya.
7. Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-implement beberapa interface.	7. Suatu interface hanya bisa meng-*extend* interface lainnya. Dan tidak bisa meng-*implement* class atau interface lainnya.

✓ **Pewarisan antar Interface**

Dalam OOP sering kali kita mendengar istilah pewarisan (Inheritance) , yaitu sebuah sub-class akan mewarisi behavior(method) ataupun attribut yang ada di dalam super-class nya .

Penggunaan inheritance dapat dilakukan secara overriding ataupun secara overloading method. Overloading berarti mendefinisikan beberapa metode yang memiliki nama sama tetapi dengan sidik yang berbeda. Sedangkan overriding berarti menyediakan suatu implementasi baru untuk suatu metode didalam subkelas.Ini menunjukan bahwa secara konsep super-class hanya menyediakan method-method yang belum terdefinisi secara explicit (jelas), sehingga sub-class dapat memanfaatkan method-method super-class tersebut sesuai kebutuhan di setiap sub-class .

Secara umum fungsi pewarisan dikatakan sebagai metode *reuseability* :

1. Behavior(method) dideklarasikan dalam superclass, behavior tersebut otomatis diwariskan ke seluruh subclass.
2. Kita dapat menggunakan kelas yang kita buat sebelumnya (superclass) dengan membuat kelas-kelas baru (subclass) berdasar super class tersebut, dengan karakteristik yang lebih khusus dari behaviour umum yang dimiliki super class.
3. Kita dapat membuat super class yang hanya mendefinisikan behaviour namun tidak memberi implementasi dari metode-metode yang ada (framework class) , superclass seperti ini disebut kelas abstrak (dengan modifier kelas dan method abstract) dan sub-classnya disebut kelas kongkret , sehingga sub-class dapat mengimplementasi method dari superclass sesuai dengan kebutuhan di sub-classnya tanpa mempengaruhi super-classnya.
4. Kita dapat mendefinisi method hanya sekali dan method tersebut dapat digunakan oleh seluruh subclass.
5. Sebuah subclass hanya perlu mengimplementasikan perbedaan antara dirinya sendiri dan parent-nya (super-classnya).

✓ **Contoh pewarisan**

Listing Program

```
1  // yang dijadikan super-class
2  public abstract  class Burung{
3// method abstract
4  public abstract void suara();
5// method non-abstract yang akan dioverride
6  public void bisaTerbang(){
7  System.out.println("bisaTerbang donk!!");
8  }
9  }
```

Listing Program

```
1  // interface Pernafasan
2  interface Pernafasan {
3  // method yang akan di implementasikan
4  void bernafasLewat();
5  }
```

Listing Program

```
q//sub-class pertama
2public class bebek extends Burung implements Pernafasan{
3 //meng-override method bisaTerbang() dari kelas Burung
4 @Override
5 public void bisaTerbang() {
6 System.out.println("ups , cuma berjalan !");
7 }
8 // meng-implements method suara() dari kelas burung
9 public void suara() {
10 System.out.println("kowek-kowek");
11 }
12//mengimplements method bernafasLewatdari
interfacePernafasan
13 public void bernafasLewat() {
14 System.out.println("lewat paru-paru");
15 }
16 }
```

Listing Program

```
1  //sub-class kedua
2  public class perkutut extends burung implements
Pernafasan{
3  // meng-implements method suara() dari kelas burung
4  public void suara() {
5 System.out.println("kuruk-kuruk");
6 }
7//mengimplements method bernafasLewat dari interface
Pernafasan
8public void bernafasLewat() {
9System.out.println("bernafas Lewat paru-paru juga");
10 }
11 }
```

Pada pemrograman Java, sebuah inheritansi yang ditangani dengan metode abstraksi ini disebut single inheritance.Apabila kita ingin membuat pewarisan banyak (multiple inheritance) kita dapat menggunakan metode antar-muka (interface) .

Dalam kode diatas dapat ditemukan sebuah interface ‘Pernafasan’ . Perlu diingat bahwa interface bukan sebuah kelas untuk membuat interface kita menggunakan modifier kelas Interface.

C. Rangkuman

Interface merupakan kumpulan dari method-method yang belum terdapat operasi di dalam tubuh method tersebut.Interface bisa diturunkan atau diwariskan kepada class yang ingin memakai method yang ada dalam masing-masing interface tersebut dengan keyword **extends** [interface yang didefinisikan]. Sebuah class dapat mengimplementasikan 1 interface yang sudah dibuat dengan keyword **implement.**Interface dapat mendefinisikan konstanta. Interface juga tidak dapat membuat instance atau objek baru dari sebuah interface.Baik interface maupun class dapat mendefinisikan method.

Abstract class adalah class yang mempunyai sedikitnya satu abstract method.Abstract class hanya digunakan sebagai super class dan dapat diturunkan dari class abstract lainnya.Pendeklarasian abstract class dengan menggunakan keyword **abstract class.**Abstract class berisi method tanpa diimplementasikan yang disebut dengan method abstract.

Dalam interface terdapat istilah pewarisan (inheritance) yaitu pewarisan method dan atribut dari super-class kepada sub-classnya. Penggunaan inheritance dapat dilakukan secara overriding dan overloading method.

D. Tugas

Tugas 1

Buatlah program berikut :

'Class Bentuk' akan digunakan sebagai abstract class untuk class turunannya yaitu 'Class Lingkaran', 'Class Segitiga', dan 'Class SegiEmpat'

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
q1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

a. Sebutkan definisi dari beberapa istilah berikut :
   a. Interface
   b. Abstract class
   c. Inheritance
   d. Method abstract

b. Jelaskan perbedaan overloading method dan overriding method!

c. Identifikasilah kesalahan yang terdapat pada program berikut !

Listing Program

```
1 ArrayList list new ArrayList();
2 list.add ("Mataram");
3 list.add ("Yogyakarta");
4 list.add (new java.util.Date());
5 String kota list.get(0);
6 list.get (3, "Siantar");
7 System.out.println(list.get(3));
```

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a)

Interface.....

.....

.....

.....

.....

b) Abstract Class

…….....

.....

.....

.....

c) Inheritance

.....

.....

.....

.....

d) Method Abstract

……….....

.....

.....

.....

LJ- 02 : Perbedaan overloading method dan overriding method :

.....

.....

.....

.....

.....

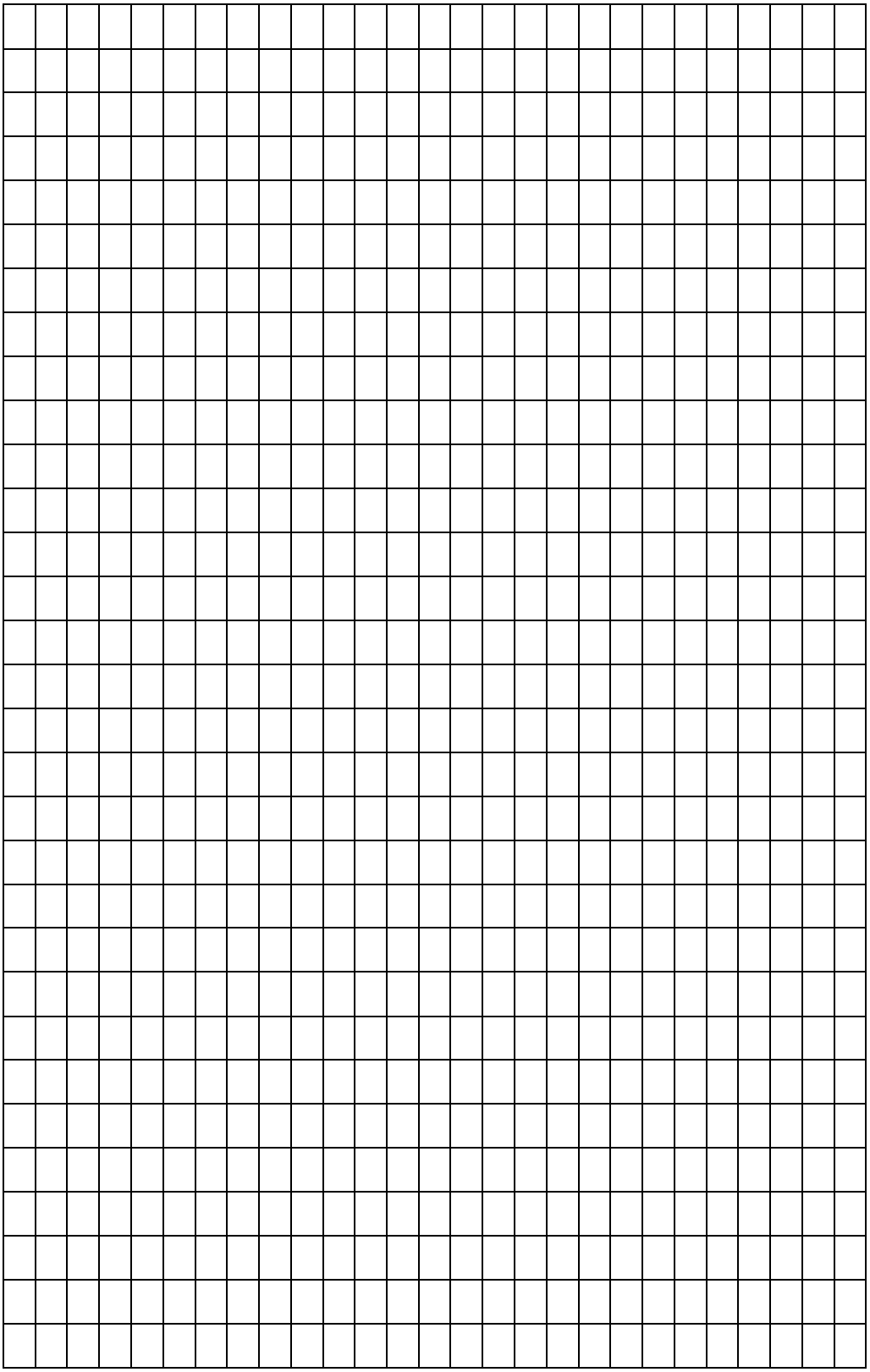
.....

.....

LJ- 03 : Kesalahan pada program :

.....
.....
.....
.....
.....

G. Lembar Kerja Siswa



2. Kegiatan 2 :Class Built-in

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 2 ini siswa diharapkan dapat :

1) Menganalisis pemanfaatan class built in.
2) Menyajikan beberapa class – class built in dan penerapanya dalam memecahkan masalah.

B. Uraian Materi

1) Class Math

Javajugamenyediakankonstantadanmethoduntukmenunjukkanperbedaanoperasi matematikasepertifungsitrigonometridanlogaritma.Selamamethod-methodini semuastatic, kita dapatmenggunakannyatanpamemerlukansebuahobjekMath.Untukmelengkapidaft arkonstantadanmethod-methodini, lihatlahacuanpada dokumentasiJavaAPI.Dibawahinibeberapamethod-methodumumyangsering digunakan.

- ✓ **Public Static double abs (double a)**

Menghasilkan nilai mutlak.Sebuah method yang dioverload.Dapatjuga menggunakannilaifloatatauintegerataujugalongintegersebagaiparameter, dengankondisitipekembaliannyajugamenggunakanfloatatauintegerataulong.

- ✓ **Public Static double random()**

Method ini digunakan untuk membangkitkan suatu nilai double acak dengan rentang lebih besar atau sama dengan nol (0) dan lebih rendah dari 1 **(0 <=Math.random() < 1.0)**. Method ini digunakan untuk menuliskan suatu ekspresi sederhana untuk membangkitkan angka-angka acak dengan sembarang rentang.

- ✓ **Public Static double max (double a, double b)**

Menghasilkannilaimaksimum, diantaraduanilaidouble, adan b. Sebuahmethodyangdi-overload.Dapatjugamenggunakannilaifloatatauintegerataujugalong

integersebagaiparameter,dengankondisitipekembalinyajugamenggunakanfloatata uintegerataulonginteger,secaraberturut-turut.

- ✓ **Public Static double min (double a, double b)**

Menghasilkannilaiminimumdiantaraduanilaidouble,a dan b.Sebuahmethodyangdi-overload.Dapatjugamenggunakannilaifloatatauintegerataujugalonginteger sebagaiparameter,dengankondisitipekembaliannyajugamenggunakanfloatatauint egerataulonginteger,secaraberturut-turut.

- ✓ **Public Static double ceil (double a)**

Menghasilkanbilanganbulatterkecilyanglebihbesaratausamadengana.Mengembal ikan terkecil (paling dekat dengan infinity negatif) nilai ganda yang lebih besar dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.
Kasuskhusus : Jika nilai argumen sudah sama dengan bilangan bulat matematika, maka hasilnya adalah sama dengan argumen. Jika argumen adalah NaN atau tak terhingga atau positif nol atau negatif nol, maka hasilnya adalah sama dengan argumen. Jika nilai argumen kurang dari nol tetapi lebih besar dari -1.0, maka hasilnya adalah nol negatif. Perhatikan bahwa nilai Math.ceil (x) adalah persis nilai-Math. Floor(-x).
Parameter: a-nilai.
Pengembalian: yang terkecil (paling dekat dengan infinity negatif) nilai floating-point yang lebih besar dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

- ✓ **Public Static double floor (double a)**

Menghasilkanbilanganbulatterbesaryanglebihkecilatausamadengana.Mengembal ikan terbesar (paling dekat dengan infinity positif) nilai ganda yang kurang dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.
Kasus khusus : Jika nilai argumen sudah sama dengan bilangan bulat matematika, maka hasilnya adalah sama dengan argumen. Jika argumen adalah NaN atau tak terhingga atau positif nol atau negatif nol, maka hasilnya adalah sama dengan argumen.
Parameter : a- nilai.

Pengembalian: yang terbesar (paling dekat dengan infinity positif) nilai floating-point yang kurang dari atau sama dengan argumen dan sama dengan bilangan bulat matematika.

- ✓ **Public Static double exp (double a)**

MenghasilkanangkaEuler,epangkata. Pengembalian Euler nomor e pangkat dari nilai ganda.

Kasus khusus:Jika argumen adalah NaN, hasilnya adalah NaN.Jika argumen adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argumen adalah infinity negatif, maka hasilnya adalah nol positif. Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat. Hasil harus semi-monoton.

Parameter: a - eksponen untuk meningkatkan e untuk.

Pengembalian : ea nilai, dimana e adalah basis dari logaritma natural.

- ✓ **Public Static double log (double a)**

Menghasilkanlogaritmanaturaldaria.Mengembalikan logaritma natural (base e) dari nilai ganda.

Kasuskhusus : Jika argumen adalah NaN atau kurang dari nol, maka hasilnya adalah NaN. Jika argumen adalah infinity positif, maka hasilnya adalah tak terhingga positif. Jika argumen positif nol atau negatif nol, maka hasilnya adalah tak terhingga negatif. Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat. Hasil harus semi-monoton.

Parameter : a – nilai.

Pengembalian: nilai ln a, logaritma natural dari.

- ✓ **Public Static double pow (double a, double b)**

Menghasilkanapangkatb. Mengembalikan nilai argumen pertama pangkat dari argumen kedua .

Kasus khusus : Jika argumen kedua adalah positif atau negatif nol, maka hasilnya adalah 1,0. Jika argumen kedua adalah 1,0, maka hasilnya adalah sama sebagai argumen pertama. Jika argumen kedua adalah NaN, maka hasilnya adalah NaN.Jika argumen pertama adalah NaN dan argumen kedua

adalah nol, maka hasilnya adalah NaN.(Dalam keterangan di atas, nilai floating - point dianggap integer jika dan hanya jika itu adalah terbatas dan titik tetap dari metode ceil atausama titik tetap lantai metode. Nilai A adalah titik tetap dari metode satu - argumen jika dan hanya jika hasil dari penerapan metode untuknilai sama dengan nilai). Hasilnya dihitung harus berada dalam 1 ulp hasil yang tepat .Hasil harus semi- monoton.

Parameter: a - dasar . b - eksponen .

Pengembalian :nilai ab .

- ✓ **Public Static long round (double a)**

Menghasilkanpembulatankeataskelongterdekat.Sebuahmethodyangdi-overload.Dapatjugamenggunakanfloatpadaargumentdanakanmenghasilkanpembulatankeataskeintterdekat.

- ✓ **Public Static double sqrt (double a)**

Menghasilkanakarkuadrata.Mengembalikanbulatbenarakar kuadratpositifdarinilai ganda.Kasus khusus:Jika argumentadalahNaNataukurang darinol, maka hasilnya adalahNaN.Jika argument adalahinfinitypositif, maka hasilnyaadalah tak terhinggapositif.Jika argumentpositifnolatau negativenol, maka hasilnya adalahsama denganargumen.Jika tidak, hasilnya adalahnilaigandayang paling dekat denganakar kuadratmatematikayang benardari nilaiargumen.

Parameter :a-nilai.

Pengembalian:akar kuadratpositifdari nilai argumen. Jika argumentadalahNaNataukurang darinol, hasilnya adalahNaN.

- ✓ **Public Static double sin (double a)**

Menghasilkansinussudutadalamradian dan mengembalikan sinus trigonometri dari sudut.Kasus khusus:Jika argumen adalah NaN atau tak terhingga, maka hasilnya adalah NaN. Jika argumen adalah nol, maka hasilnya adalah nol dengan tanda yang sama sebagai argumen. Hasilnya dihitung harus berada dalam 1 penghitungan hasil yang tepat. Hasil harus semi-monoton.

Parameter : a - sudut, dalam radian.

Pengembalian : sinus dari argumen.

- ✓ **qPublic Static double toDegrees (double angrad)**

Menghasilkannilaiderajatyangkira-kirasetaradengannilairadianyangdiberikan.Mengubah sudut diukur dalam radian ke sudut kurang lebih setara diukur dalam derajat. Konversi dari radian ke derajat umumnya eksak, pengguna tidak harus mengharapkan cos (toRadians (90,0)) untuk persis sama 0,0.

Parameter: angrad - sudut, dalam radian.

Pengembalian: pengukuran sudut angrad dalam derajat.

- ✓ **Public Static double toRadians (double angdeg)**

Menghasilkannilairadianyangkira-kirasetaradengannilaiderajatyangdiberikan. Dan mengubahsudutdiukur dalam derajatke sudutkurang lebih setaradiukur dalamradian. Konversi dariderajat keradianumumnyaeksak.

Parameter:angdeg-sudut, dalam derajat.

Pengembalian:pengukuranangdegsudutdalam radian.

2) Class String

ClassStringdisediakanolehJavaSDKdenganmenggunakankombinasicharacterliterals.Tidaksepertibahasapemrogramanlainnya,sepertiCatauC++,stringsdapat digunakanmenggunakanarraydaricharacterataudisederhanakandenganmenggunakanclassString.Sebagaicatatan,bahwasebuahobjekStringberbedadari sebuaharraydaricharacter.

- ✓ **Method-Method String**
  - **Public charAt (intidex)**

Mengirimkarakterdiindeksyangditentukanolehparameterindex.Mengembalikannilaiarang padaindeks tertentu.Sebuah indeksberkisar dari 0 sampaipanjang1.NilaiCharpertamaurutanberada padaindeks0, berikutnya pada indeks1, danseterusnya,seperti untukpengindeksan array.Jika nilaiarangyang ditentukan olehindeksadalahpengganti, nilai penggantidikembalikan.Ditentukanoleh:charAtdalam antarmukaCharSequence.

Parameter:Indeks-indeksdari nilaiarang.

Pengembalian :nilaiarang padaindeks tertentudaristring ini.NilaiCharpertamaadalahpada

indeks0.Melempar:IndexOutOfBoundsException–jikaargumentindeksnegatif atautidak kurang daripanjang stringini.

- **Public intCompare To (String another String)**

MembandingkanduaStringdanmengirimbilanganintyangmenentukanapakahobjek stringpemanggilkurangdariatausamadengananotherString.Bernilainegatif jika objekyangdilewatkan(passedstring)lebihbesar,0jikakeduastringsama, dan bernilaipositifjikaobjekstringpemanggil(callingstring)lebihbesar.Jika tidak ada indeks posisi di mana mereka berbeda, maka string yang lebih pendek leksikografi mendahului lagi tali. Dalam hal ini, compareTo mengembalikan perbedaan panjang dari string - yaitu, nilai :

this.length ( ) - anotherString.length ( )

Ditentukan oleh : compareTo dalam antarmuka Sebanding <String>.

Parameter : anotherString - String yang akan dibandingkan .

Pengembalian : nilai 0 jika argumen string sama dengan string ini, nilai kurang dari 0 jika string ini adalah leksikografis kurang dari argumen string, dan nilai lebih besar dari 0 jika string ini adalah leksikografi lebih besar dari argumen string.

- **Public intCompare To Ignore Case (String str)**

SerupadengancompareTotetapicaseinsensitivity.Membandingkan dua string leksikografi, mengabaikan perbedaan kasus. Metode ini mengembalikan sebuah integer yang tanda adalah bahwa memanggil compareTo dengan versi normal dari string di mana perbedaan kasus telah dieliminasi dengan memanggil Character.toLowerCase (Character.toUpperCase (karakter)) pada masing-masing karakter. Perhatikan bahwa metode ini tidak mengambil lokal ke account, dan akan menghasilkan pemesanan memuaskan untuk lokal tertentu. Paket java.text menyediakan collators untuk memungkinkan pemesanan lokal-sensitif.

Parameter : str - String yang akan dibandingkan.

Pengembalian : bilangan bulat negatif, nol, atau bilangan bulat positif sebagai String yang ditentukan lebih besar dari, sama dengan, atau kurang dari String ini, mengabaikan pertimbangan kasus.

- **qPublic Boolean Equals (Object an Object)**

Menghasilkannilaitruejikaparametertunggalnyatersusundarikarakteryangsamaden gan objek tempat Anda memanggil equals. Sedangkan jika parameter yangditentukan bukan sebuah objek String atau jika tidak cocok dengan urutan simbol padastring, method akan dikembalikan dengan nilai false.

- **Public Boolean Equals Ignore Case (String another String)**

Serupa dengan equals tetapi case insensitivity.Membandingkan String ini untuk String lain, mengabaikan pertimbangan kasus. Dua string dianggap sama mengabaikan kasus jika mereka adalah sama panjang dan sesuai karakter dalam dua string yang sama kasus mengabaikan. Dua karakter c1 dan c2 dianggap mengabaikan kasus yang sama jika setidaknya salah satu dari berikut ini benar : Dua karakter yang sama (dibandingkan dengan operator ==) Menerapkan metode Character.toUpperCase (char) untuk masing-masing karakter menghasilkan hasil yang sama menerapkan metode Character.toLowerCase (char) untuk masing-masing karakter menghasilkan hasil yang sama.

Parameter : anotherString - String untuk membandingkan String ini terhadap Pengembalian : true jika argumen tidak null dan merupakan String mengabaikan kasus setara; false jika tidak.

- **Public Void getChars (int srcBegin, int srcEnd,char[ ] dst, int dst Begin)**

Mendapatkan characters dari string yang dimulai pada index srcBegin hingga indexsrcEnd dan menyalin karakter-karakter tersebut pada array dst dimulai pada indexdstBegin.

- **Public intlength()**

Menghasilkan panjang String. Public String replace(char oldChar, char newChar). Mengganti karakter, semua yang kemunculan oldChar diganti newChar.

- **Public String Substring(int beginIndex, int endIndex)**

Mengirim substring dimulai dari indeks beginIndex yang ditentukandan berakhirdengan indeks endIndex yang ditentukan.
Parameter : beginIndex - indeks awal, inklusif.
Pengembalian : substring yang ditentukan.
Melempar : IndexOutOfBoundsException - jika beginIndex negatif atau lebih besar dari panjang objek String ini.

- **Public char[] toCharArray()**

Mengembalikan array karakteryang sama dengan string ini.

- **Public StringTrim()**

Menghilangkan whitespacedi awal dan akhir objek String.Mengembalikan salinan dari string , dengan terkemuka dan trailing spasi dihilangkan. Jika objek String ini merupakan urutan karakter kosong, atau yang pertama dan terakhir karakter dari urutan karakter diwakili oleh objek String ini keduanya memiliki kode lebih besar dari ' \ u0020 ' (karakter spasi), maka referensi ke objek String ini dikembalikan.Jika tidak, jika tidak ada karakter dengan kode yang lebih besar dari ' \ u0020 ' dalam string, maka objek String baru yang mewakili string kosong dibuat dan dikembalikan. Jika tidak, biarkan k menjadi indeks dari karakter pertama dalam string yang lebih besar dari kode ' \ u0020 ', dan biarkan m menjadi indeks dari karakter terakhir dalam string yang lebih besar dari kode ' \ u0020 '. Sebuah objek String baru dibuat , mewakili substring dari string ini yang diawali dengan karakter pada indeks k dan berakhir dengan karakter pada indeks m- yaitu, hasil dari this.Substring (k , m +1).Metode ini dapat digunakan untuk memangkas spasi (seperti dijelaskan di atas) dari awal dan akhir string. Pengembalian :Salinan string ini dengan leading dan trailing spasi dihapus, atau string ini jika tidak memiliki terkemuka atau tertinggal spasi.

- **Public Static StringValueOf(-)**

Dapat menggunakan tipe data sederhana seperti boolean, integer ataucharacter, ataujuga menggunakan sebuah objek sebagai parameter. Mengirim objek String yangmerepresentasikan tipe tertentu yang dilewatkan sebagai parameter.

3) Class StringBuilder/StringBuffer

KetikaobjekStringdiciptakan,objekStringtidakbisalagidimodifikasi.ObjekStringBufferserupadenganobjekString, kecualikenyataanbahwaobjekStringBuffer bersifatdapatberubahataudapatdimodifikasi,sedangkanpadaobjectStringbersifat konstan. Dapat dikatakan bahwa class StringBuffer ini lebih fleksibel disbanding class String. Panjangdanisidapatdiubahhinggabeberapapemanggilanmethod. Class StringBuilder sama dengan class StringBuffer kecuali bahwa metode metode untuk memodifikasi buffer didalam StringBuffer telah disinkronisasi. Class StringBuilder memiliki 3 konstruktor dan lebih dari 30 metode. StringBuffer dapat digunakan jika diakses oleh beberapa perkerjaan secara bersamaan.Sedangkan StringBuilder digunakan jika diakses oleh satu pekerjaan saja.

4) Class Wrapper

Sesungguhnya, tipedataprimitivesepertiint, charandlongbukanlahsebuahobjek.Sehingga, variabel-variabeltipedatainitidakdapatmengaksesmethod-methoddari classObject.Hanyaobjek-objeknyata,yangdideklarasikanmenjadireferensitipedata, dapatmengaksesmethod-methoddariclassObject.Adasuatukeadaan,bagaimanapun,ketikaAndamembutuhkansebuahrepresentasiobjekuntukvariabel-variabeltipe primitivedalamrangkamenggunakanmethod-methodJavabuilt-in.Sebagaicontoh, Andabolehmenambahkanvariabletipeprimitif padaobjekCollection.Disinilahclasswrappermasuk.Classwrapperadalahrepresentasiobjeksederhanadarivariabel- variablenon-objekyangsederhana. Ada 10 tipe data Wrapper, yaitu Boolean, Byte, Character, Double, Float, Integer, Long, Number, Short, dan Void.

Perlu diperhatikan bahwa tipe data wrapper dan tipe data dasar (boolean, byte, char, double, float, int, long, short, void) tidak saling menggantikan. Tipe data dasar dilewatkan ke method dengan *pass by value*, jadi jika membutuhkan *pass by reference* harus memanfaatkan kelas tipe data wrapper. Kelas ini menyediakan versi objek dari tipe data dasar, maka dimungkinkan menambah

method-method untuk masing-masing tipe.

5) Class Process

The ProcessBuilder.start ( ) dan metode Runtime.exec membuat proses asli dan mengembalikan sebuah instance dari subclass Proses yang dapat digunakan untuk mengontrol proses dan mendapatkan informasi tentang hal itu . Proses kelas menyediakan metode untuk melakukan input dari proses, melakukan output ke proses, menunggu proses untuk menyelesaikan , memeriksa status keluar dari proses, dan menghancurkan (membunuh) proses.Metode yang menciptakan proses mungkin tidak bekerja dengan baik untuk proses khusus pada platform asli tertentu, seperti proses asli windowing, proses daemon, proses Win16/DOS pada Microsoft Windows, atau script shell .Secara default, yang subproses dibuat tidak memiliki terminal atau console sendiri. Yang saya standar / O (yaitu stdin, stdout, stderr) operasi akan diarahkan ke proses induk, di mana mereka dapat diakses melalui sungai diperoleh dengan menggunakan metode getOutputStream ( ), getInputStream ( ), dan getErrorStream ( ). Proses induk menggunakan sungai-sungai ini untuk memberi makan masukan ke dan mendapatkan output dari subproses tersebut. Karena beberapa platform asli hanya menyediakan ukuran buffer terbatas untuk standar input dan output stream, kegagalan untuk segera menulis input stream atau membaca output stream dari subproses dapat menyebabkan subproses untuk memblokir, atau bahkan kebuntuan.Dimana diinginkan, subproses I / O juga dapat diarahkan menggunakan metodedari kelas ProcessBuilder.Subproses tidak tewas ketika tidak ada lagi referensi ke objek Proses, melainkan subproses terus melaksanakan asynchronously.Tidak ada persyaratan bahwa proses diwakili oleh objek Proses mengeksekusi asynchronous atau bersamaan sehubungan dengan proses Java yang memiliki objek Proses .

6) Method Process

- ✓ **Public abstract InputStream getInputStream()**

Mengembalikaninput streamterhubung ke outputnormalsubprosestersebut.Arusmemperoleh datapipa darioutput standardari prosesdiwakili olehobjekProsesini.Jikaoutputstandarsubprosestelahdiarahkanmenggunakan ProcessBuilder.redirectOutputmaka metode iniakanmengembalikanaliranmasukannull.Jika tidak, jikastandard errorsubprosestelahdiarahkanmenggunakanProcessBuilder.redirectErrorStream makainput streamdikembalikan olehmetode iniakan menerimaoutput standardigabungdanstandard errordarisubprosestersebut.CatatanPelaksanaan:Iniadalah ide yang baikuntukinput streamkembali kebuffered.Pengembalian:input streamterhubung ke outputnormalsubprosesyang

- ✓ **Public abstract InputStream getErrorStream()**

Mengembalikaninput streamterhubung ke outputkesalahansubprosestersebut.Arusmemperoleh datadisalurkandarikesalahan outputdari prosesdiwakili olehobjekProsesini.Jika kesalahanstandarsubprosestelahdiarahkanmenggunakanProcessBuilder.redirectErroratauProcessBuilder.redirectErrorStreammaka metode iniakanmengembalikanaliranmasukannull.CatatanPelaksanaan:Iniadalah ide yang baikuntukinput streamkembali kebuffered.Pengembalian:input streamterhubung ke outputkesalahanyangsubprocess.

7) Class System

ClassSystemmenyediakanberapafielddanmethodbermanfaat, sepertistandard input,standardoutputdansebuahmethodyangbergunauntukmempercepatpenyalinanbagiansebuaharray.Dibawahinibeberapamethodmenarikdariclass System.Sebagaicatatan,bahwasemuamethod-methodclassadalahstatic.

C. Rangkuman

Ada beberapa class built-in didalam pemrograman java.Yang pertama yaitu class math.Class math dapat berisi method untuk menunjukkan perbedaan operasi matematika seperti fungsi trigonometri dan logaritma.Selama method-

method ini semua static, kita dapat menggunakannya tanpa memerlukan sebuah objek Math.

Yang kedua adalah Class String. Dalam Java, string dapat digunakan menggunakan array dari character atau disederhanakan dengan menggunakan class String.Yang ketiga Class StringBuffer, class ini serupa dengan objek string, kecuali kenyataan bahwa objek String Buffer bersifat dapat berubah atau dapat dimodifikasi, sedangkan pada object String bersifat konstan.
Yang keempat yaitu Class Wrapper, merupakan representasi objek sederhana dari variabel- variable non-objek yang sederhana.

Dan yang kelima yaitu Class Proccess, class ini menyediakan metode untuk melakukan input dari proses, melakukan output ke proses, menunggu proses untuk menyelesaikan, memeriksa status keluar dari proses, dan menghancurkan (membunuh) proses.

D. Tugas

Tugas 1

Buatlah method untuk mengurutkan String menggunakan header sebagai berikut :

```
Public static String urut (String s)
```

❖ **Mengamati Listing Program dan Output Program**

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan perbedaan dari StringBuffer dan StringBuilder !
2. Kesalahan apakah yang terdapat pada kode berikut ini ?

Listing Program

```
1 public void Test{
2 String teks;
3 public void Test (String s){
4 this.text = s;
5 }
6 public static void main (String[] args){
7 Test test = new Test (“ABC”);
8 System.out.println(test);
9 }
10 }
```

3. Pada method process, apa yang dimaksud dengan ‘Public Abstract Input Stream get Input Stream()’ !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Perbedaan StringBuffer dan StringBuilder :

.....
.....
.....
.....
.....
.....
.....
.....

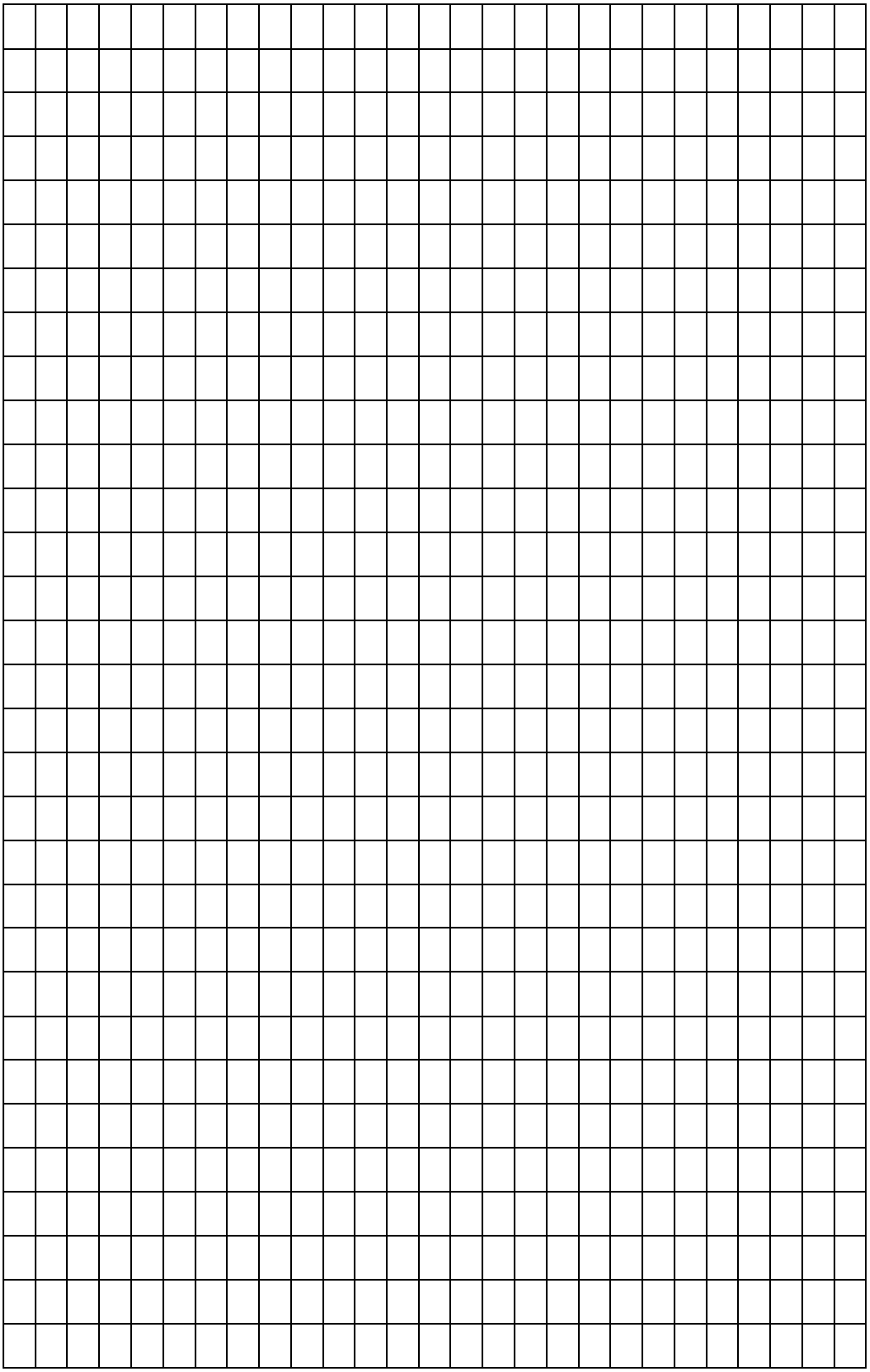
LJ- 02 : Kesalahan dalam kode :

.....
.....
.....
.....
.....
.....
.....
.....

LJ- 03 : Pengertian dari Public Abstract Input Stream get Input Stream() :

.....
.....
.....
.....
.....
.....

G. Lembar Kerja Siswa



3. Kegiatan 3 : Exception Handling

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1) Menganalisismekanisme penangganan kesalahan.

2) Menyajikanbermacam-macam cara untuk mencari tipe kesalahan Uraian Materi.

B. Uraian Materi

1) Dasar Exception

Exception adalah sebuah event yang menjalankan alur proses normal pada program. Event ini biasanya berupa kesalahan(error) dari beberapa bentuk. Ini disebabkan program kita berakhir tidak normal. Dalam bahasa java, ketika terjadi kesalahan, otomatis akan dilemparkan sebuah objek yang disebut exception, yang kemudian dapat diproses lebih lanjut oleh method yang siap menangani kesalahan tersebut. Method tersebut dapat dipilih untuk menangani exception berdasarkan tipe tertentu.Exception dapat muncul tidak beraturan dalam suatu method, atau dapat juga dibuat secara menual dan nantinya melaporkan sejumlah keadaan kesalahan ke method yang memanggil.

2) Tipe-Tipe Exception

Beberapa exception yang telah digunakan dalam begian-bagian terdahulu adalah ArithmeticException, FileNotFoundException, dan InputMisMatchException. Masih banyak kelas exception lain yang digunakan dalam java, antara lain NullPointerException,ClassNotFoundException,IOException, RunTimeException, IndexOutOBoundsException, IllegalArgumentException, dan masih banyak lagi kelas exception yang digunakan dalam java.

Kelas **Throwable** merupakan akar dari semua kelas exception.Semua kelas exception java mewarisi secara langsung atau tidak langsung dari Throwable. Kita bisa menciptakan kelas exception sendiri dengan cara mewarisi exception atau subclass exception.

Kelas-kelas exception dapat diklasifikasikan menjadi 3 tipe utama : error system, exception, dan exception runtime.

- Error system dilempar oleh JVM dan direpresentasikan oleh kelas **Error**. Kelas error mendeskripsikan error internal. Error semacam ini jarang terjadi. Jika terjadi, kita dapat memberitahukan kepada user dan menghentikan program secara normal.contoh sub-kelas Error adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
LinkageError	Kelas punya ketergantungan pada kelas lain, tapi kelas lain telah diubah tanpa menjaga kompatibilitasnya setelah kompilasi dilakukan terhadap kelas pertama.
VirtualMachineError	JVM telah kehabisan sumber daya yang dibutuhkan untuk melanjutkan operasi.

- Exception direpresentasikan dalam kelas **Exception**, yang mendeskripsikan error-error yang diakibatkan oleh program kita dan oleh lingkungan luar. Error ini ditangkap dan ditangani oleh program kita. Beberapa contoh subkelas dari Exception adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
ClassNotFoundException	Percobaan menggunakan suatu kelas yang tidak ada.exception ini terjadi jika kelas yang dijalankan tidak menggunakan perintah java.
IOException	Berkaitan dengan operasi masukan/ keluaran yang tidak valid, membaca melampaui akhir suatu file, dan membuka file yang tidak ada. Subkelas dari IOException antara lain :**InterruptedIOException, EOFException, FileNotFoundException.**

- Exception runtime direpresentasikan oleh kelas **RunTimeException**, yang mendeskripsikan kesalahan pemrograman, seperti casting yang salah, pengaksesan array diluar batas, dan kesalahan numerik. Exception runtime dilempar oleh JVM. Beberapa sub-kelas RunTimeException adalah sebagai berikut :

Kelas	Alasan yang mungkin terjadinya Exception
ArithmeticException	Kesalahan pada operasi aritmatika
IndexOutOfBoundsException	Beberapa jenis indeks diluar batas
NegativeArraySizeException	Array diciptakan dengan ukuran negatif
NullPointerException	Penggunaan acuan null yang tidak valid
ArrayStoreException	Penyimpanan array dengan tipe data yang tidak sesuai
ClassCastException	Cast yang tidak valid
IllegalArrayArgumentException	Argumen yang tidak benar
SecurityException	Aturan sekuriti yang dilanggar
IllegalMonitorStateException	Operasi monitor illegal
IllegalStateException	Lingkungan yang tidak benar
UnsupportedOperationException	Operasi yang tidak didukung

- ✓ **Penanganan Exceptions**

Untuk menangani exception dalam java, kita gunakan blok try-catch-throw-throws-finally. Apa yang kita lakukan dalam program kita adalah kita menempatkan pernyataan yang mungkin menghasilkan exception dalam blok ini. Bentuk umum dari blok try-catch-finally adalah :

Listing Program

```
1  try{
2  //tulis pernyataan yang dapat mengakibatkan exception
3  //dalam blok ini
4  }
5  catch( <exceptionType1><varName1> ){
6   //tulis aksi apa dari program Anda yang dijalankan
jika ada
7  //exception tipe tertentu terjadi
8  }
```

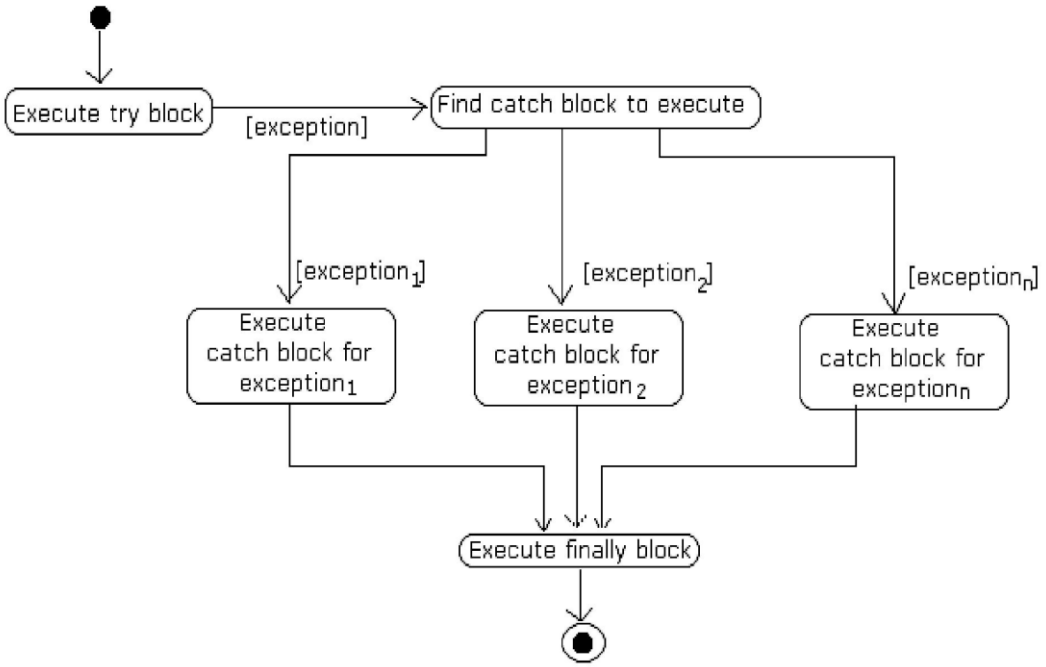
Exception dilemparkan selama eksekusi dari blok *try* dapat ditangkap dan ditangani dalam blok*catch*. Kode dalam blok *finally* selalu di-eksekusi.

Berikut ini adalah aspek kunci tentang sintak dari konstruksi try-catch-finally:

1. Notasi blok bersifat perintah.
2. Setiap blok *try*, terdapat satu atau lebih blok *catch*, tetapi hanya satu blok *finally.*
3. Blok *catch* dan blok *finally* harus selalu muncul dalam konjungsi dengan blok *try*, dan diatasurutan.
4. Blok *try* harus diikuti oleh **paling sedikit** satu blok *catch* ATAU satu blok *finally,* ataukeduanya.
5. Setiap blok *catch* mendefinisikan sebuah penanganan exception. Header dari blok *catch*harus membawa satu argumen, dimana exception pada blok tersebut akan ditangani.

Exception harus menjadi class pelempar atau satu dari subclassesnya.

q



Ada lima keywords yang digunakan oleh Java untuk menangani exception ini, yaitu :***Try*, *catch*, *finally*, *throw*, *throws.***

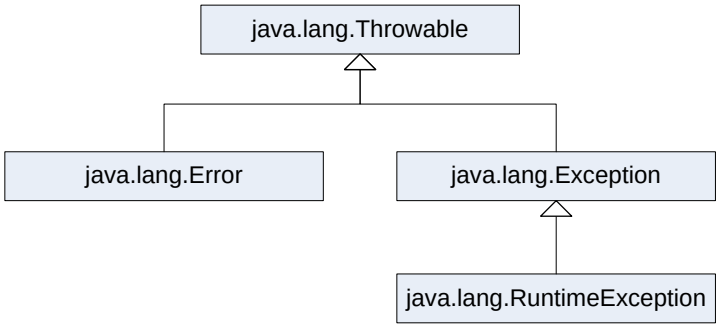
Semua class exception terdapat dalam package **java.lang.**Superclass tertinggi adalah class **Throwable**, tetapi kita hampir tidak pernah menggunakan class ini secara langsung.

Class **Error➔** tipe exception yang seharusnya tidak ditangani dengan menggunakan blok try catch karena berhubungan dengan Java run-time

system/evironment. Jadi exception yang terjadi kemungkinannya sangat kristis yang sebaiknya tidak ditangani oleh program kita sendiri.

Class **Exception➔** tipe exception yang sebaiknya ditangani oleh program kita secara langsung.

Dalam penggunaannya, kita akan banyak menangani exception yg merupakan turunan dari class **Exception** ini. Salah satu turunannya yang perlu diperhatikan adalah class **RuntimeException**,karena Java memperlakukan class ini & turunannya secara berbeda.



- ✓ **Menampilkan Pesan Exception**

Beberapa method standard yang dapat digunakan untuk menampilkan pesan exception merupakan anggota dari kelas java.lang.Throwable

No	Method Pesan Exception	Deskripsi
1	getMessage()	Mengembalikan nilai string yang berisi pesan rinci tentang objek Throwable yang mengalami exception
2	toString()	Mengembalikan nilai string yang berisi pesan singkat tentang objek yang mengalami exception
3	getLocalizedMessage()	Menampilkan pesan exception lokal (yang terjadi pada subkelas saja)
4	printStackTrace()	Method ini bersifat void, dan hanya mencetak informasi tentang objek Throwable

2) Exception Handling

Pada dasarnya, *Exception* merupakan subkelas dari kelas *java.lang.Throwable*. "*Bukalah dokumentasi java untuk lebih menyakinkan anda*". Karena *Exception* adalah sebuah kelas maka hakikatnya ketika program berjalan dan muncul sebuah bug atau kesalahan maka *bug* tersebut dapat dianggap sebuah *object*. Sehingga ketika *object* ini di tampilkan di layar maka java akan secara otomatis memanggil method *toString* yang terdapat dalam *object* bertipe *Exception* ini. Java memberikan akses kepada *developer* untuk mengambil *object* bug yang terjadi ini dengan mekanisme yang dikenal *ExceptionHandling*. *Exceptionhandling* merupakan fasilitas di java yang memberikan flexibilitas kepada *developer* untuk menangkap *bug* atau kesalahan yang terjadi ketika program berjalan. Contoh *ExceptionHandling* akan dibahas pada bagian berikutnya.

✓ **Perbedaan antara *ClassError* dan *ClassException* di java**

Seperti yang telah dijelaskan diatas bahwa kelas *Exception* merupakan kelas turunan dari kelas *Throwable* di *packageJava.Lang*. Selain *Exception*, java.*lang*.*Throwable* juga memiliki *subclass* yaitu *classError*. Tentu, kita bertanya-tanya, sebetulnya apa sih perbedaan antara *classError* dengan *classException*.

✓ **Penjelasan dari ClassError**

"An Error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions" (JDK 5.0 *Documentation*)

✓ **Penjelasan dari *classException***

"The class *Exception* and its subclasses are a form of *Throwable* that indicates conditions that a reasonable application might want to catch. " (JDK 5.0 *Documentation*)Seperti dari penjelasan yang diberikan oleh *JDKDocumentation*, maka dapat kita lihat bahwa *error* dan *exception* pada dasarnya berbeda. *Error* merupakan masalah yang muncul tapi tidak ada alasan yang kuat untuk menangkapnya. Sedangkan *Exception* merupakan kesalahan kecil yang muncul dan ingin diperlakukan sesuai keinginan developer.

3) Keyword penting pada Exception Handling

Ada 5 *keyword* penting dalam java dalam hal *exceptionhandling* :

✓ **Try**

Keywordtry biasanya digunakan dalam suatu *blockprogram*. Keyword ini digunakan untuk mencoba menjalankan *blockprogram*, kemudian mengenai dimana munculnya kesalahan yang ingin diproses.Keyword ini juga harus dipasangkan dengan keywordcatchatau keywordfinally yang akan dibahas pada point kedua dan ketiga.

Contoh penggunaan :

Listing Program

```
1   public class A
2   {
3   public static void main(String[] args) {
4   try
5   {
6    int a = 1 / 0; // berpotensi untuk menimbulkan
kesalahan yaitu
7   // pembagian dengan bilangan 0
8   System.out.println("perintah selanjutnya");
9   }
10  catch (Exception kesalahan)
11  {
12  System.err.println(kesalahan);
13  }
14  }
15  }
```

Output :

Output

```
java.lang.ArithmeticException: / by zero
```

Perhatikan contoh diatas, ada beberapa hal penting yang perlu dilihat. Pertama, *blockprogram* yag diyakini menimbulkan kesalahan maka ada di dalam *blocktry* and *catch*.Kedua, kesalahan yang muncul akan dianggap sebagai *object* dan ditangkap *catch* kemudian di *assign* ke *variable* kesalahan dengan tipe *Exception*. Ketiga, perintah setelah munculnya kesalahan pada *blocktry* tidak akan dieksekusi.

- ✓ **Catch**

Jika anda sudah melihat contoh *try* maka secara tidak langsung anda sudah memahami kegunaan dari keyword ini. Dalam java, keyword *catch* harus dipasangkan dengan *try*. Kegunaan keyword ini adalah menangkap kesalahan atau *bug* yang terjadi dalam *blocktry.* Setelah menangkap kesalahan yang terjadi maka *developer* dapat melakukan hal apapun pada *blockcatch* sesuai keinginan *developer*.

Contoh Program :

Listing Program

```
1  catch(Exception kesalahan)
2  {
3  System.out.println("mohon maaf, terdapat kesalahan
pada program");
4  //lakukan hal lainnya disini
5 }
```

Keywordcatchjuga dapat diletakan berulang-ulangsesuai dengan kebutuhan.

Contoh :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
```

```
4  try
5  {
6 int a = 1/0;  //berpotensi untuk menimbulkan kesalahan
yaitu pembagian dengan bilangan 0
7  System.out.println("perintah selanjutnya");
8  }
9  catch(NullPointerException e)
10 {
11 }
12 catch(ArrayIndexOutOfBoundsException e)
13 {
14 }
15 catch(Exception e)
16 {
17 }
18 }
19 }
```

- ✓ **Finally**

Keywordfinally merupakan keyword yang menunjukan bahwa *blockprogram* tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada. Setiap try membutuhkan sekurang-kurangnya satu bagian catch atau finally yang cocok. Jika tidak mendapatkan bagian catch yang cocok, maka bagian finally akan dieksekusi sebelum akhir program, atau setiap kali suatu method akan kembali ke pemanggilnya, melalui exception yang tidak dapat ditangkap, atau melalu pernyataan return, bagian finally akan dieksekusi sebelum kembali ke method lagi.

Contoh implementasinya pada program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
```

```
6  int a = 1/0; }q
7  finally
8  {
9  System.out.println("terima  kasih  telah  menjalankan
program");
10  }
11  }
12  }
```

Output Program diatas:

Output

terima kasih telah menjalankan program

Jika saya lakukan modifikasi program diatas menjadi :

Listing Program

```
1   public class A
2   {
3   public static void main(String[] args) {
4   try
5   {
6   int a = 1/0;
7   }
8   catch (Exception e)
9   {
10  System.out.println("ada kesalahan yang muncul");
11  }
12  finally
13  {
14   System.out.println("terima  kasih  telah  menjalankan
program");
15  }
16  }
17  }
```

Output Program :

Output

```
ada kesalahan yang munculterima kasih telah menjalankan program
```

Perhatikan kedua contoh diatas, *blockfinally* akan selalu dieksekusi meskipun adanya kesalahan atau tidak pada *blocktry*.Berbeda dengankeyword**catch,** keyword**finally** hanya dapat diletakan 1 kali setelah keyword**try**.

- ✓ **Throw**

KeywordThrow digunakan untuk melemparkan suatu *bug* yang dibuat secara manual.

Contoh program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  throw new Exception("kesalahan terjadi");
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 }
```

Output Program :

Output

```
java.lang.Exception: kesalahan terjadi
```

Seperti yang anda lihat pada program diatas, pada *keyword***throw new Exception("kesalahan terjadi");**akan melempar *object* bertipe **exception**yang merupakan *subclass* dari *class***Exception** sehingga akan dianggap sebagai suatu kesalahan yang harus ditangkap oleh keyword**catch.**

Perhatikan contoh berikut ini :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6   throw new B(); //cobalah  ganti baris ini dengan à
new B();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 }
14 class B extends Exception
15 {
16 B()
17 {
18 }
19 public String toString()
20 {
21 return "object dengan tipe kelas B";
22 }
23 }
```

Output Program :

Output

```
object dengan tipe kelas B
```

Program diatas telah mendefinisikan suatu kelas B *mengextends* dari kelas *Exception*. Ketika kita melakukan *thrownewB();* maka object dari kelas bertipe B ini akan dianggap kesalahan dan ditangkap oleh *blockcatch*. Sekarang jika anda menghilangkan keywordthrow apa yang terjadi?

.

- ✓ **Throws**

Keywordthrows digunakan dalam suatu method atau kelas yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya.Cara mendefinisikannya dalam method adalah sebagai berikut : <*method modifier*> *type method-name throws* **exception-list1, exceptio-list2, …** {}.

Contoh Program :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  f();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 public static void f() throws NullPointerException,
ArrayIndexOutOfBoundsException
14 {
15 //implementasi method
16 throw new NullPointerException();
```

```
17 //throw new ArrayIndexOutOfBoundsException();
18 }
19 }
```

Output Program :

Output

```
java.lang.NullPointerException
```

Contoh program lainnya :

Listing Program

```
1  public class A
2  {
3  public static void main(String[] args) {
4  try
5  {
6  f();
7  }
8  catch(Exception e)
9  {
10 System.out.println(e);
11 }
12 }
13 public static void f() throws NullPointerException,
ArrayIndexOutOfBoundsException
14 {
15 //implementasi method
16 //throw new NullPointerException();
17 throw new ArrayIndexOutOfBoundsException();
18 }
19 }
```

Output Program :

Output

```
java.lang.ArrayIndexOutOfBoundsException
```

Perhatikan kedua contoh pengguaan keyword*throws* pada *method*. Ketika method tersebut dipanggil dalam *blocktry*. Maka method tersebut akan membuat *object* yang merupakan *subclass* dari *classThrowable* dan methodtersebut akan melemparkan kesalahan yang ada dalam *blockmethod* kedalam *blocktry*. Di dalam *blocktry*, kesalahan tersebut kemudian ditangkap kedalam *blockcatch*.

C. Rangkuman

Exception adalah sebuah peristiwa yang menjalankan alur proses normal pada program. Peristiwa ini biasanya berupa kesalahan(error) dari beberapa bentuk. Ini disebabkan program kita berakhir tidak normal. Untuk menangani exception dalam java, kita gunakan blok try-catch finnaly. Ada lima keywords yang digunakan oleh Java untuk menangani exception ini, yaitu :Try, catch, finally, throw, throws.

Semua class exception terdapat dalam package java.lang. Superclass tertinggi adalah class Throwable, tetapi class ini hampir tidak pernah digunakan secara langsung.

Class Error merupakan tipe exception yang seharusnya tidak ditangani dengan menggunakan blok try catch karena berhubungan dengan Java run-time system/evironment. Sedangkan Class Exception merupakan tipe exception yang sebaiknya ditangani oleh program kita secara langsung.

Manfaat dari penggunaan Exception adalah pemisahan antara pendeteksian error(dilakukan didalam suatu metode yang dipanggil) dengan penanganan error (dilakukan didalam metode pemanggil).

D. Tugas

Tugas 1

Tulislah suatu program yang meminta user untuk memasukkan 2 integer dan menampilkan penjumlahan atas keduanya. Program anda harus meminta pengguna memasukkan 2 integer kembali bila inputan sebelumnya tidak tepat. Tipe exception yang digunakan adalah **NumberFormatException**.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	

7.	q
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari beberapa istilah berikut :
   a. Try
   b. Catch
   c. Finally
2. Apa kegunaan dari keyword Throw dan Throws ?
3. Bagaimana cara melemparkan suatu exception? Apakah boleh melempar beberapa exception sekaligus menggunakan satu statement **throw**?
4. Apa keluaran (output) dari kode berikut ini ?

Listing Program

```
1 public class Test{
2public static void main (String [] args){
3   try{
4   int nilai = 30;
5   if (nilai < 40)
```

```
6  throw new Exception ("nilai terlalu kecil");
7  }
8  catch (Exception ex){
9  System.out.println (ex.getMessage());
10 }
11  System.out.println ("Lanjut setelah blok
catch");
12  }
13  }
```

Apakah keluaran kode bila baris

`Int nilai = 30 ;` Diganti dengan `Int nilai = 50 ;`

5. Jelaskan perbedaan antara catch dan throw !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a)Try

.....

.....

.....

.....

b) Catch

.....

.....

.....

.....

c)Finally

.....

.....

.....

.....

LJ- 02 : Kegunaan dari keyword Throw dan Throws :

.....

.....

.....

.....

.....

.....

LJ- 03 : Cara melempar suatu Exception :

.....

.....

.....

.....

.....

.....

.....

LJ- 04: Keluaran (output) dari kode :

.....

.....

.....

Bila diganti dengan (`nilai int = 50`) :

.....

.....

.....

LJ- 05: Perbedaan catch dan throw :

.....

.....

.....

.....

.....

G. Lembqar Kerja Siswa

4. Kegiatan 4 : String

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 4 ini siswa diharapkan dapat :

1) Memahami mekanisme penangganan kesalahan.

2) Menyajikan bermacam-macam cara untuk mencari tipe kesalahan.

B. Uraian Materi

1) Pengolahan String

Dalam pemrograman Java string merupakan aspek penting, karena dapatmempelajari mengenai class dan objek melalui penggunaan string. String sebenarnyamerupakan class yang terdapat dalam library Java. Java String merupakan salah satukelas dasar yang disediakan oleh Java untuk memanipulasi karakter.

- ✓ **Membuat Objek String**

Java mendefinisikan class String dalam package java.lang.String, sehingga tidak perlu melakukan impor secara eksplisit. Java String digunakan untuk mendefinisikan string yang konstant (tidak bisa berubah

- ✓ **Menggabungkan String**

Seringkali dalam pemrograman kita perlu menggabungkan String untuk mendapatkan String baru. Kita dapat menggunakan operator (+) untuk menggabungkan beberapa String.

Contoh penggunaan :

Stringku = “Ini adalah contoh”+”penggabungan String”;

Kita juga dapat menggunakan operator += untuk menggabungkan variabel String,

misalnya seperti contoh berikut :

String kata = ”Ini perkataan”;

Kata += “saya sendiri”;

✓ **Menentukan Awal Dan Akhir String**

Untuk menentukan awal dan akhir String, kita dapat menggunakan dua fungsi utama, yaitu : startsWith(String s)

Dengan fungsi ini, maka objek String yang bersangkutan akan diperiksa,
apakah diawali oleh objek String s, pada parameter fungsi ini endsWith(String s)

Dengan fungsi ini, maka objek string yang bersangkutan akan diperiksa,
apakah diakhiri oleh objek string s, pada parameter fungsi ini.

Fungsi diatas akan menghasilkan nilai boolean ***true*** bila benar dan ***false*** bila salah.

✓ **Mengurutkan String**

Dapat juga melakukan pengurutan string dengan method compareTo().Method ini membandingkan karakter-karakter pada String secara berurutan dari awal String. Misalnya string pertama bernilai “a” dan string kedua bernilai “b”, maka apabila Stringpertama.compareTo(Stringkedua) akan menghasilkan nilai negatif (<0)dan apabila dilakukan sebaliknya akan menghasilkan nilai positif (>0). Nilai 0 akandihasilkan apabila string pertama dan kedua sama.

✓ **Mendapatkan Panjang String**

Kita dapat memperoleh panjang string dengan menggunakan method length(); seperti contoh berikut ini :

String panjang = “ini panjangnya 17”;
System out.println(panjang.length());

✓ **Mencari Posisi Karater Atau SubString Dari String**

Ada dua method yang dapat digunakan untuk mencari posisi karakter dari string dan dua method untuk mendapatkan posisi subString dari string.

✓ **Method untuk mencari posisi karakter pada String :**

*IndexOfchar (karakter)*Memerlukan argumen berupa karakter dan akan mengembalikan nilai posisiindeks dari karakter yang dicari. Posisi yang dikembalikan adalah posisipertama dari karakter yang ditemukan. Bila karakter

tidak ditemukan, makaakan mengembalikan nilai -1.IndexOf(char karakter, int indeks) sama dengan sebelumnya, tetapi memerlukan argumen tambahan, yaitu indeks posisi awal pencarian dalam integer.

- ✓ **Method untuk mencari posisi subString pada String :**

indexOf(String Str)Penggunaan dan fungsi sama dengan method untuk char.
indexOf(String str, ont indeks)Penggunaan dan fungsi sama dengan method untuk char.

2) StringBuffer

StringBufferadalah pasangan class String yang menyediakan banyak fungsi string yang umum. StringBuffer merepresentasikan urutan karakter yang dapat dikembangkan dan ditulis ulang. StringBuffer dapat disisipi karakter dan subString di tengahnya, atau ditambah di belakangnya. StringBuffer memiliki default kapasitas 16 karakter, tapi biasanya ukuran diatur sendiri dengan mendefinisikan kapasitas pada saat pembuatan.
Misalnya adalah sebagai berikut:StringBuffer baru = new StringBuffer(50); Contoh diatas merupakan StringBuffer kosong dengan kapasitas 50 karakter. Ada 3 cara untuk mengefinisikan StringBuffer :

- **StringBuffer baru = new StringBuffer()**

secara tidak langsung variabel baru akan menjadi objek StringBuffer dengan ukuran 16 karakter karena defaultnya adalah 16 karakter

- **StringBuffer baru1 = new StringBuffer(50)**

objek baru1 merupakan StringBuffer dengan panjang karakter 50

- **StringBuffer baru2 = new StringBuffer(*String*)**

objek baru2 merupakan objek StringBuffer dengan panjang karakter ***String* + 16 karakter.**

Berikut merupakan contoh deklarasi StringBuffer :

Listing Program

```
1  public class SB{
2  public static void main (String args[]){
3  String kata = "Java";
4  StringBuffer baru = new StringBuffer();
```

```
5  StringBuffer baru1 = new StringBuffer(50);
6  StringBuffer baru2 = new StringBuffer(kata);
7  System.out.println("baru    : "+baru.capacity());
8  System.out.println("baru1   : "+baru1.capacity());
9  System.out.println("baru1   : "+baru2.capacity());
10 }
11 }
```

Berbeda denga type data primitif yang lain, type data String dalam Java diperlakukan sebagai object. Object String dalam Java dapat dibuat dengan dua cara, yaitu:

Penulisan sesuatu di dalam tanda antara petik ganda (literal String). Cara ini digunakan untuk mengakomodasi kebiasaan dari bahasa C/C++

- String s = ”Hello World”;
- System.out.println(“Hello World”);

Pembuatan object String dengan keyword new.

- String s = new String(“Halo”);

Class String mempunyai atribut final, sehingga Class String tidak dapat di-extends / diturunkan.

Dalam Java, terdapat dua jenis memory yaitu:

- Stack (tempat local variable dan tumpukan method)
- Heap (tempat instance variable dan object), Di dalam heap terdapat bagian memory yang disebut dengan **String constant pool.**

Bila kita membuat object String dengan penulisan sesuatu di antara tanda petik ganda (literal String), maka object String tersebut akan berada di dalam String constant pool. Sedangkan bila kita membuat String dengan keyword new, maka object String tersebut akan berada di dalam heap (tetapi diluar String constant pool). Khusus untuk pembuatan object String dengan keyword **new** (ex: String sample = new String(“Hello World”)), sebenarnya terdiri dari 3 buah proses (yang melibatkan 2 buah object String), yaitu :

- Pembuatan object String “Hello World” di dalam String constant pool. Hal ini karena “Hello World” adalah literal String yang otomatis membuat object di String constant pool.
- Pembuatan object String “Hello World” di dalam heap (non constant pool).

- Penghapusan object String “Hello World” di dalam String constant pool (bila tidak ada yang mereferensi String) ini.

Pada saat java menjumpai literal String (dalam kode program), maka java akan mencari String yang sama dengan literal String tersebut di dalam String constant pool. Bila ternyata di dalam pool ditemukan object String yang sama, maka reference akan menunjuk pada object String di dalam pool tersebut. Bila ternyata java tidak menemukan di dalam String constant pool, maka java akan membuat object String di dalam String constant pool terlebih dahulu.

Object String adalah immutable (tidak dapat diubah) . Pengertian tidak dapat diubah adalah, sekali sebuah object String berisi suatu nilai, maka nilai tersebut tidak dapat diubah (tidak peduli apakah object String tersebut berada pada heap ataupun String constant pool). Dalam praktek pemrograman, kita merasa bahwa object String dapat berubah, hal ini karena yang berubah adalah nilai reference penunjuk object String bukan object String tersebut.

String Class sangat tidak efektif bila kita ingin melakukan banyak modifikasi terhadap suatu String object, hal ini karena sifat dari String Class yang immutable (banyak modfikasi pada suatu kelas String akan dapat menyebabkan banyaknya object String yang terlibat). StringBuffer dan StringBuilder Class mengatasi permasalahan ini (mutable). StringBuffer thread safe sehingga dapat menjamin konsistensi operasi pada String object.

Pada String Class, method-method akan mengembalikan object String baru (hasil modifikasi) tanpa mengubah object String tempat method dipanggil (karena immutable)Pada kelas StringBuffer Class method-method akan memodifikasi object tempat method dipanggil, dan kemudian mengembalikan object tersebut sebagai return value dari method.Method utama pada StringBuffer adalah append dan insert method.Dan dengan Class StringBuffer kita dapat melakukan chaining method.

3) Konstruktor

Konstruktor pada Java merupakan method khusus yang dipakai oleh Java untuk membuat sebuah object didalam kelas dan tiap kelas boleh memiliki lebih dari satu konstruktor.

Karakteristik konstruktor :

1. Nama Konstruktor = Nama Kelas.
2. Tidak mengembalikan nilai atau return value termasuk void.
3. Cara menggunakan konstruktor adalah dengan menggunakan kata kunci **new**. Jika didalam kelas tidak dituliskan konstruktor, Java akan secara default menambahkan konstruktor kosong kedalam kelas tersebut.

✓ **Pemanggilan Konstruktor**

Membuat konstruktor

Listing Program

```
1  public class Constr{
2  Constr ()
3  {
4  }
5  }
```

Konstruktor dipanggil saat membuat sebuah object. Sama seperti membuat object pada class.

Contoh :

Listing Program

```
1  public class Constructor_1{
2  float nilaiAkhir;
3  public Constructor_1(int nilai_akhir){
4  nilaiAkhir=nilai_akhir;
5  }
6  public String grade(){
7  String nilaigrade;
8  if(nilaiAkhir>=50)
9  nilaigrade=”Bagus”;
10 else
11 nilaigrade=”Jelek”;
12 return nilaigrade;
13 }
14 public void cetak(){
```

```
15 System.out.println("Grade nilainya = "+grade());
16 }
17 public static void main (String [] args){
18 Constructor_1 hasil = new Constructor_1(67);
19 hasil.cetak();
20 }
21 }
```

- ✓ **Overload Konstruktor**

Sebuah class mungkin memiliki lebih dari satu konstruktor dengan parameter yang berbeda satu sama lainnya.

Contoh :

Listing Program

```
1  public class Constructor_2{
2  float nilai1, nilai2;
3  public Constructor_2(int bil1){
4  nilai1 = bil1;
5  }
6  public Constructor_2(int bil1, bil2){
7  nilai = bil1+bil2;
8  }
9  public String grade()
10 {
11 String nilaigrade;
12 if(nilai1>=50)
13 nilaigrade = "Bagus";
14 else
15 nilai grade = "Jelek";
16 return nilaigrade;
17 }
18 public void cetak(){
19 System.out.println("Grade nilainya = "+grade());
20 }
21 public static void main (String [] args){
```

```
22 Constructor_2 hasil = new Constructor_2 (45);
23 hasil.cetak();
24 Constructor_2 hasilnya = new Constructor_2 (45,35);
25 hasilnya.cetak();
26 }
27 }
```

4) Modifier

Modifier digunakan untuk menentukan sifat dari suatu kelas dan menentukan preveledge (hak akses) dari kelas lain. Modifier juga digunakan untuk menentukan relasi (extend atau implements) dengan kelas lainnya. Berikut ini adalah wilayah modifier akses :

Wilayah Akses	Public	Protected	Default	Private
Dikelas yang sama	✓	✓	✓	✓
Beda kelas, di package yang sama	✓	✓	✓	×
Beda kelas, beda package, dikelas turunan	✓	✓	×	×
Beda kelas, beda package, tidak di kelas turunan	✓	×	×	×

Keterangan :

1. Public : menyatakan bahwa kelas/ method/ attribute dapat diakses oleh kelas lain dimanapun letaknya.
2. Protected : menyatakan bahwa kelas/ method/ attribute tersebut dapat diakses oleh kelas lain yang berada dalam satu package atau kelas lain tersebut merupakan turunannya.
3. Private : menyatakan bahwa kelas tersebut tidak dapat diakses sama sekali oleh kelas lain bahkan tidak dapat diakses oleh kelas turunannya. Attribute yang bersifat private hanya dapat diakses oleh method dalam kelas yang sama, kelas lain masih dapat mengakses melalui method tersebut asal method tersebut modifiernya public.

5) Class String

String pada java adalah object dan sifatnya read-only (immutable).Karena sifat immutable ini setiap perubahan terhadap isi string akan dibuat string baru untuk menampung perubahan tersebut.

Contoh :

String s1 = “hello world”;

S1 = “hello java”;

Akan menyebabkan dua kali pengalokasian memory untuk dua objek string tersebut diatas, dengan referensi terakhir s1 ke string “hello java”.

✓ Operator Pada String

Anda dapat menggunakan operator + untuk mengabungan dua string atau string dengan type data primitive lainnya (menyebabkan konversi otomatis dari type lain ke string).

Contoh :Str = “Jumlah data “ + jd;

✓ Method pada class String

Str.length()

Str.compareTo(String Str1)

Str.compareToIgnoreCase(String Str1)

Str.concat(String Str1)

Str.equals(String Str1), membandingkan isi Str sama dengan Str1

Str.equalsIgnoreCase(String Str1)

Str.startsWith(String prefix)

Str.startsWith(String prefix, int offset)

Str.endsWith(String suffix)

Str.indexOf(int ch)

Str.indexOf(int ch, int fromIndex)

Str.indexOf(String str)

Str.indexOf(String str, int fromIndex)

Static String.valueOf(value), mengembalikan string dari suatu nilai Boolean, char, char array, int, long, double, float.

Untuk pencarian dari akhir string gunakan lastIndexOf

Str.trim(), menghapus white space diawal dan akhir string

Str.toLowerCase()
Str.toUpperCase()
Str.charAT()
Contoh : Metode Enkripsi Ceasar secara substitusi. M = (C+3) Mod 26

- ✓ **Class StringBuffer**

StringBuffer merupakan string yang sifatnya mutable yang isinya dapat dimanipulasi, sebingga sangat menghemat system resource dan performance yang lebih baik pada operasi yang banyak melakukan manipulasi terhadap isi string.
StringBuffer sb =new StringBuffer() ; //StringBuffer dengan kapasitas 16
StringBuffer sb = new StringBuffer(30); //StringBuffer dengan kapasitas 30StringBuffer sb = new StringBuffer(“HendraSoewarno”);

- ✓ **Method pada class StringBuffer**

Sb.ensureCapacity(int kapasitas), set kapasitas StringBuffer ke ukuran baru.
Sb.append(value).
Sb.append(Char[] str, int offset, int len)
Sb.capacity(), mengembalikan ukuran kapasitas StringBuffer
Sb.charAt(int index), mengembalikan karakter pada posisi tertentu dari StringBuffer, index dimulai dari 0 (nol).
Sb.delete(int start, int end), menghapus sejumlah karakter dari StringBuffer
Sb.deleteChatAt(int index)
Sb.getChars(int srcBegin, int srcEnd, Char[] dst, int dstBegin)
Sb.insert(int offset, value)
Sb.length()
Sb.reverse()
Sb.setCharAt(int index, char ch)
Sb.setLength(int newLength)
Sb.substring(int start)
Sb.substring(int start, int end)
Sb.toString()

Contoh :

Listing Program

```
1  class C3 {
2  protected String plainText;
3  C3() {
4  plainText = "";
5  }
6  void setPlainText(String newValue) {
7  plainText = newValue;
8  }
9  String getCipherText() {
10 StringBuffer result = new StringBuffer("");
11 char huruf;
12 for (int i=0; i < plainText.length(); i++) {
13 if (plainText.charAt(i) != 32) {
14 huruf = (char)((plainText.charAt(i) - 'A' + 3) % 26 +
'A');
15 result.append(huruf);
16 }
17 }
18 return result.toString();
19 }
20 }
```

Contoh dengan Argument String

Listing Program

```
1  class testArgs {
2  public static void main(String[] args) {
3  C3 c = new C3();
4  if (args.length < 1)
5      System.out.println("Usage:      java      testArgs
<plaintext>");
6  else {
7  c.setPlainText(args[0]);
```

```
8         System.out.println("Hasil         adalah         :
"+c.getCipherText());
9   }
10  }
11  }
```

Untuk menjalankan program diatas gunakan :

java testArgs <plaintext>

Contoh dengan GUI

Listing Program

```
1   import javax.swing.*;
2   import java.awt.*;
3   import java.awt.event.*;
4   class testString {
5   public static void main(String[] args) {
6   final C3 c = new C3();
7   final JFrame frame = new JFrame("Enkripsi RC3");
8   JLabel label = new JLabel("Plaintext :");
9   final JLabel labelh = new JLabel("Hasil :");
10  final JTextField textField = new JTextField(20);
11  JButton button = new JButton("Encrypt");
12  button.setMnemonic('E');
13  button.addActionListener(new ActionListener() {
14  public void actionPerformed(ActionEvent e) {
15  c.setPlainText(textField.getText());
16  labelh.setText("Hasil adalah : "+c.getCipherText());
17  frame.pack();
18  }
19  });
20  frame.getContentPane().setLayout(new FlowLayout());
21  frame.getContentPane().add(label);
22  frame.getContentPane().add(textField);
```

```
23 frame.getContentPane().add(button);
24 frame.getContentPane().add(labelh);
25 frame.pack();
26 frame.show();
27 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28 }
29 }
```

- ✓ **Class StringBuilder**

Kelas StringBuilder menyediakan beberapa method ter-overload untuk menyambung Boolean, char, array char, double, float, int, long, dan String kedalam suatu StringBuilder. Contohnya seperti pada kode dibawah ini menyambungkan String dan char kedalam StringBuilder untuk membentuk suatu String baru.

Listing Program

```
1  stringBuilder stringBuilder = new StringBuilder();
2  stringBuilder.append("JAVA");
3  stringBuilder.append(' ');
4  stringBuilder.append("itu");
5  stringBuilder.append(' ');
6  stringBuilder.append("Tangguh !");
```

Kelas StringBuilder juga dapat menyisipkan boolean, char, array char, double, float, int, long, dan String. Contohnya :

Listing Program

```
1  stringBuilder.insert (5, "dan HTML");
```

Beberapa karakter dapat dihapus dari suatu String dalam StringBuilder menggunakan dua method **delete**, membalikkan string menggunakan method **reserve,** mengganti karakter menggunakan method **replace**, atau menetapkan suatu karakter baru didalam suatu String menggunakan method **setCharAt.**

Contoh :

Listing Program

```
1   stringBuilder.delete (4,7) mengubah builder menjadi
JAVA Tangguh!.
2    stringBuilder.deleteCharAt  (5)  mengubah  builder
menjadi JAVA tu Tangguh!.
3   stringBuilder.reserve  ()  mengubah  builder  menjadi
!huggaT uti AVAJ
4    stringBuilder.replace  (0,  3,  "HTML")  mengubah
builder menjadi HTML itu Tangguh !.
5   stringBuilder.setCharAt(0,  'j')  menetapkan  builder
menjadi jAVA itu Tangguh!.
```

C. Rangkuman

String merupakan class yang terdapat dalam library java. Java string merupakan salah satu kelas dasar yang disediakan oleh java untuk memanipulasi karakter. String bersifat immutable, sehingga perubahan terhadap isi String akan dibuat String baru untuk menampung perubahan tersebut.

StringBuffer merepresentasikan urutan karakter yang dapat dikembangkan dan ditulis ulang. StringBuffer dapat disisipi karakter dan sub String ditengah atau dibelakang. Default StringBuffer memiliki 16 karakter, namun ukuran default ini dapa diatur dengan mendefinisikan kapasitas saat pembuatan.StringBuffer bersifat mutable yang isinya dapat dimanipulasi, sehingga dapat menghemat resource dan performance lebih baik pada operasi yang memanipulasi isi String.
Konstruktor merupakan method khusus yang dipakai oleh java untuk membuat sebuah objek didalam kelas, dan setiap kelas boleh memiliki lebih dari satu kontruktor.

D. Tugas

Tugas 1

Buatlah suatu metode yang mencari jumlah huruf didalam String menggunakan header berikut ini :

```
public static int hitungHuruf (String s)
```

Buat program uji yang meminta pengguna untuk memasukkan suatu string dan yang menampilkan jumlah huruf didalam string.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	

6.	
7.	
8.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa perbedaan konstruktor dengan method?

2. Apakah output dari kode berikut ini !

Listing Program

```
1  String s1 = "JAVA itu tangguh!";
2  String s2 = s1.replace ("V" , "abc");
3  System.out.println(s1);
4  System.out.println(s1);
```

3. Sebutkan dan jelaskan 3 konstruktor dalam StringBuffer !

4. Tuliskan 3 statement untuk membalik string s menggunakan method reserve dalam kelas StringBuilder !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Perbedaan konstruktor dengan method

.....
.....
.....
.....
.....
.....

LJ- 02 : Output dari kode :

.....
.....
.....
.....
.....
.....

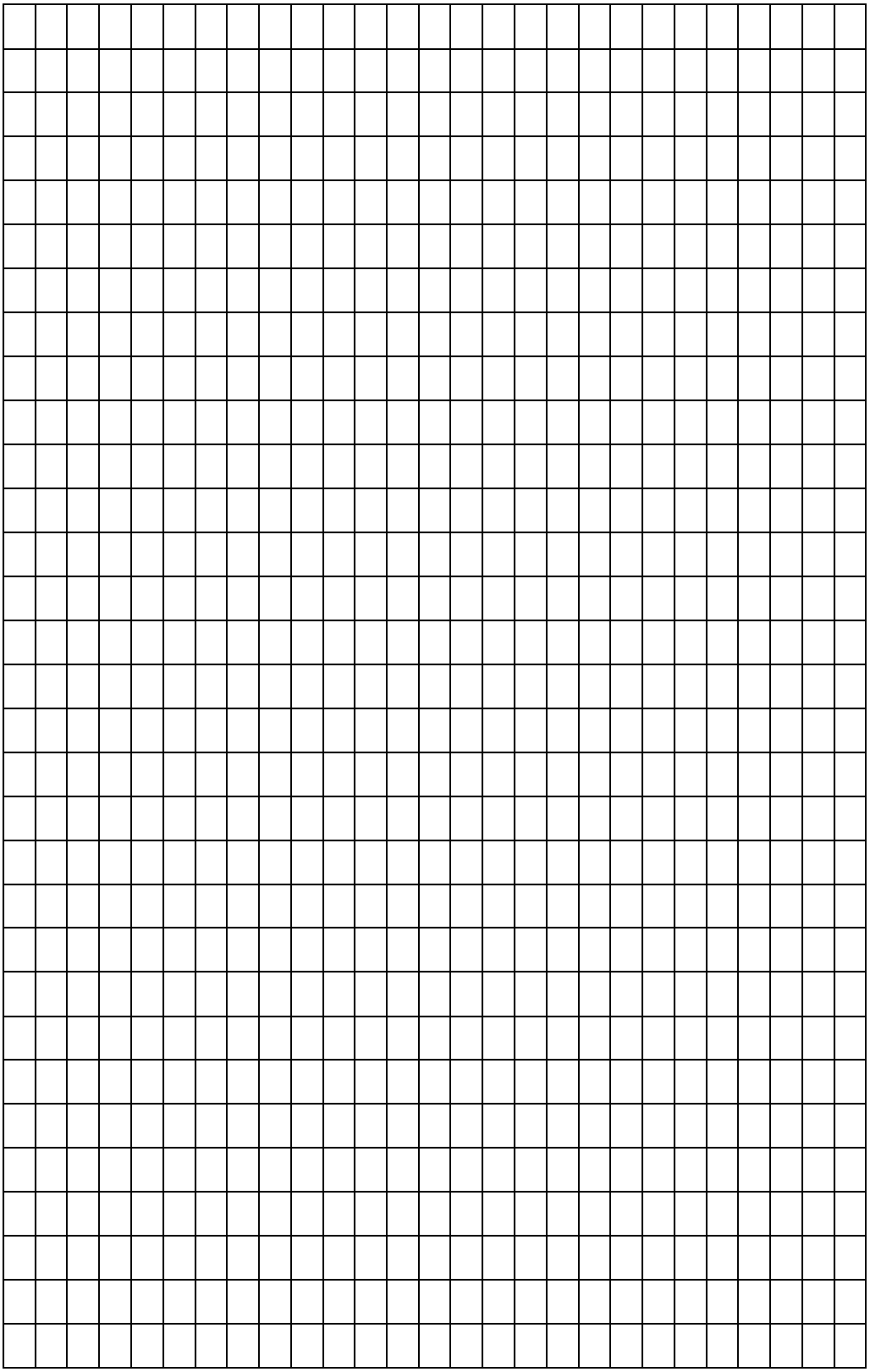
LJ- 03 : 3 konstruktor dalam StringBuffer

.....
.....
.....
.....
.....
.....
.....

LJ- 04: 3 Statement untuk membalik string s menggunakan method reserve :

.....
.....
.....
.....
.....

G. Lembar Kerja Siswa



5. Kegiatan 5 : Array

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 5 ini siswa diharapkan dapat :

1) Memahami data collection sebagai media penyimpanan
2) Menyajikan data collection sebagai penyimpan data

B. Uraian Materi

1) Deklarasi Array

Array digunakan untuk membuat variabel bisa menampung beberapa data dengan tipe data yang sama alias satu tipe data. Ciri khas varibel yang menggunakan array adalah terdapat simbol **[].** Unyuk mengetahui panjang array yang telah kita buat, kita dapat memakai property *length***,** dengan format sebagai berikut :

`var_array.length` → total elemen array pada dimensi 1.

`var_array[i].length` → total elemen array pada dimensi 2 untuk indeks ke-i pada dimensi 1.

`var_array[i][i].length` → total elemen array pada dimensi 3 untuk indeks ke-I pada dimensi 1 dan indeks ke-I pada dimensi 2dan seterusnya.

Isi dari suatu array dapat kita copy pada array yang lain dengan memanfaatkan method arraycopy() pada class System. Format penulisannya sebagai berikut :

```
System.arraycopy (array1, p1, array2, p2, n);
```

Array memiliki dua tipe, yaituSingle Dimension dan Multi Dimension, sedangkan Proses pendeklarasian variabel yang menggunakan array ada 2 tipe :

- ✓ **Array Single Dimension**
  - ○ **Pendeklarasian Variabel Menggunakan Array Tipe Pertama**

Variabel yang menggunakan array, array ditentukan length/panjangnya terlebih dahulu, sehabis itu baru di inisialisasi.

Contoh

:

Listing Program

```
1 public class Array1{
2 public static void main (String args []) {
3 int nilai [] = new int [3];
4 nilai [0] = 70;
5 nilai [1] = 80;
6 nilai [2] = 65;
7 double ratarata = 0.0;
8 for (int i=0; i<nilai.length; i++)ratarata+=nilai [i];
9 ratarata/=nilai.length;
10 System.out.println("Nilai rata-rata = "+ ratarata);
11 }
12 }
```

Perlu diingat bahwa index array dimulai dari 0.

- **Pendeklarasian Variabel Menggunakan Array Tipe Kedua**

Variabel yang menggunakan array langsung diinisialisasi.

Contoh :int[] nilai={50,60,70,80};

Lantas bagaimana caranya agar kita mengetahui panjang arraynya, di atas variabel nilai menggunakan array langsung diinisalisasi jika ingin mengetahui panjang arraynya **nilai.length.**

- ✓ **Array MultiDimension**

Pada konsep array multidimension, konsepnya sama seperti baris dan kolom pada tabel.

- **Pendeklarasian Variabel Menggunakan Array Tipe Pertama**

Sama seperti pada pendeklarasian variabel menggunakan array single dimension, perbedaannya kita harus menentukkan panjang kolomnya, jadi kita menentukan panjang baris dan panjang kolomnya.

Contoh :

Listing Program

```
1  import java.text.NumberFormat;
2  public class Array2{
```

```
3  public static void main (String args []){
4  NumberFormat nf = NumberFormat.getInstance();
5  nf.setMaximumFractionDigits(3);
6  int nilai [][]=new int [2][3];
7  nilai [0][0]=85;
8  nilai [0][1]=81;
9  nilai [0][2]=78;
10 nilai [1][0]=65;
11 nilai [1][1]=73;
12 nilai [1][2]=71;
13 String MK[]={"RPL","PBO"};
14 double rataratаMK[]= new double[nilai.length];
15 for (int i=0; i<nilai.length; i++){
16 for (int j=0; j<nilai[0].length; j++){
17 rataratаMK[i]+=nilai [i][j];
18 }
19 }
20 rataratаMK[i]/=nilai[0].length;
21 }
22 System.out.println("Nilai Mata Pelajaran\n");
23   System.out.println   ("MK\   tMinggu1   \   tMinggu2\
tMinggu3\ tRata-Rata");
24 for (int i=0; i<nilai.length; i++){
25 System.out.print(MK[i]+"\t");
26 for (int j=0;j<nilai[0].length; j++){
27 }
28 System.out.print(nf.format(rataratаMK[i]+"\n");
29 }
30 }
31 }
```

Seperti biasa indeks baris dimulai dari 0 dan indeks kolom dimulai dari 0.

- **Pendeklarasian Variabel Menggunakan Array Tipe Kedua**

Tipe kedua, proses pendeklarasian langsung dengan inisialisasinya :

Listing Program

```
char [][] abjad={{'A','B'},{'C','D'},{'E','F'}};
```

Untuk mengetahui panjang baris **abjad.length,** sedangkan untuk mengetahui panjang kolom bisa gunakan **abjad.length[0],** Kenapa harus 0 parameter untuk mengetahui panjang kolomnya?, alasannya karena sudah pasti indeks array dimulai dari 0.

- ✓ **Membuat Array**

Setelah mendeklarasikan array, maka perlu dibuat array nya terlebih dahulu.Tidak seperti mendeklarasikan variabel tipe data primitif, deklarasi suatu variabel array tidak mengalokasikan memori untuk array tersebut.Kita tidak dapat menugaskan elemen-elemen kepada suatu array kecuali jika array tersebut telah diciptakan. Setelah suatu variabel array diciptakan, kita dapat menciptakan suatu array menggunakan operator **new**dengan sintaks seperti berikut :

```
varRefArray = new tipeElemen[ukuranArray];
```

Statemen ini melakukan dua hal, yaitu menciptakan suatu array menggunakan **new tipeelemen[ukuranArray];** dan menugaskan array yang baru saja diciptakan kepada **varRefArray.**

- ✓ **Ukuran Array dan Nilai Default**

Agar memori untuk suatu array dialokasikan, ukuran array harus diberikan untuk menentukan jumlah elemen yang dapat disimpan didalam array itu.Ukuran array tidak dapat diubah setelah array diciptakan.

Contoh :

TipeArray namaArray[];

namaArray = new TipeArray [jumlah];

Listing Program

```
1 int nilai [];
2 nilai = new int [5];
```

Dalam program tersebut '[jumlah]' merupakan jumlah data yang dapat ditampung oleh array.

- ✓ **Memproses Array**

Ketika memproses elemen-elemen array, seringkali kita menggunakan loop for, karena semua elemen suatu array bertipe sama, dan ukuran array yang telah diketahui, maka sangat cocok untuk menggunakan loop for.

Sebagai contoh :

Listing Program

```
1 double [] myList = new double [10];
```

Berikut ini adalah contoh pemrosesan array :

1. Loop berikut ini menginisialisasi array myList dengan nilai-nilai dari pengguna:

Listing Program

```
1 java.util.Scanner masukkan = new java.util.Scanner
(System.in);
2 System.out.print("masukkan " +myList.length+ "buah
nilai : ");
3 for (int i=0; i<myList.length;i++)
4 myList[i]=masukkan.nextDouble();
```

2. Loop berikut ini menginisialisasi array myList dengan nilai-nilai acak antara 0.0 sampai 100.0, tetapi kurang dari 100.0:

Listing Program

```
1 for (int i=0; i<myList.length; i++){
2 myList [i]= Math.random()*100;
```

3. Untuk menampilkan suatu array, kita harus menampilkan setiap elemen array menggunakan suatu loop sebagai berikut :

Listing Program

```
1 for (int i=0; i<myList.length; i++){
2 System.out.print (myList[i] +" ");
3 }
```

2) Collection

Collection merupakan istilah umum yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan suatu teknik tertentu pula. Semua class yang berhubungan dengan pengelompokan objek ini dalam java tergabung dalam *Java Collection Framework*, dimana Framework ini diletakan dalam package java.util dan mempunyai dua interface utama, yaitu collection dan map.

Jenis Pengelompokan Collectin ini merupakan pengelompokan satu dimensi. Berdasarkan teknik pengelompokannya terbagi menjadi tiga kelompok yaitu set, list, dan queue. Semua class dalam kelompok collection ini merupakan implementasi dari interface yang sama, maka pada dasarnya cara penggunan class ini adalah sama. Yang membedakannya hanyalah kapan kita harus menggunakannya karena hal ini sangat terkait dengan kebutuhan.

3) Set

Merupakan pengelompokan mengikuti model himpunan dimana setiap anggota-nya harus unik. Urutan maupun letak dari anggotanya tidaklah penting, hanya keberadaan anggotanya saja yang penting.

Contoh dalam Program:

Listing Program

```
1  import java.util.*;
2  public class setCollection {
3  public static void main(String[] args) {
4  Set<String> set1 = new HashSet<String>();
```

```
5  Colqlections.addAll(set1, "A B C D E ".split(" "));
6  set1.add("M");
7   System.out.println("B:  " +  set1.contains("B"));//ada
H
8  System.out.println("H: " + set1.contains("H"));//ada N
9  Set<String> set2 = new HashSet<String>();
10 Collections.addAll(set2, "E F G H".split(" "));
11 //Apakah huruf2 di set2 ada di set 1 ?? -> true
12
System.out.println("set2inset1:"+set1.containsAll(set2));
13 set1.remove("H");
14 System.out.println("set1: " + set1);
15      System.out.println("set2      in      set1:"      +
set1.containsAll(set2));
16 set1.removeAll(set2);
17 System.out.println("set2 dihapus dari set1:" + set1);
18 Collections.addAll(set1, "X Y Z".split(" "));
19 System.out.println("'X Y Z' ditambahkan ke set1: " +
set1);
20 }
21 }
```

4) List

List merupakan pengelompokan berdasarkan urutan seperti layaknya array, karena itu ia memiliki posisi awal dan juga posisi akhir. Dengan list kita bisa menyimpan suatu objek pada awal atau akhir list, menyisipkan, mengakses serta menghapus isi list pada posisi index tertentu dimana semua proses ini selalu didasarkan pada urutannya. Selain itu isi list tidak harus unik. Beberapa Class java yang mengimplementasi kelas ini antara lain : *Vector, Stack, LinkedList,* dan *ArrayList.*

Contoh dalam Program :

✓ **Vector**

Listing Program

```
1  import java.util.Scanner;
2  import java.util.Vector;
3  public class VektorJava{
4  public static void main(String[] args) {
5  String data = "";
6  Vector<String> V = new Vector<String>();
7  Scanner input = new Scanner(System.in);
8  for(int i=0;i<=3;i++){
9  System.out.print("Masukan Data "+i+": ");
10 data = input.next();
11 V.add(data);
12 }
13 System.out.println(V);
14 String nama = (String) V.get(0);
15 System.out.println(nama);
16 }
17 }
```

✓ **Stack**

Listing Program

```
1  import java.io.*;
2  import java.util.*;
3  public class StackImplement {
4  Stack<Integer> stack = new Stack<Integer>();
5  String str;
6  int num, n;
7  public static void main(String[] args) {
8  StackImplement q = new StackImplement();
9  }
10 StackImplement() {
11 try {
12 BufferedReader bf = new BufferedReader(
```

```
13 new InputStreamReader(System.in));
14 System.out.print("Banyak Data : ");
15 str = bf.readLine();
16 num = Integer.parseInt(str);
17 for (int i = 1; i <= num; i++) {
18 System.out.print("Masukan Elemen " + i + " : ");
19 str = bf.readLine();
20 n = Integer.parseInt(str);
21 stack.push(n);
22 }
23 } catch (IOException e) {
24 }
25 System.out.println("Stack : ");
26 while (!stack.empty()) {
27 System.out.print(stack.pop() + "  ");
28 }
29 System.out.println();
30 }
31 }
```

- ✓ **LinkedList**

Listing Program

```
1  import java.util.LinkedList;
2  public class LinkListCollection {
3  public static void main(String[] args) {
4  LinkedList<String> A = new LinkedList<String>();
5  String[] nama = {"David","Alfa","Benny"};
6  //Tambah data data diambil dari array nama;
7  for(int nList = 0;nList<nama.length;nList++){
8  A.add(nama[nList]);
9  }
10 //Tampil Data
11  System.out.println("Data Asli : ");
12  for(int nList = 0;nList<nama.length;nList++){
```

```
13            System.out.println("Indeks      "+nList+":
"+A.get(nList));
14   }
15    System.out.println("\nTambah  data  di  Index  ke-3  :
");
16    A.add(3, "Danni");
17    for(int nList = 0;nList<=nama.length;nList++){
18            System.out.println("Indeks      "+nList+":
"+A.get(nList));
19  }
20 System.out.println("\nDelete data di Index ke-2 : ");
21   A.remove(2);
22   for(int nList = 0;nList<nama.length;nList++){
23          System.out.println("Indeks          "+nList+":
"+A.get(nList));
24   }
25   System.out.println("\nTambah data di Awal list : ");
26   A.addFirst("Marcelo");
27   for(int nList = 0;nList<=nama.length;nList++){
28          System.out.println("Indeks          "+nList+":
"+A.get(nList));
29   }
30    System.out.println("\nTambah  data  di  Akhir  list  :
");
31   A.addLast("Ferdi");
32   for(int nList = 0;nList<=nama.length+1;nList++){
33          System.out.println("Indeks          "+nList+":
"+A.get(nList));
34   }
35 System.out.println("\nTambah data di Akhir list : ");
36   A.remove(4);
37   for(int nList = 0;nList<=nama.length;nList++){
38            System.out.println("Indeks      "+nList+":
"+A.get(nList));
```

```
39  }
40  }
41  }
```

✓ **ArrayList**

Listing Program

```
1   import java.util.ArrayList;
2   import java.util.Scanner;
3   public class ArrayListCollection {
4    public static void main(String[] args) {
5    String str = "";
6   ArrayList<String> senerai = new ArrayList<String>();
7    Scanner scan = new Scanner(System.in);
8    while (true) {
9    try {
10   System.out.println("\n1. Tambah Data");
11   System.out.println("2. Lihat Data");
12   System.out.println("3. Delete Data");
13   System.out.println("4. Exit");
14   System.out.print("Masukan pilihan : ");
15   int pilihan = scan.nextInt();
16   switch (pilihan) {
17   case 1:
18   int jmlDt = 0;
19   System.out.print("Tambah data : ");
20   str = scan.next();
21   senerai.add(str);
22   jmlDt++;
23   break;
24   case 2:
25   System.out.println("\nIsi senerai sekarang : ");
26        System.out.println("Jumlah  Element  :  "  +
senerai.size());
27   for (int i = 0; i < senerai.size(); i++) {
```

```
28   System.out.println("Index   "   +   i   +   ".   "   +
senerai.get(i));
29    }
30   System.out.println("");
31   break;
32   case 3:
33   int indeks;
34   try {
35   System.out.print("Masukan  Index  list  yang  akan  di
hapus: ");
36   indeks = scan.nextInt();
37   senerai.remove(indeks);
38   System.out.println(">remove data sukses<");
39   } catch (IndexOutOfBoundsException a) {
40   System.out.println("Indeks Array melebihi batas");
41   }
42   break;
43   case 4:
44   System.out.println("Terima kasih");
45   System.exit(0);
46   default:
47   System.out.println("Inputan Tidak tersedia \n");
48 }
49  } catch (Exception e) {
50   System.err.println("Salah Input");
51   System.exit(1);
52   }
53   }
54   }
55   }
```

Implementasi dari ArrayList dan Vector ini mirip seperti array, dimana kita dapat menggakses anggotanya melalui indeksnya. Kelebihannya adalah ia dapat menyesuaikan ukurannya sesuai dengan kebutuhan. Perbedaan dari kedua

kelas ini, vector secara internal telah mendukung sinkronisasi, sedangkan ArrayList secara internal tidak mendukung sinkronisasi.

5) Queue

Queue merupakan model pengelompokan berdasarkan metode antrian suatu prioritas tertentu(contoh FIFO-First In First Out). Beberapa Class java yang mengimplementasi interface *Queue* ini antara lain *PriorityQueue* dan *LinkedList.*Dalam queue terdapat fungsi enqueue yang digunakan untuk mengatur inputan data yang akan masuk antrian, pada fungsi ini juga diperiksa apakah antrian sudah penuh atau belum, jika sudah penuh maka tidak dapat diisi lagi. Fungsi dequeue digunakan untuk mengatur data agarkeluar dari antrian secara tertib, karena queue menggunakan prinsip FIFO.

Contoh :

Listing Program

```
1   import java.io.*;
2   import java.util.*;
3   public class QueueJava{
4   String str;
5   Int num;
6  public static void main(String[] args){
7   QueueJava q = new QueueJava();
8   }
9   public QueueJava(){
10 try{
11 LinkedList<Integer> list = new LinkedList<Integer>();
12  BufferedReader bf = new BufferedReader(
13  new InputStreamReader(System.in));
14  System.out.print("Banyak Data : ");
15  str = bf.readLine();
16  if((num = Integer.parseInt(str)) == 0){
17  System.out.println("Anda menekan angka nol.");
18  System.exit(0);
19  }
20  else{
```

```
21  for(int i = 0; i < num; i++){
22  System.out.print("Masukan Elemen "+i+" : ");
23  str = bf.readLine();
24  int n = Integer.parseInt(str);
25  list.add(n);
26  }
27  }
28  System.out.println("\nElement Pertama   : "
29  + list.removeFirst());
30  System.out.println("Element Terakhir  : "
31  + list.removeLast());
32  System.out.println("Element Tengah    : ");
33  while(!list.isEmpty()){
34  System.out.print(list.remove() + " ");
35  }
36  System.out.println("");
37  }
38  catch(Exception e){
39  System.out.println(e.getMessage()
40  + " adalah String.");
41  System.exit(0);
42  }
43  }
44  }
```

C. Rangkuman

Array merupakan objek yang dapat digunakan untuk menyimpan sejumlah data. Data yang ditampung di array dapat berupa tipe data ataupun kelas(objek). Ciri dari array adalah terdapat simbol [ ].Array ada dua tipe yaitu Array 1 dimention dan Array 2 dimention.

Collection merupakan istilah yang dipakai untuk setiap objek yang berfungsi untuk mengelompokkan beberapa objek tertentu menggunakan teknik tertentu.Semua class yang berhubungan dengan pengelompokkan objek ini tergabung dalam Java Collection Framework.

Set merupakan pengelompokkan mengikuti model himpunan dimana setiap anggotanya harus unik.

List merupakan pengelompokkan berdasarkan urutan seperti array. List dapat menyimpan suatu objek pada awal atau akhir list, menyisipkan, mengakses serta menghapus isi list pada posisi index tertentu dimana semua proses selalu didasarkan pada urutannya. Isi list tidak harus unik.

Queue merupakan model pengelompokkan berdasar metode antrian suatu prioritas tertentu. Contohnya FIFO (first input first output).Dalam queue terdapat fungsi enqueue, berfungsi mengatur inputan pada queue. Dan juga terdapat fungsi dequeue yang mengatur output dari queue menjadi tertib.

D. Tugas

Tugas 1

Buatlah program yang membaca skor siswa, yang mencari skor terbaik, dan menugaskan nilai berdasarkan skema berikut ini :

1. Nilai A jika skor >= Terbaik – 10;
2. Nilai B jika skor >= Terbaik – 20;
3. Nilai C jika skor >= Terbaik – 30;
4. Nilai D jika skor >= Terbaik – 40;
5. Nilai E jika skor >= Terbaik – 50;

Program meminta user meng-inputkan jumlah total siswa, semua skor, dan kemudian menyimpulkan nilai berdasar skema diatas.

❖ Mengamati Listing Program dan Output Program

7. Tentukan nama class yang akan digunakan.
8. Tentukan variabel yang akan digunakan.
9. Tentukan method yang akan digunakan.
10. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

11. Buatlah listing programnya.
12. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

❖ Bandingkan dan Simpulkan
Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Sebutkan definisi dari istilah berikut ini :
   a. Queue
   b. Enqueue
   c. Dequeue
5. Apakah dapat baris-baris didalam suatu array dua dimensi memiliki panjang yang berbeda?
6. Jelaskan apa yang dimaksud dengan collection ! Disebut dengan apa objek yang ada dalam collection?
7. Apa pengertian dari linked list? Sebutkan macam-macam dari linked list !

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Definisi dari istilah :

a)Queue.....
.....
.....
.....
.....

b) Enqueue
.....
.....
.....
.....

c)Dequeue
.....
.....
.....
.....

LJ- 02 : Dapatkah baris dalam array dua dimensi memiliki panjang berbeda :

.....
.....
.....
.....
.....

LJ- 03 : Yang dimaksud dengan collection :

.....
.....
.....

.....

.....

.....

LJ- 04: Definisi dari linked list :

.....

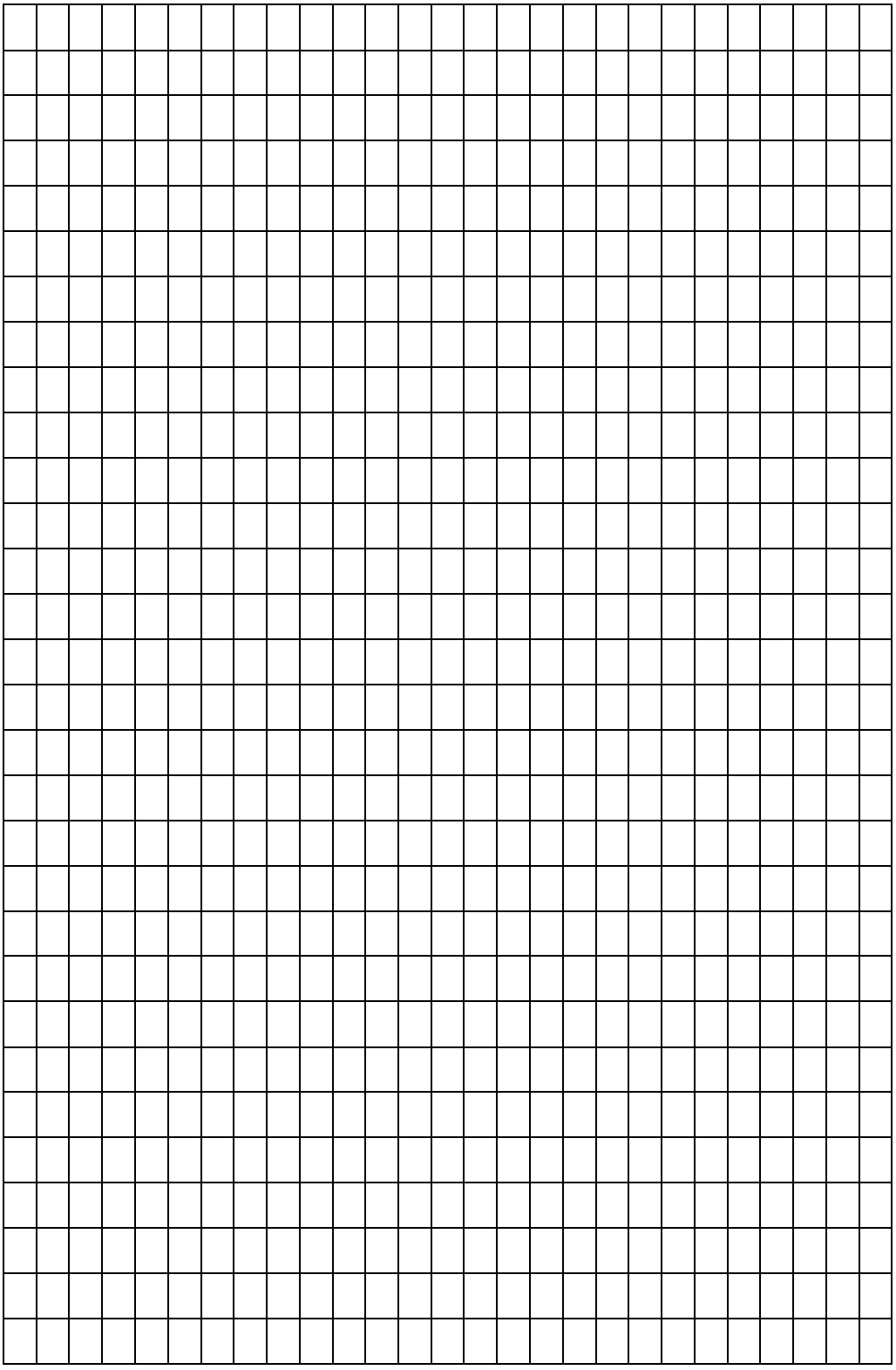
.....

.....

.....

.....

G. Lembar Kerja Siswa



6. Kegiatan 6 : Sistem File

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 6 ini siswa diharapkan dapat :

1) Menerapkan operasi file dan Input Output (IO)
2) Menyajikan operasi file dan operasi Input

B. Uraian Materi

1) Sistem File

Class dasar I/O Reader, Writer, InputStream dan OutputStream hanya menyediakan operasi I/O sangat dasar. Misalnya, class InputStream memiliki metode instansi**publicint** read() **throws** IOExceptionuntuk membaca satu byte data dari aliran input. Jika sampai pada akhir dari aliran input , metode read() akan mengembalikan nilai -1. Jika ada kesalahan yang terjadi pada saat pengambilan input, maka pengecualian IOException akan dilemparkan. Karena IOException adalah class pengecualian yang harus ditangani, artinya kita harus menggunakan metode read() di dalam penyataan **try** atau mengeset subrutin untuk **throws** IOException.

Class InputStream juga memiliki metode untuk membaca beberapa byte data dalam satu langkah ke dalam array **byte**. Akan tetapi InputStream tidak memiliki metode untuk membaca jenis data lain, seperti **int** atau **double** dari aliran. Ini bukan masalah karena dalam prakteknya kita tidak akan menggunakan objek bertipe InputStream secara langsung. Yang akan kita gunakan adalah class turunan dari InputStream yang memiliki beberapa metode input yang lebih beragam daripada InputStream itu sendiri.

Begitu juga dengan class OutputStream memiliki metode output primitif untuk menulis satu byte data ke aliran output, yaitu metode**publicvoid** write(**int** b) **throws** IOException, tapi kita hampir pasti akan menggunakan class turunannya yang mampu menangani operasi yang lebih kompleks.Class Reader dan Writer memiliki operasi dasar yang hampir sama, yaitu read dan write, akan tetapi class ini berorientasi karakter (karena digunakan untuk membaca dan menulis data

yang bisa dibaca manusia). Artinya operasi baca tulis akan mengambil dan menulis nilai **char** bukan byte. Dalam prakteknya kita akan menggunakan class turunan dari class-class dasar ini.

Salah satu hal menarik dari paket I/O pada Java adalah kemungkinan untuk menambah kompleksitas suatu aliran dengan membungkus aliran tersebut dalam objek aliran lain. Objek pembungkus ini juga berupa aliran, sehingga kita juga bisa melakukan baca tulis dari objek yang sama dengan tambahan kemampuan dalam objek pembungkusnya.Misalnya, PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusial. Jika kita memiliki objek bertipe Writer atau turunannya, dan kita ingin menggunakan metode pada PrintWriter untuk menulis data, maka kita bisa membungkus objek Writer dalam objek PrintWriter.

Contoh jika baskomKarakter bertipe Writer, maka kita bisa membuatPrintWriter printableBaskomKarakter = **new** PrintWriter(baskomKarakter);Ketika kita menulis data ke printableBaskomKarakter dengan menggunakan metode pada PrintWriter yang lebih canggih, maka data tersebut akan ditempatkan di tempat yang sama dengan apabila kita menulis langsung pada baskomKarakter. Artinya kita hanya perlu membuat antar muka yang lebih baik untuk aliran output yang sama. Atau dengan kata lain misalnya kita bisa menggunakan PrintWriter untuk menulis file atau mengirim data pada jaringan.

Untuk lengkapnya, metode pada class PrintWriter memiliki metode sebagai berikut :

// Metode untuk menulis data dalam

// bentuk yang bisa dibaca manusia

publicvoid print(String s)

publicvoid print(**char** c)

publicvoid print(**int** i)

publicvoid print(**long** l)

publicvoid print(**float** f)

publicvoid print(**double** d)

publicvoid print(**boolean** b)

// Menulis baris baru ke aliran
publicvoid println()
// Metode ini sama dengan di atas
// akan tetapi keluarannya selalu
// ditambah dengan baris baru
publicvoid println(String s)
publicvoid println(**char** c)
publicvoid println(**int** i)
publicvoid println(**long** l)
publicvoid println(**float** f)
publicvoid println(**double** d
publicvoid println(**boolean** b)

Catatan bahwa metode-metode di atas tidak pernah melempar pengecualian IOException. Akan tetapi, class PrintWriter memiliki metode**publicboolean** checkError()yang akan mengembalikan true jika ada kesalahan yang terjadi ketika menulis ke dalam aliran. Class PrintWriter menangkap pengecualian IOException secara internal, dan mengeset nilai tertentu di dalam class ini jika kesalahan telah terjadi. Sehingga kita bisa menggunakan metode pada PrintWriter tanpa khawatir harus menangkap pengecualian yang mungkin terjadi. Akan tetapi, jika kita ingin membuat progam yang tangguh tentunya kita harus selalu memanggil checkError() untuk melihat apakah kesalahan telah terjadi ketika kita menggunakan salah satu metode pada PrintWriter.

Ketika kita menggunakan metode PrintWriter untuk menulis data ke aliran, data tersebut diubah menjadi rangkaian karakter yang bisa dibaca oleh manusia. Bagaimana caranya jika kita ingin membuat data dalam bentuk bahasa mesin?
Paket java.Io memiliki class aliran byte, yaitu DataOutputStream yang bisa digunakan untuk menulis suatu data ke dalam aliran dalam format biner. DataOutputStream berhubungan erat dengan OutputStream seperti hubungan antara PrintWriter dan Writer.

Artinya, OutputStream hanya berisi metode dasar untuk menulis byte, sedangkan DataOutputStream memiliki metode writeDouble(**double** x) untuk menulis nilai

double, writeInt(**int** x) untuk menulis nilai int, dan seterusnya. Dan juga kita bisa membungkus objek bertipe OutputStream atau turunannya ke dalam aliran DataOutputStream sehingga kita bisa menggunakan metode yang lebih kompleks.
Misalnya, jika baskomByte adalah variabel bertipe OutputStream, makaDataOutputStream baskomData = **new** DataOutputStream(baskomByte);untuk membungkus baskomByte dalam baskomData.Untuk mengambil data dari aliran, java.io memiliki class DataInputStream. Kita bisa membungkus objek bertipe InputStream atau turunannya ke dalam objek bertipe DataInputStream. Metode di dalam DataInputStream untuk membaca data biner bisa menggunakan readDouble(), readInt() dan seterusnya. Data yang ditulis oleh DataOutputStream dijamin untuk bisa dibaca kembali oleh DataInputStream, meskipun data kita tulis pada satu komputer dan data dibaca pada komputer jenis lain dengan sistem operasi berbeda. Kompatibilitas data biner pada Java adalah salah satu keunggulan Java untuk bisa dijalakan pada beragam platform.

Salah satu fakta yang menyedihkan tentang Java adalah ternyata Java tidak memiliki class untuk membaca data dalam bentuk yang bisa dibaca oleh manusia. Dalam hal ini Java tidak memiliki class kebalikan dari PrintWriter sebagaimana DataOutputStream dan DataInputStream. Akan tetapi kita tetap bisa membuat class ini sendiri dan menggunakannya dengan cara yang persis sama dengan class-class di atas.

Class PrintWriter, DataInputStream, dan DataOutputStream memungkinkan kita untuk melakukan input dan output semua tipe data primitif pada Java. Pertanyaannya bagaimana kita melakukan baca tulis suatu objek?Mungkin secara tradisional kita akan membuat fungsi sendiri untuk memformat objek kita menjadi bentuk tertentu, misalnya urutan tipe primitif dalam bentuk biner atau karakter kemudian disimpan dalam file atau dikirim melalui jaringan. Proses ini disebut serialisasi (serializing) objek.
Pada inputnya, kita harus bisa membaca data yang diserialisasi ini sesuai dengan format yang digunakan pada saat objek ini diserialisasi. Untuk objek kecil, pekerjaan semacam ini mungkin bukan masalah besar. Akan tetapi untuk

ukuran objek yang besar, hal ini tidak mudah.Akan tetapi Java memiliki cara untuk melakukan input dan output isi objek secaraotomatis,yaitu dengan menggunakan ObjectInputStream danObjectOutputStream.Class-class ini adalah class turunan dari InputStream danOutputStream yang bisa digunakan untuk membaca dan menulis objek yang sudah diserialisasi.ObjectInputStream dan ObjectOutputStream adalah class yang bisa dibungkus oleh class InputStream dan OutputStream lain. Artinya kita bisa melakukan input dan output objek pada aliran byte apa saja.Metde untuk objek I/O adalah readObject() yang tersedia pada ObjectInputStream dan writeObject(Object obj) yang tersedia dalam ObjectOutputStream. Keduanya bisa melemparkan IOException. Ingat bahwa readObject() mengembalikan nilai bertipe Object yang artinya harus di-type cast ke tipe sesungguhnya.ObjectInputStream dan ObjectOutputStream hanya bekerja untuk objek yang mengimplementasikan interface yang bernama Serializable. Lebih jauh semua variabel instansi pada objek harus bisa diserialisasi, karena interface Serializable tidak mempunyai metode apa-apa. Interface ini ada hanya sebagai penanda untuk kompiler supaya kompiler tahu bahwa objek ini digunakan untuk baca tulis ke suatu media.

Yang perlu kita lakukan adalah menambahkan "**implementsSerializable**" pada definisi class. Banyak class standar Java yang telah dideklarasikan untuk bisa diserialisasi, termasuk semua komponen class Swing dan AWT. Artinya komponen GUI pun bisa disimpan dan dibaca dari dalam perangkat I/O menggunakan ObjectInputStream dan ObjectOutputStream.

2) Stream, Reader, dan Writer

Tanpa bisa berinteraksi dengan dunia lain, suatu program tidak ada gunanya. Interaksi suatu program dengan dunia lain sering disebut input/output atau I/O. Sejak dulu, salah satu tantangan terbesar untuk mendesain bahasa pemrograman baru adalah mempersiapkan fasilitas untuk melakukan input dan output. Komputer bisa terhubung dengan beragam jenis input dan output dari berbagai perangkat. Jika bahasa pemrograman harus dibuat secara khusus untuk setiap jenis perangkat, maka kompleksitasnya akan tak lagi bisa ditangani. Salah satu kemajuan terbesar dalam sejarah pemrograman adalah adanya konsep (atau abstraksi) untuk memodelkan perangkat I/O. Dalam Java, abstraksi

ini disebut dengan aliran (stream). Bagian ini akan memperkenalkan tentang aliran, akan tetapi tidak menjelaskan dengan komplit. Untuk lebih lengkapnya, silakan lihat dokumen resmi Java.

Ketika berhubungan dengan input/output, kita harus ingat bahwa ada dua kategori data secara umum : data yang dibuat oleh mesin, dan data yang bisa dibaca manusia. Data yang dibuat mesin ditulis dengan model yang sama dengan bagaimana data tersebut disimpan di dalam komputer, yaitu rangkaian nol dan satu. Data yang bisa dibaca manusia adalah data dalam bentuk rangkaian huruf. Ketika kita membaca suatu bilangan 3.13159, kita membacanya sebagai rangkaian huruf yang kita terjemahkan sebagai angka. Angka ini akan ditulis dalam komputer sebagai rangkaian bit yang kita tidak mengerti.

Untuk menghadapi kedua jenis data ini, Java memiliki dua kategori besar untuk aliran : aliran byte untuk data mesin (byte stream), dan aliran karakter (character stream) untuk data yang bisa dibaca manusia. Ada banyak kelas yang diturunkan dari kedua kategori ini.Setiap objek yang mengeluarkandata ke aliran byte masuk sebagai kelas turunan dari kelas abstrak OutputStream. Objek yang membaca data dari aliran byte diturunkan dari kelas abstrak InputStream. Jika kita menulis angka ke suatu OutputStream, kita tidak akan bisa membaca data tersebut karena ditulis dalam bahasa mesin. Akan tetapi data tersebut bisa dibaca kembali oleh InputStream. Proses baca tulis data akan menjadi sangat efisien, karena tidak ada penerjemahan yang harus dilakukan : bit yang digunakan untuk menyimpan data di dalam memori komputer hanya dikopi dari dan ke aliran tersebut.

Untuk membaca dan menulis data karakter yang bisa dimengerti manusia, kelas utamanya adalah Reader dan Writer. Semua kelas aliran karakter merupakan kelas turunan dari salah satu dari kelas abstrak ini. Jika suatu angka akan ditulis dalam aliran Writer, komputer harus bisa menerjemahkannya ke dalam rangkaian karakter yang bisa dibaca maunsia.

Membaca angka dari aliran Reader menjadi variabel numerik juga harus diterjemahkan, dari deretan karakter menjadi rangkaian bit yang dimengerti

komputer. (Meskipun untuk data yang terdiri dari karakter, seperti dari editor teks, masih akan ada beberapa terjemahan yang dilakukan. Karakter disimpan dalam komputer dalam nilai Unicode 16-bit. Bagi orang yang menggunakan alfabet biasa, data karakter biasanya disimpan dalam file dalam kode ASCII, yang hanya menggunakan 8-bit. Kelas Reader dan Writer akan menangani perubahan dari 16-bit ke 8-bit dan sebaliknya, dan juga menangani alfabet lain yang digunakan negara lain.)

Adalah hal yang mudah untuk menentukan apakah kita harus menggunakan aliran byte atau aliran karakter. Jika kita ingin data yang kita baca/tulis untuk bisa dibaca manusia, maka kita gunakan aliran karakter. Jika tidak, gunakan aliran byte. System.in dan System.out sebenarnya adalah aliran byte dan bukan aliran karakter, karenanya bisa menangani input selain alfabet, misalnya tombol enter, tanda panah, escape, dsb.

Kelas aliran standar yang akan dibahas berikutnya didefinisikan dalam paket java.io beserta beberapa kelas bantu lainnya. Kita harus mengimpor kelas-kelas tersebut dari paket ini jika kita ingin menggunakannya dalam program kita. Artinya dengan menggunakan "import java.io.*" di awal kode sumber kita.
Aliran tidak digunakan dalam GUI, karena GUI memiliki aliran I/O tersendiri. Akan tetapi kelas-kelas ini digunakan juga untuk file atau komunikasi dalam jaringan. Atau bisa juga digunakan untuk komunikasi antar thread yang sedang bekerja secara bersamaan. Dan juga ada kelas aliran yang digunakan untuk membaca dan menulis data dari dan ke memori komputer.

C. Rangkuman

Class dasar I/O (input output) Reader, Writer, InputStream dan OutputStream hanya menyediakan operasi I/O sangat dasar. Misalnya, class InputStream memiliki metode instansi publicint read() throws IOException untuk membaca satu byte data dari aliran input. Jika sampai pada akhir dari aliran input , metode read() akan mengembalikan nilai -1. Jika ada kesalahan yang terjadi pada saat pengambilan input, maka pengecualian IOException akan dilemparkan.

Class InputStream memiliki metode untuk membaca byte data dalam satu langkah ke dalam array byte. Tetapi InputStream tidak memiliki metode untuk membaca jenis data lain, seperti int atau double dari aliran.

Class OutputStream memiliki metode output primitif untuk menulis satu byte data ke aliran output, yaitu metode publicvoid write(int b) throws IOException, tapi kita hampir pasti akan menggunakan class turunannya yang mampu menangani operasi yang lebih kompleks.

Class Reader dan Writer memiliki operasi dasar yang samayaitu read dan write, akan tetapi class ini berorientasi karakter. Artinya operasi baca tulis akan mengambil dan menulis nilai char bukan byte.

PrintWriter adalah class turunan dari Writer yang memiliki metode tambahan untuk menulis tipe data Java dalam karakter yang bisa dibaca manusia.Jika kita memiliki objek bertipe Writer atau turunannya, dan kita ingin menggunakan metode pada PrintWriter untuk menulis data, maka kita bisa membungkus objek Writer dalam objek PrintWriter.

Class PrintWriter menangkap pengecualian IOException secara internal, dan mengeset nilai tertentu di dalam class ini jika kesalahan telah terjadi.Ketika kita menggunakan metode PrintWriter untuk menulis data ke aliran, data tersebut diubah menjadi rangkaian karakter yang bisa dibaca oleh manusia.

Untuk membaca dan menulis data karakter yang dapat dimengerti manusia, kelas utamanya adalah Reader dan Writer.Semua kelas aliran karakter merupakan kelas turunan dari salah satu dari kelas abstrak ini. Jika suatu angka akan ditulis dalam aliran Writer, komputer harus bisa menerjemahkannya ke dalam rangkaian karakter yang bisa dibaca maunsia.

D. Tugas

Tugas 1

Buatlah program untuk menampilkan data seperti dibawah ini :

```
Rismon H Sianipar 90
Rebecca V Siahaan 85
```

Buat program menggunakan `java.util.Scanner`, dengan tipe data String dan int. dalam program tersebut, user harus memasukkan datanya, lalu program menampilkan hasil dari inputan user.

❖ Mengamati Listing Program dan Output Program

1. Tentukan nama class yang akan digunakan.
2. Tentukan variabel yang akan digunakan.
3. Tentukan method yang akan digunakan.
4. Buatlah class diagram yang menggambarkan bagian-bagian dari class yang telah ditentukan. Contoh class diagram :

Nama class
Method
Operasi

5. Buatlah listing programnya.
6. Lakukan kompilasi dan debug pada program.

No	Output Program
1.	
2.	
3.	
4.	

5.	
6.	
7.	
8.	
9.	
10.	

❖ **Bandingkan dan Simpulkan**

Bandingkan hasil program yang telah kamu buat dengan hasil program teman sebangku atau kelompok lain

Dari hasil perbandingan tersebut, hal penting apa yang harus dirumuskan secara bersama

E. Tes Formatif

Dalam test ini setiap anda harus membaca dengan cermat dan teliti setiap butir soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test formatif yang telah disediakan.

1. Apa yang dimaksud dengan kelas PrintWriter ?
2. Jelaskan pengertian metode public boolean pada class PrintWriter !
3. Apa yang kamu ketahu tentang kelas dasar IO Reader, Writer, InputStream, dan OutputStream ?
4. Jelaskan metode-metode pembacaan jenis data pada Class InputStream !
5. Apa pengertian dari ObjectInputStream dan ObjectOutputStream ?

F. Lembar Jawaban Test Formatif (LJ).

LJ- 01 : Pengertian dari kelas PrintWriter :

.....
.....
.....
.....
.....
.....

LJ- 02 : Pengertian metode public Boolean pada class PrintWriter :

.....
.....
.....
.....
.....
.....
.....
.....

LJ- 03 : kelas dasar IO Reader, Writer, InputStream, dan OutputStream :

.....
.....
.....
.....
.....
.....
.....

LJ- 04: Metode-metode pembacaan jenis data pada Class InputStream :

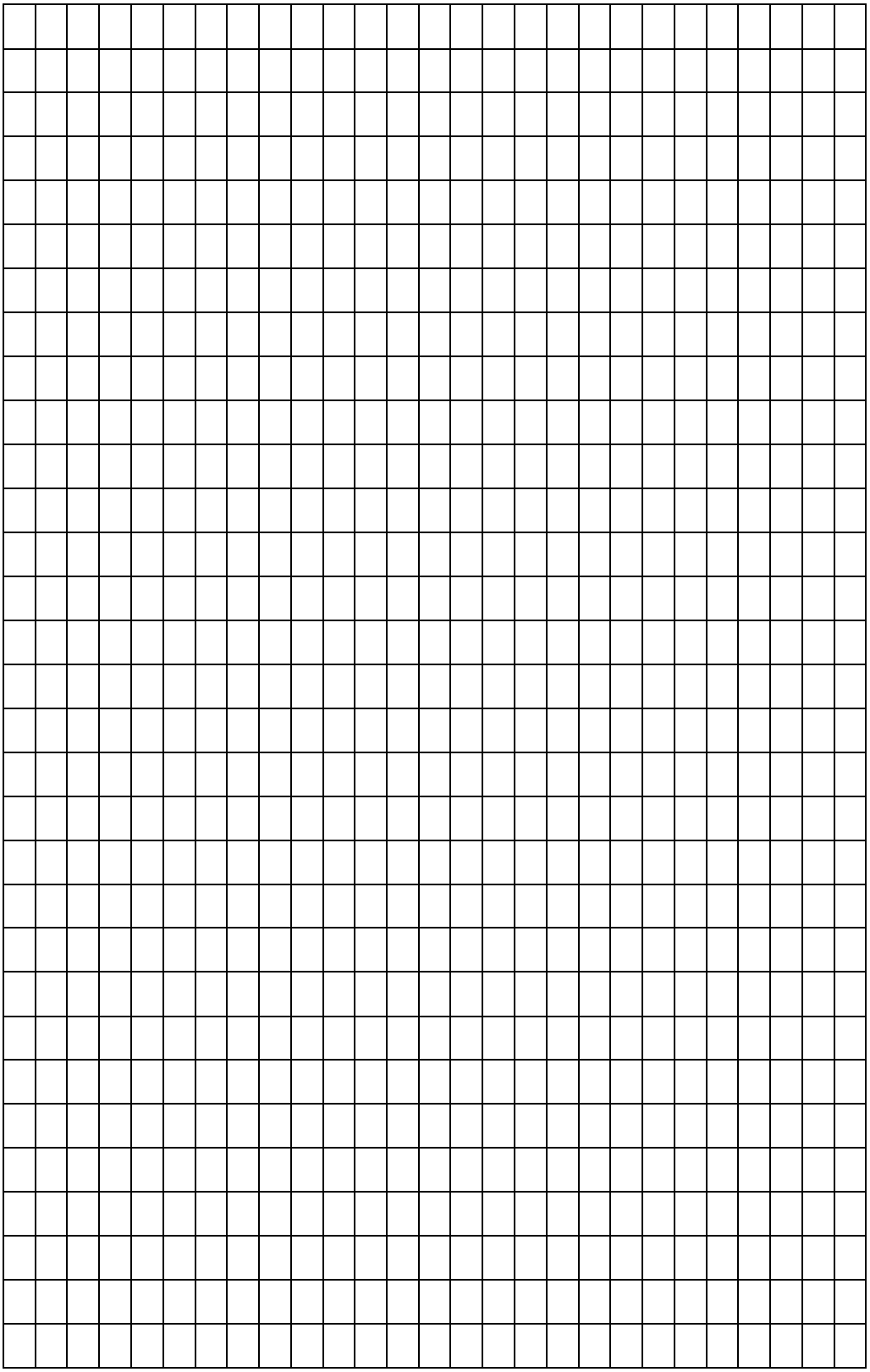
Pengertian ObjectInputStream dan ObjectOutputStream :

.....
.....
.....
.....
.....

LJ- 05: Pengertian ObjectInputStream dan ObjectOutputStream :

.....
.....
.....
.....
.....

G. Lembar Kerja Siswa



DAFTAR PUSTAKA

Christian Musnter, Java 2 JDK 5 –Grundlagen , Herdt – Verlag forbildungsmedien Gagh, Bodenheilm, 2006

C. Thomas wu, An Introduction to Object- Oriented Programming with Java, McGraw Hill 2001

Deitel, Java : How to program, Prentice Hall, New jersey, 2002

Joyce Avestro , Introduction Programming 1, Java Education Development Initiatif, 2003

Joyce Avestro , Introduction Programming 2, Java Education Development Initiatif, 2003

Patric Noughton , The Java Handbook, McGrawHill, Inc, 2006

R.H.Sianipar , Teori dan Implementasi Java, Informatika Bandung,2013