Modul Praktikum Pemrograman Berbasis Objek

Kartu Kontrol

Nama Mahasiswa : ……..

NIM : ……..

Pertemuan	Hari/ Tanggal	Paraf Dosen / Asisten			
		Tugas Pendahuluan	Respon	Kehadiran	Laporan Praktikum
1	/				
2	/				
3	/				
4	/				
5	/				
6	/				
7	/				
8	/				
9	/				
10	/				

Daftar Isi

Praktikum Pemrograman Berorientasi Objek

Nama Dosen: Adhy Rizaldy, S.Kom., M.Kom.

Pendahuluan

Praktikum ini membahas tentang bagaimana membuat pengkodean berbasis objek pada bahasa Java sebagai salah satu platform OOP terbaik. Praktikan akan mempelajari mulai dari cara Input Output, menggunakan operator, operasi method dan class, hingga Generic di Java. Diharapkan dari setiap modul dalam buku ini Syarat – Syarat Kompetensi pada RPS Mata Kuliah Pemrograman Berorientasi Objek dapat tercapai.

Deskripsi Praktikum

Pertemuan 1: Class dan Object
Pertemuan 2: Encapsulation
Pertemuan 3: Percabangan
Pertemuan 4: Perulangan
Pertemuan 5: Inheritance
Pertemuan 6: Polymorphism
Pertemuan 7: Abstraction
Pertemuan 8: Java Collections
Pertemuan 9: Generic
Pertemuan 10: Exception handling

Modul Praktikum 1: Class dan Object

Deskripsi Pertemuan

Praktikum ini membahas tentang pendefinisian Class dan objek. Bagaimana membangun aplikasi Java anda yang pertama. Bagaimana mengembangkan setiap kodingan dimulai dari sebuah Class/objek, dan method-method lainnya.

Syarat Kompetensi

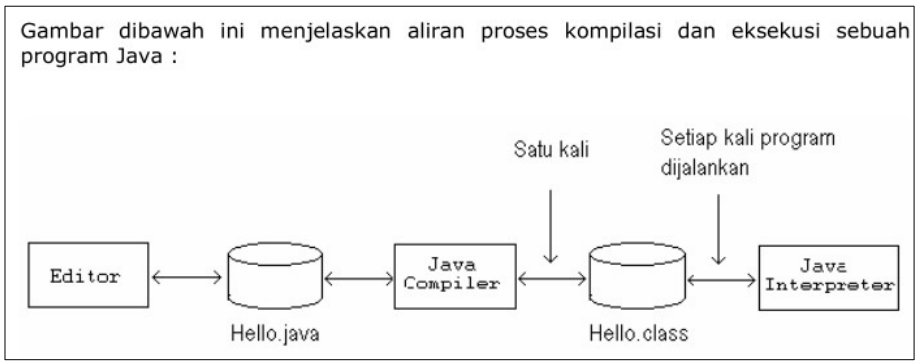
a. Membuat aplikasi Hello World dengan Android Studio.
b. Menjalankan aplikasi di emulator atau perangkat.
c. Mengimplementasikan TextView dalam layout untuk aplikasi.
d. Membuat dan menggunakan sumber daya string.
e. Memahami alur kerja dasar Android Studio.

Standar Kompetensi

1. Mahasiswa dapat melakukan kompilasi berkas ekstensi Java
2. Mengidentifikasi bagian dasar dari program java
3. Menggunakan tipe data dasar, tipe variabel, keyword dasar dan operator pada Java.
4. Mengembangkan

Dasar Teori

Fase pemrograman Java



Kode program yang dibuat kemudian tersimpan dalam sebuah berkas berekstensi .java. Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan Java Compiler. Hasil dari kompilasi berupa berkas bytecode dengan ekstensi .class. Berkas yang mengandung bytecode tersebut kemudian akan dikonversikan oleh Java Interpreter menjadi bahasa mesin sesuai dengan jenis dan platform yang digunakan.:

Proses	*Tool*	*Hasil*
Menulis kode program	*Text editor*	Berkas berekstensi *.java*
Kompilasi program	Java Compiler	Berkas berekstensi *.class* (Java Bytecodes)
Menjalankan program	Java Interpreter	Program Output

Tabel 1: Ringkasan Fase dari sebuah Program Java

Operator

Operator Aritmatika

Berikut ini adalah dasar operator aritmatika yang dapat digunakan untuk membuat suatu program Java,

Operator	*Penggunaan*	*Keterangan*
+	op1 + op2	Menambahkan op1 dengan op2
*	op1 * op2	Mengalikan op1 dengan op2
/	op1 / op2	Membagi op1 dengan op2
%	op1 % op2	Menghitung sisa dari pembagian op1 dengan op2
-	op1 - op2	Mengurangkan op2 dari op1

Class

Elemen – elemen sebuah Class terdiri dari Constructor, Instance Variable, Class Variables, Methods, Modifier, Parameters.

Instansiasi Objek

Methods

Setiap Class memiliki method – method. Adapun jenis method terdiri dari :

- **main** method
- **accessor** method
- **mutator** method

Input Output

Untuk menerima Input keyboard dari user, dibutuhkan import library Scanner. Untuk mencetak output dapat menggunakan *System.out* dan method-method di dalamnya,

Tugas Pendahuluan

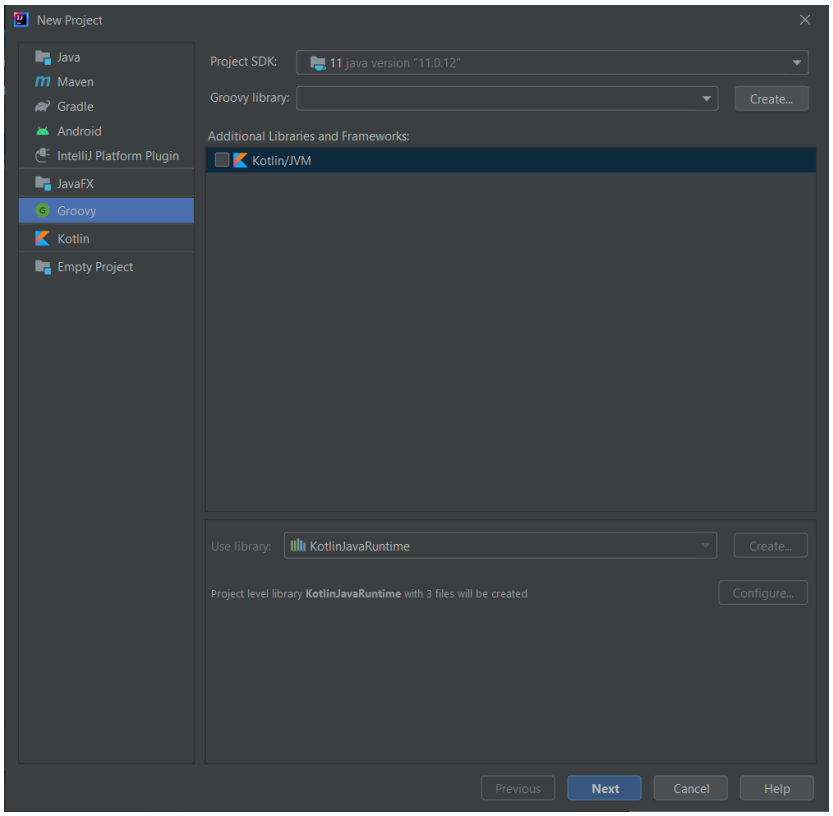
1. Jelaskan mengenai konsep Class dan Objek, apa perbedaan keduanya.
2. Apa yang dimaksud Constructor , bagaimana penerapannya.
3. Buat tampilan berisi nama , NIM, dan kata Hello World pada IDE Anda.

Langkah-Langkah Praktikum

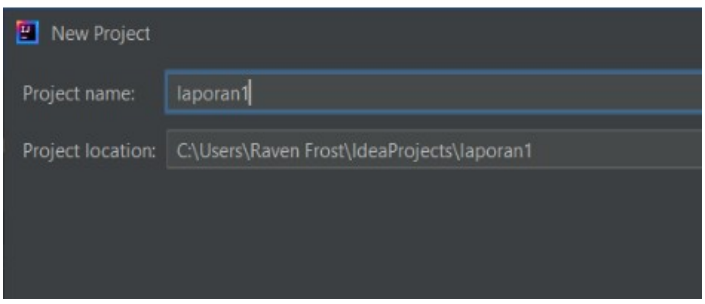
Tugas 1: Membuat proyek "Hello Toast" yang baru

1.1 Create New Project

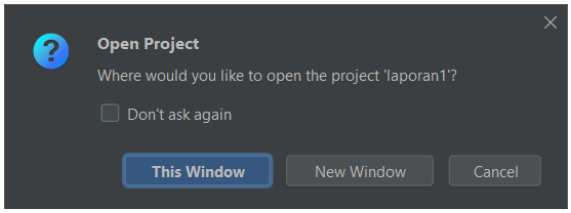
1) Bukalah aplikasi Intellij IDEA lalu pilih File > New > Project. Pilih versi java yang sudah diunduh dan klik Next



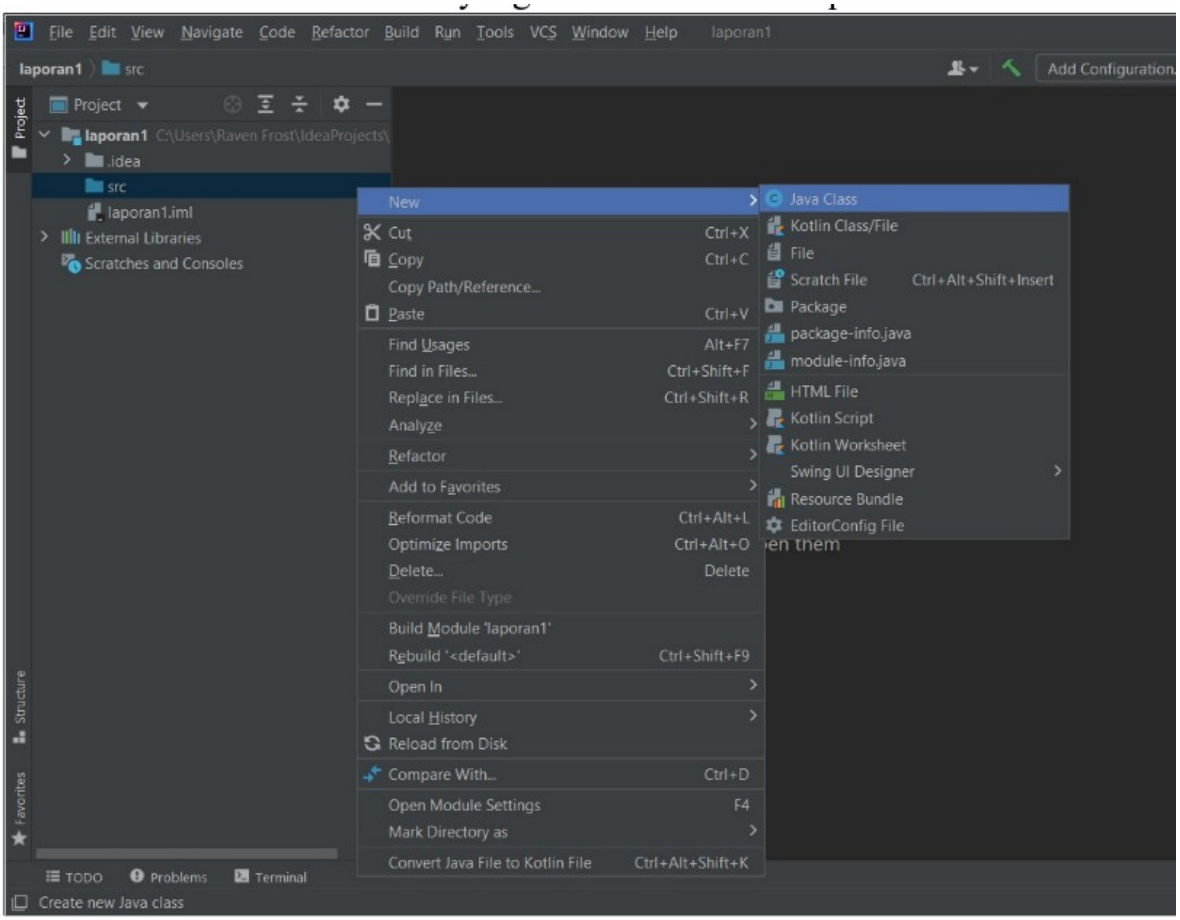
2) Berikan nama pada project lalu klik Finish



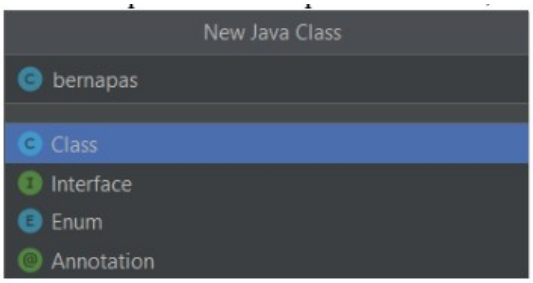
3) Akan muncul window untuk memilih ingin membuka project ini di window baru atau tidak. Silahkan pilih sesuai keinginan



4) Klik kanan di file sebelah kiri yang bernama “src” lalu pilih New > Java Class



5) Akan tampil window seperti di bawah, ini merupakan class. Berilah nama pada class



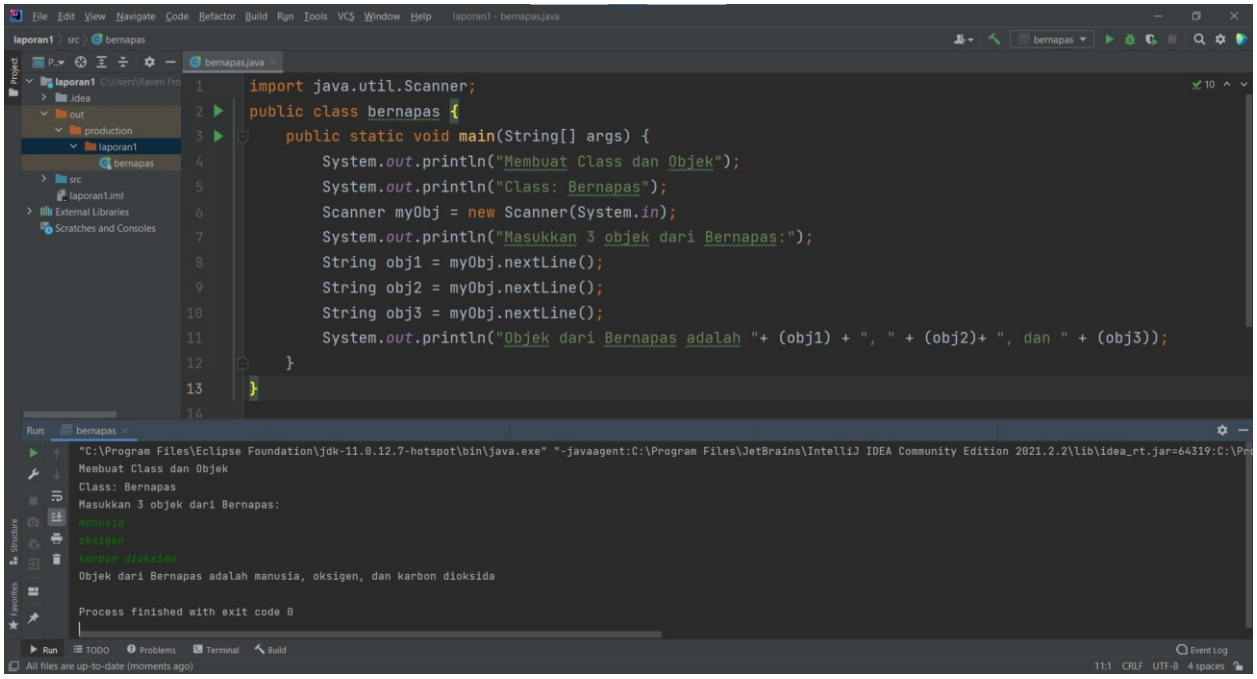
6) Ketikkan kode program java yang dapat menerima inputan dari user.

Catatan : Inputan user adalah objek

```
import java.util.Scanner;
public class bernapas {
    public static void main(String[] args) {
        System.out.println("Membuat Class dan Objek");
        System.out.println("Class: Bernapas");
        Scanner myObj = new Scanner(System.in);
        System.out.println("Masukkan 3 objek dari Bernapas:");
        String obj1 = myObj.nextLine();
        String obj2 = myObj.nextLine();
        String obj3 = myObj.nextLine();
        System.out.println("Objek dari Bernapas adalah "+ (obj1) + ", " + (obj2)+ ", dan " + (obj3));
    }
}
```

7) Jika sudah dan tidak terdapat error merah pilih menu Run di tools bagian atas dan pilih Run "(nama class)". Jika tidak bisa, silahkan Build terlebih dahulu class dengan menekan F7 lalu Ctrl + Alt + Shift + R

8) Sesuai dengan kode program yang telah dibuat, user akan diminta untuk memasukkan 3 jenis objek dari bernapas. Disini, user menginput manusia, oksigen, dan karbon dioksida sehingga outputnya adalah sebagai berikut



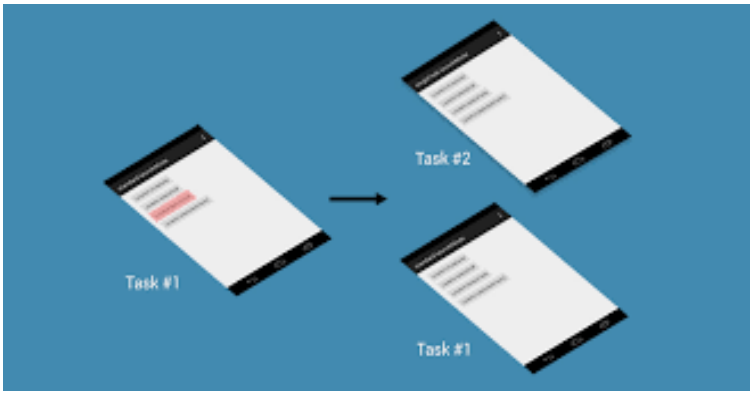
Analisa

1. Di bahasa pemrograman Java, kita harus mengompile atau Build hasil project terlebih dahulu sebelum menjalankannya

2. Di aplikasi Intellij IDEA akan muncul notifikasi seperti peringatan jika kode yang sedang diketik salah/tidak sesuai dengan format bahasa yang digunakan sehingga kita dapat menghindari terjadinya kesalahan sebelum menjalankan program

Tugas Laporan

[Diberikan saat praktikum]



Modul Praktikum 2: Encapsulation

Deskripsi Pertemuan

Modul ini membahas mengenai salah satu pilar OOP yaitu Encapsulasi atau pembungkusan.

Syarat Kompetensi

1. Membuat dan menjalankan aplikasi dalam Android Studio.
2. Membuat dan mengedit elemen UI dengan Layout Editor grafis atau langsung di file layout XML.
3. Menambahkan fungsionalitas onClick ke Button.
4. Membuat dan menggunakan aktivitas.

Standar Kompetensi

1. Membuat package dalam sebuah proyek.
2. Menghubungkan antar obyek dengan Object Interaction.
3. Menggunakan method dengan nilai return dan access modifier.
4. Mengetahui cara penggunaan access modifier : public dan private.

Dasar Teori

Pembungkusan merupakan konsep yang mana objek-objek seperti data-data dan method dikemas atau dibundling di dalam kelas. Dengan Pembungkusan, objek

memiliki karakter nya sendiri, namun dapat dimanipulasi sesuai keperluan. Manipulasi ini dapat dilakukan dengan *modifier-modifier.*

Access Modifier

Salah satu tipe modifier adalah access modifier. Ada beberapa macam access modifier yang dapat digunakan yaitu private, default, protected, dan public.

Tugas Pendahuluan

1. Berikan dan uraikan dalam bentuk contoh kegunaan dan perbedaan dari setiap *access modifier.*
2. Jelaskan mengenai reference *this*

Langkah-Langkah Praktikum

Tugas 1

Private, access modifier private akan membatasi akses hanya di dalam class. Private biasanya digunakan sebagai modifier dari member dan metode suatu class.

1.1 Buatlah proyek baru dengan nama AccessModifier dan nama package

latihan.encapsulation.namaAnda

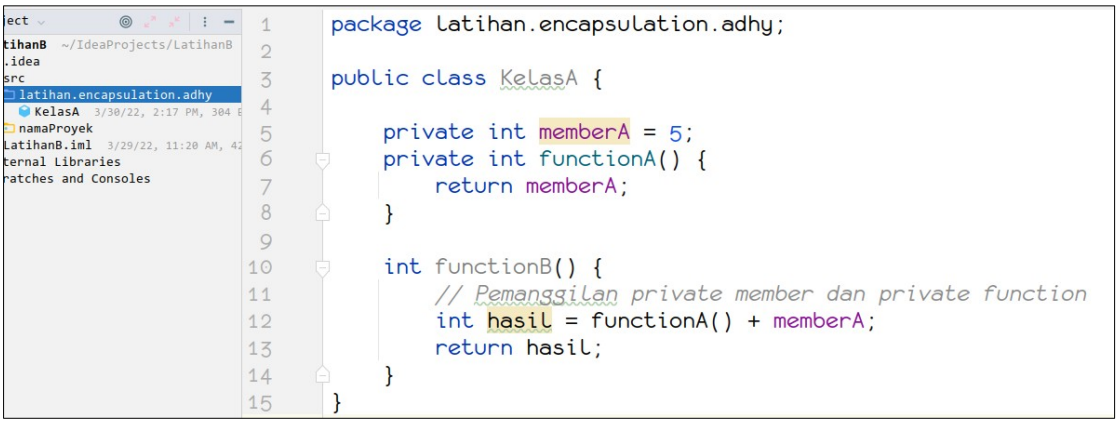
sehingga di explorer tercipta susunan folder seperti :

```
[~/IdeaProjects/LatihanB/src]$ cd latihan
[~/IdeaProjects/LatihanB/src/latihan]$ cd encapsulation
[~/IdeaProjects/LatihanB/src/latihan/encapsulation]$ cd ad
[~/IdeaProjects/LatihanB/src/latihan/encapsulation/adhy]$
```

1. Buatlah sebuah kelas baru di dalamnya dengan nama KelasA, kemudian tambahkan kode berikut:

   Perhatikan nama file fisik di explorer sama dengan yang tertulis di editor yaitu sama **“KelasA”.**

   Lihat pada baris 5 & 6 dimana kita memasang private pada kedua obyek yaitu variable memberA dan method functionA.



2. Buatlah kelas Main dan tambahkan kode berikut:

```
public class Main {
    public static void main(String[] args) {
        KelasA kelasA = new KelasA();

        System.out.println(kelasA.memberA);
        System.out.println(kelasA.functionA());

        System.out.println(kelasA.functionB());
    }
}
```

kode diatas akan terjadi eror karena memberA dan functionA dalam keadaan private, hal tersebut mengatikbatkan tidak bisa diakses dari luar kelas.

3. Sekarang ubah kode di Langkah 2 menjadi seperti ini:

```
public class Main {
    public static void main(String[] args) {
        KelasA kelasA = new KelasA();
        System.out.println(kelasA.functionB());
    }
}
```

4. Bila sukses, seharusnya Console akan menampilkan output seperti ini.

```
/usr/local/java/java-8-lama/bin/java ...
10

Process finished with exit code 0
```

1.2 Access Modifier Default

1. Mari kita tambahkan beberapa default modifier baru pada contoh sebelumnya yaitu KelasA.

```
3   public class KelasA {
4
5       private int memberA = 5;
6
7       char memberB = 'A';
8       double memberC = 1.5;
9
10      private int functionA() {
11          return memberA;
12      }
```

Tidak menambahkan public atau private pada baris ke 7 dan 8

2. Bukalah kelas Main dan tambahkan kode berikut:

```
3   public class Main {
4       public static void main(String[] args) {
5           KelasA kelasA = new KelasA();
6           System.out.println(kelasA.functionB());
7
8           System.out.println(kelasA.memberB);
9           System.out.println(kelasA.memberC);
10      }
11  }
```

3. Jalankan kode di atas

Tugas 2

Default

Default modifier berarti penulisan kodenya tanpa atribut modifier. Ini berlaku untuk semua kelas, member, atau fungsi yang kita tuliskan tanpa access modifier. Modifier default bisa diakses selama masih dalam satu package.

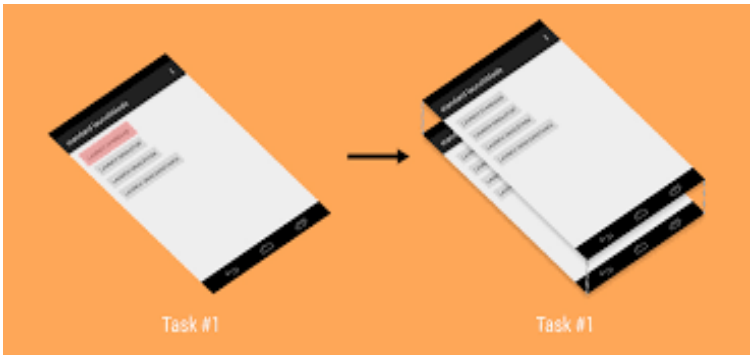
Tugas 3

Protected

Access modifier **protected** bisa diakses selama masih dalam satu package. Protected memiliki sedikit perbedaan dengan default modifier. Perbedaannya adalah protected bisa diakses dari luar package. Akan tetapi, satu-satunya cara untuk akses dari luar package adalah kelas yang hendak mengakses, merupakan kelas turunannya.

Tugas Laporan

[Diberikan saat praktikum]



Modul Praktikum 3:

Java Basic : Percabangan

Deskripsi Pertemuan

Pada bagian ini, kita mempelajari tentang struktur kontrol yang bertujuan agar kita dapat menentukan urutan statement yang akan dieksekusi. Struktur kontrol keputusan adalah statement dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode dan mengabaikan blok kode yang lain. Ada tiga macam perintah percabangan, yaitu **if, if ... else,** dan **switch.**

Syarat Kompetensi

1. Membuat dan menggunakan aktivitas.
2. Membuat dan mengirim intent di antara aktivitas.

Standar Kompetensi

1. Membuat aplikasi baru untuk mengirimkan intent implisit.
2. Mengimplementasikan dua intent implisit yang membuka halaman web dan membuka lokasi pada peta.
3. Mengimplementasikan tindakan untuk membagikan cuplikan teks.
4. Membuat aplikasi baru yang dapat menerima intent implisit untuk membuka laman web.

Dasar Teori

Mahasiswa diharapkan dapat menggunakan struktur kontrol keputusan (if, else, switch) yang digunakan untuk memilih blok kode yang akan dieksekusi, menggunakan struktur kontrol pengulangan (while, do-while, for) yang digunakan untuk melakukan pengulangan pada blok kode yang akan dieksekusi, menggunakan statement percabangan (break, continue, return) yang digunakan untuk mengatur redirection dari program. Pada bab sebelumnya, kita sudah mendapatkan contah dari program sequential, dimana statement dieksekusi setelah statement sebelumnya dengan urutan tertentu.

◦ Percabangan if merupakan percabangan yang hanya memiliki satu blok pilihan saat kondisi bernilai benar.

◦ Percabangan if/else merupakan percabangan yang memiliki dua blok pilihan. Pilihan pertama untuk kondisi benar, dan pilihan kedua untuk kondisi salah (else).

◦ Percabangan if/else/if merupakan percabangan yang memiliki lebih dari dua blok pilihan.

◦ Percabangan switch/case adalah bentuk lain dari percabangan if/else/if.

Tugas Pendahuluan

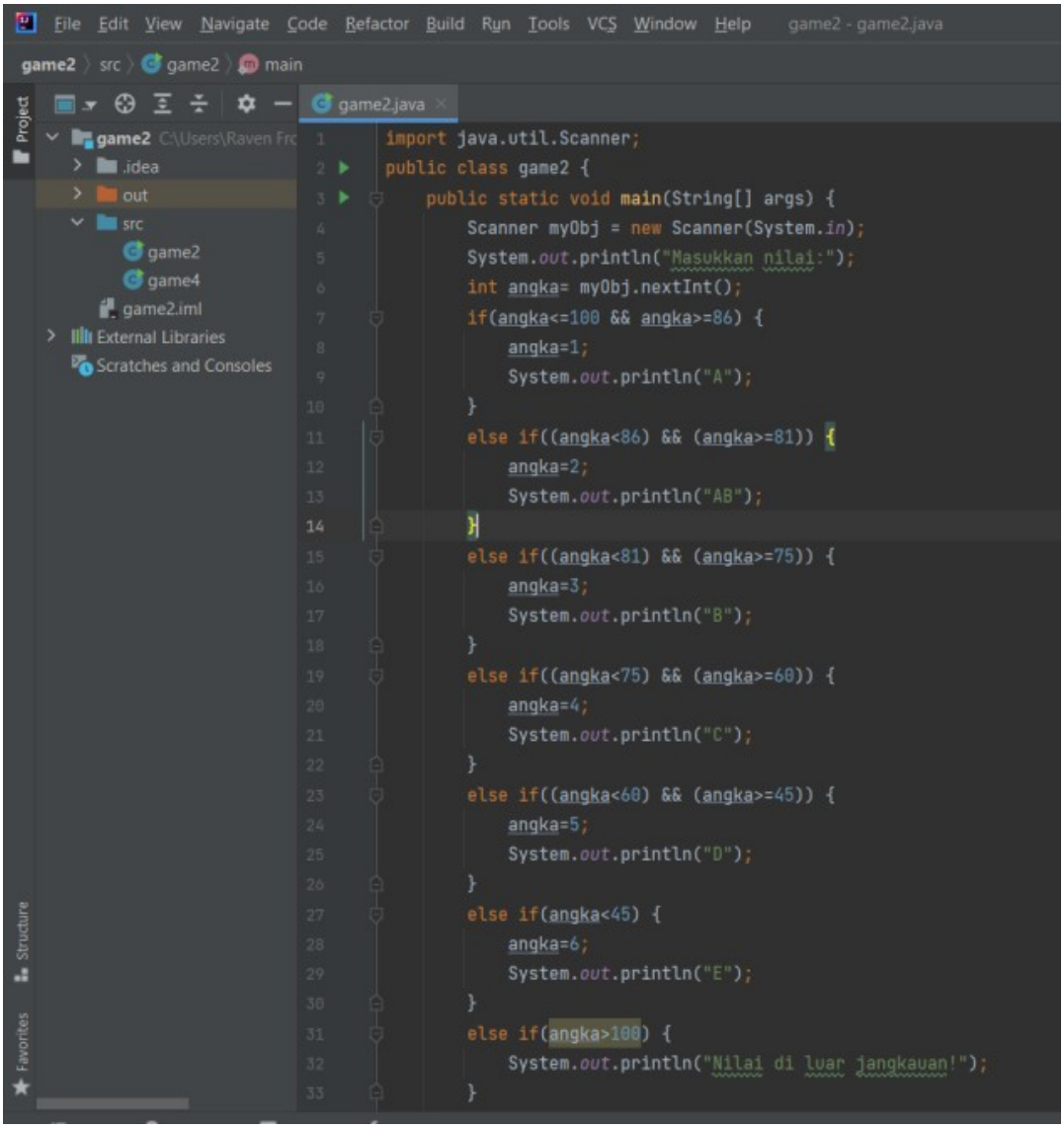
1. Jelaskan perbedaan switch case dan if else...
2. Sebutkan notasi percabangan pada flowchart

Langkah-Langkah Praktikum

Tugas 1

1.1 Membuat struktur pengkondisian “if ... else”

1. Buat proyek baru dengan nama Percabangan dan nama package latihan.percabangan.namaAnda
2. Isi method main sesuai gambar berikut



Tugas 2

2.1 Membuat struktur pengkondisian “switch .. case”

1. Lanjutkan code di atas dengan seperti gambar berikut

```java
else if(angka<45) {
    angka=6;
    System.out.println("E");
}
else if(angka>100) {
    System.out.println("Nilai di luar jangkauan!");
}
switch (angka) {
    case 1:
        System.out.println("Pintar");
        break;
    case 2:
        System.out.println("Rajin");
        break;
    case 3:
        System.out.println("LULUS");
        break;
    case 4:
        System.out.println("LULUS");
        break;
    case 5:
        System.out.println("TIDAK LULUS");
        break;
    case 6:
        System.out.println("TIDAK LULUS");
}
}
}
```

2. Jalankan, sehingga output seperti berikut

```
"C:\Program Files\Eclipse Foundation\jdk-11.0.12.7-hotspot\bin\java.exe" "-jav
Masukkan nilai:
86
A
Pintar

Process finished with exit code 0
```

Tugas Laporan

[Diberikan saat praktikum]

Modul Praktikum 4:

Java Basic : Perulangan

Deskripsi Pertemuan

Pada praktek ini, mahasiswa mempelajari cara menggunakan perintah for dan for-each atau yang biasa disebut dengan perulangan (looping) pada bahasa Java. Untuk melatih kemampuan mahasiswa, maka akan berlatih untuk 2 buah pertanyaan, yang pertama menggunakan tipe data array dan yang kedua menggunakan variabel input. Dengan ini, mahasiswa akan menerapkan konsep perulangan yang digabung dengan pengkondisian.

Syarat Kompetensi

1. Mengakses elemen UI dari kode Anda menggunakan findViewById()
2. Mengonversi teks dalam tampilan menjadi string menggunakan getText().toString()
3. Menangani klik Button.
4. Menampilkan pesan toast.
5. Memulai aktivitas dengan aplikasi lain menggunakan intent implisit.

Standar Kompetensi

1. Menambahkan peringatan dengan Oke dan Batal untuk mengambil keputusan pengguna.
2. Menambahkan kontrol masukan spinner untuk menampilkan menu tarik-turun dengan nilai, tempat pengguna bisa memilih salah satunya.

3. Menambahkan picker tanggal dan picker waktu ke proyek baru dan menggunakan listener untuk merekam pilihan pengguna.
4. Menyetel handler onClick untuk gambar yang digunakan sebagai Button untuk meluncurkan aktivitas kedua.

Dasar Teori

Perulangan atau dalam istilah lain disebut dengan loop. **Perulangan** dipakai ketika kita dihadapkan pada suatu masalah dalam jumlah besar yang membutuhkan penyelesaian terkadang sama dengan pola yang telah kita ketahui. Contoh perulangan dikehidupan nyata. Tuliskan "Ini adalah Perulangan" sebanyak 100 kali.

Kita lihat ada 100 masalah dengan 100 penyelesaian sama dengan pola yang sama, disinilah perulangan sangat berguna, mengefisiensikan pekerjaan kita.

Pada bahasa pemrograman Java ada 3 macam perulangan, dengan menggunakan for, while-do, dan do-while.

If statements digunakan untuk mengevaluasi ekspresi yang ada didalam tanda kurung (). Jika hasilnya TRUE maka kode yang ada didalam if akan dijalankan. Namun jika hasilnya False maka kode didalam if tidak akan dijalankan dan hanya dilewati begitu saja.

If else, bedanya kita memiliki kondisi kedua jika kondisi pertama tidak terpenuhi. Jika hasilnya True maka kode program yang ada didalam if akan dijalankan, sedangkan kode yang ada didalam else akan diabaikan. Namun jika hasilnya false maka kode program yang ada didalam if yang diabaikan, dan akan menjalankan kode yang ada didalam else

break adalah sebuah pernyataan yang digunakan untuk melompat dari sebuah pernyataan looping, maupun kondisi if else dan switch case.

Perulangan for atau looping for termasuk dalam jenis perulangan yang cukup banyak digunakan bukan hanya di java namun dibahasa pemrograman lain seperti PHP, C++ dan Python. Perulangan ini digunakan ketika ingin mengeksekusi perintah program yang sama dengan jumlah proses perulangan yang sudah diketahui dengan mengacu pada kondisi yang ditetapkan.

continue merupakan pernyataan jika suatu kondisi tertentu terjadi maka akan langsung dilanjutkan dengan iterasi selanjutnya dalam sebuah looping atau perulangan.

Return merupakan salah satu keyword yang digunakan untuk menyelesaikan eksekusi suatu metode dan dapat digunakan untuk mengembalikan nilai dari suatu metode.

Tugas Pendahuluan

Buat sebuh program perulangan sederhana yang mencetak karakter * di terminal jadi bangun segitiga sama sisi.

Langkah-Langkah Praktikum

Tugas 1 Perulangan untuk operasi tiap elemen array

Latihan ini kita akan menentukan jumlah nilai A pada sebuah himpunan nilai-nilai ujian (angka). Untuk rentang nilai huruf adalah :

- A adalah nilai untuk angka yang lebih besar atau sama dengan 81,
- B adalah nilai untuk angka yang lebih besar atau sama dengan 75 namun kurang dari 81,
- C adalah nilai untuk angka yang lebih besar atau sama dengan 60 namun kurang dari 75,
- D adalah nilai untuk angka yang lebih kecil dari 60.

1.1 Salin code dari gambar berikut ke sebuah proyek Java baru.

a. buat 1 variabel tipe array dengan 10 inputan nilai di dalamnya

b. Gunakan fungsi for (looping) untuk operasi otomatis pada setiap iterasi elemen/data array.

   a) dan pada setiap iterasi lakukan pengecekan apakah memenuhi syarat nilai A.

   b) Perintah count++ berfungsi untuk merubah value dari var count, yang tugasnya menyimpan informasi banyaknya jumlah nilai A. (dapat dituliskan seperti (i + 1))

c. Perintah continue berfungsi untuk tetap menjalankan program walaupun sudah menemukan elemen array yang bernilai true sampai batas pada variabel array.

```java
public class no1 {
    public static void main(String[] args) {
        int angka[]= {45, 56, 57,89,78,23,68,99,96,75};
        int count=0;
        for(int i=0; i< angka.length; i++){
            if(angka[i]>= 81) {
                count++;
                continue;
            }
        }
        System.out.println("Nilai A ada "+ count);
        for(int i=0; i< angka.length; i++){
            if(angka[i]>= 75 && angka[i]<81) {
                count++;
                continue;
            }
        }
        for(int i=0; i< angka.length; i++){
            if(angka[i]>= 60 && angka[i]<75) {
                count++;
                continue;
            }
        }
        for(int i=0; i< angka.length; i++){
            if(angka[i]<60) {
                count++;
                continue;
            }
        }
    }
}
```

1.2 Running dan pastikan output sesuai .

```
Run: no1
"C:\Program Files\Java\jdk-17\bin\java.exe" "-jav
Nilai A ada 3

Process finished with exit code 0
```

Tugas 2 : Menentukan predikat nilai

Latihan ini kita akan menentukan predikat nilai setiap Mata Kuliah dalam satu Semester berdasarkan nilai /skor masing- masing. Untuk rentang nilai adalah :

- **"Perfect"** adalah untuk angka yang lebih besar atau sama dengan 81,
- **"Good Job"** adalah untuk angka yang lebih besar atau sama dengan 75 namun kurang dari 81,
- **"Average"** adalah untuk angka yang lebih besar atau sama dengan 60 namun kurang dari 75,
- **"Poor"** adalah untuk angka yang lebih kecil dari 60.
- **"Out of Bound"** adalah untuk angka yang lebih kecil dari 0.

2.1 Salin code dari gambar berikut ke sebuah proyek Java baru.

```
import java.util.Scanner;
public class Latihan_2 {

    public static void main(String[] args) {
        for (int input = 1; input <= 6; input++) {
            Scanner key = new Scanner(System.in);
            System.out.print("Masukkan nilai : ");

            int grade = key.nextInt();
            if ((grade <= 100) && (grade >= 81)) {
                System.out.println("Perfect!");
            }else if ((grade < 80) && (grade >= 75)) {
                System.out.println("Good job");
            }else if ((grade < 74) && (grade >= 60)) {
                System.out.println("Average");
            }else if ((grade < 60) && (grade >= 0)) {
                System.out.println("Poor");
            }else {
                System.out.println("Out of Bound");
            }
        }
    }
}
```

- Buat 1 variabel int yang akan menerima inputan dari keyboard
- User akan memasukkan nilai sebanyak 5 kali (dari 1-100) dimana setiap nilai akan mencetak kata sesuai dengan nilai yang diinputkan. Contoh, nilai 90 akan mencetak kata Perfect

2.2 Running dan pastikan output sesuai .

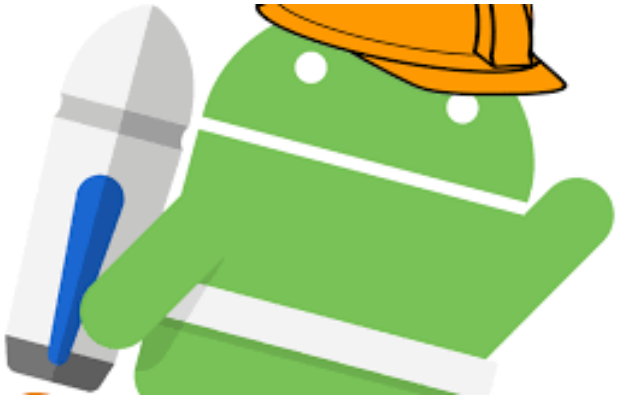
```
"C:\Program Files\Java\jdk-17\bin\java.exe"
Masukkan nilai anda: 100
Perfect!
Masukkan nilai anda: 76
Good job
Masukkan nilai anda: 66
Average
Masukkan nilai anda: 50
Poor
Masukkan nilai anda: 101
Out of Bound
Masukkan nilai anda: -2
Out of Bound

Process finished with exit code 0
```

Dengan ini, kita dapat menggabungkan dua perintah yaitu if-else untuk pengkondisian dan for untuk mencetak kata/kalimat yang diinginkan jika terdapat perintah if-else yang bernilai True.

Tugas Laporan

[Diberikan saat praktikum]



Modul Praktikum 5: Inheritance

Deskripsi Pertemuan

Dalam modul ini, kita akan membicarakan konsep Inheritance (pewarisan) , yaitu bagaimana suatu class dapat mewariskan sifat dari class yang sudah ada. Class ini dinamakan subclass dan induk class dinamakan superclass. Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object.

Syarat Kompetensi

1. Mengakses elemen UI dari kode Anda menggunakan findViewById()
2. Mengonversi teks dalam tampilan menjadi string menggunakan getText().toString()
3. Menangani klik Button.
4. Menampilkan pesan toast.
5. Memulai aktivitas dengan aplikasi lain menggunakan intent implisit.

Standar Kompetensi

1. Menambahkan item menu ke menu opsi.
2. Menambahkan ikon untuk item menu agar tampil di bilah tindakan.
3. Menghubungkan klik item menu ke handler kejadian yang memproses kejadian klik.

Dasar Teori

Dalam bagian ini, kita akan membicarakan bagaimana suatu class dapat mewariskan sifat dari class yang sudah ada. Class ini dinamakan subclass dan induk class dinamakan superclass.

Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object. Contoh hirarki class diperlihatkan di bawah ini.

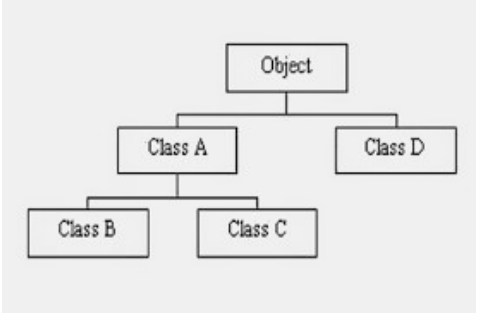


*Figure 1: Hirarki Objek Class**

Beberapa class di atas class utama dalam hirarki class dikenal sebagai superclass. Sementara beberapa class di bawah class pokok dalam hirarki class dikenal sebagai subclass dari class tersebut. Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclasss atau child – child nya.

*Image source: http://tugas068.blogspot.com/

Tugas Pendahuluan

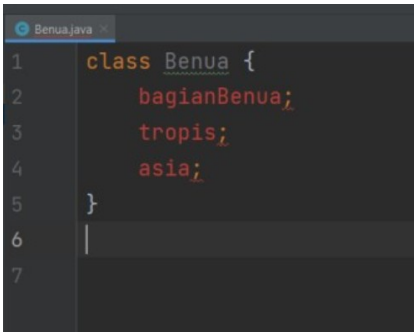
- Jelaskan mengenai properti dan behaviour di dua kelas untuk sebuah Pewarisan tunggal.
- Jelaskan perbedaan method pada superclass dan subclass

Langkah-Langkah Praktikum

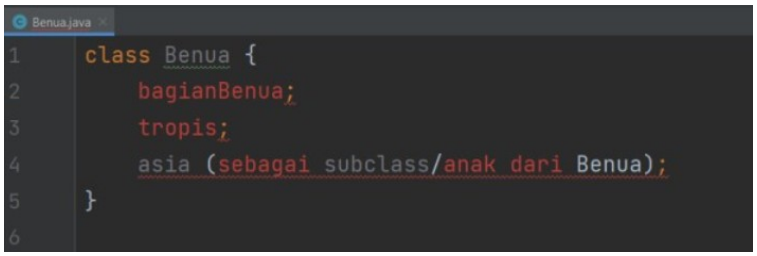
Tugas 1: Class turunan dan elemennya

Dalam tugas ini Anda akan melatih pembuatan class induk dan class child lalu bagaimana menuliskan elemen – elemen dari sebuah class turunan.

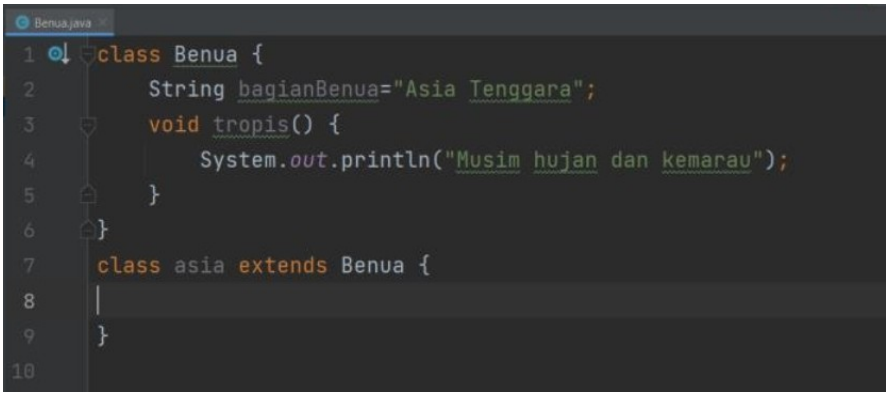
1.1 Identifikasi sebuah objek besar , lalu buat Class dan tuliskan objek – objek kecil atau data properti pendukungnya.



1.2 Identifikasi subClass atau anak dari 1.1



1.3 Buat No.2 sebagai Class turunan



1.4 Generate method-method (Override) milik Class Parent di Class turunan

```java
class Benua {
    String bagianBenua="Asia Tenggara";
    void tropis() {
        System.out.println("Musim hujan dan kemarau");
    }
}
class asia extends Benua {
    static void main(String[] args) {
        asia myCountry = new asia();
        myCountry.tropis();
    }
}
```

1.5 Buat method-method sendiri (Class turunan)

```java
class Benua {
    String bagianBenua="Asia Tenggara";
    void tropis() {
        System.out.println("Musim hujan dan kemarau");
    }
}
class asia extends Benua {
    String namaNegara= "Indonesia";
    static void main(String[] args) {
        asia myCountry = new asia();
        myCountry.tropis();
        System.out.println(myCountry.bagianBenua + " " + myCountry.namaNegara);
    }
}
```

1.6 Melengkapi class & method dengan modifier : public/protected/private

```java
public class Benua {
    protected String bagianBenua="Asia Tenggara";
    public void tropis() {
        System.out.println("Musim hujan dan kemarau");
    }
}
class asia extends Benua {
    private String namaNegara= "Indonesia";
    public static void main(String[] args) {
        asia myCountry = new asia();
        myCountry.tropis();
        System.out.println(myCountry.bagianBenua + " " + myCountry.namaNegara);
    }
}
```

Analisis

- Salah satu cara untuk menentukan subclass (child) dari suatu superclass (parent) adalah dengan menjabarkan komponen superclass tersebut dalam bentuk hirarki
- Panggil method-method yang ada di class parent menggunakan psvm di class child
- Pastikan method yang terdapat di class parent berupa public atau protected, bukan private, agar class turunannya tetap dapat mengakses method tersebut
- Method pada class child dibolehkan bersifat private karena hanya akan diakses oleh classnya sendiri.

Tugas Laporan

[Diberikan saat praktikum]

Modul Praktikum 6: Polymorphism

Deskripsi Pertemuan

Modul ini akan digunakan sebagai pembelajaran mengenai Polymorphism (banyak bentuk).

Syarat Kompetensi

1. Mengerti menggunakan array pada Java
2. Membuat aplikasi Hello World dengan Android Studio.
3. Mengimplementasikan layout yang berbeda untuk aplikasi.
4. Membuat dan menggunakan sumber daya string.
5. Menambahkan handler onClick ke tampilan.

Standar Kompetensi

1. Dapat mendesain tampilan list untuk sebuah aplikasi Android
2. Menggunakan kelas RecyclerView untuk menampilkan item di daftar yang bisa digulir.
3. Secara dinamis menambahkan item ke RecyclerView saat terlihat melalui pengguliran.
4. Melakukan tindakan saat pengguna mengetuk item tertentu
5. Menampilkan Button aksi mengambang dan melakukan tindakan saat pengguna mengetuknya.

Dasar Teori

Sebelumnya, kita telah mengetahui yang dinamakan subclass dan induk class dinamakan superclass. Dalam bagian ini, kita akan membicarakan sifat khusus dari Java dimana kita dapat secara otomatis memakai method yand tepat untuk setiap object tanpa memperhatikan asal dari subclass object. Sifat ini dinamakan polimorfisme.

Kita telah mengenal pewarisan pada Modul sebelumnya. Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass. Subclass hanya butuh mengimplementasikan perbedaannya sendiri dan induknya. Ini termasuk penerapan dari polimorfisme. Penerapan berikutnya yaitu pada penggunaan Interface yang akan dibahas di modul berikutnya.

Tugas Pendahuluan

1. Jelaskan pengertian abstract class
2. Jelaskan mengenai Overloading dan Override method.
3. Jelaskan mengenai final pada variable dan final pada method

Langkah-Langkah Praktikum

Tugas 1: Generalisasi objek Person

1.1. Buat proyek baru

Buat proyek baru dengan nama Inheritance, lalu masukkan pengkodean berturut -turut dari bawah ini:

```
public class Person {

	protected String name;

	protected String address;

	/**
	 * Default constructor
	 */

	public Person(){

		System.out.println("Inside Person:Constructor");

		name = "";

		address = "";

	}

	/**
	* Constructor dengan dua parameter

	*/

	public Person( String name, String address) {

		this.name = name;

		this.address = address;

	}

	/**
	 * Method accessor
	 */

	public String getName() {

		System.out.println("Person Name : " +name);

		return name;
```

```
        }
        public String getAddress() {
                return address;
        }
        public void setName(String name) {
                this.name = name;
        }
        public void setAddress(String add) {
                this.address = add;
        }
}
public class Student extends Person{
         public Student()

        {
                  //super( "SomeName", "SomeAddress");
                  //super();
                  //super.name = "name";
                 System.out.println("Inside Student:Constructor");
         }
         public String getName() {
              System.out.println("Student Name : " +name);
              return name;
         }
         public static void main( String[] args) {
              Student anna = new Student();
```

```
        }
}
public class Employee extends Person{
    public String getName() {
        System.out.println("Employee Name:" +name);
        return name;
    }
    public static void main(String[] args)
    {
        Person ref;
        Student studentObject = new Student();
        Employee employeeObject = new Employee();

        ref = studentObject; //Person menunjuk kepada object Student

        String temp = ref.getName(); //getName dari Student class dipanggil
        System.out.println(temp);

        ref = employeeObject; //Person menunjuk kepada object Employee

        temp = ref.getName(); //getName dari Employee class dipanggil
        System.out.println(temp);
    }
}
```

Jalankan aplikasi Anda.

Tugas Laporan

[Diberikan saat praktikum]

Modul Praktikum 7: Abstraction

Deskripsi Pertemuan

Abstraksi adalah Pilar OOP yang berikutnya. Pada OOP kita berpikir abstraksi, yaitu kita tidak memperdulikan How/proses nya sesuatu objek, selama kita mengerti output atau outcome nya.

Pada praktiknya kita membuat class yang bekerja/bertingkah sesuai dengan fungsi utama objek asli. Kita membuat class kemudi mobil yang bekerja sesuai setir sebagai kemudi di sebuah mobil nyata. Dan kita tidak memperdulikan cara bekerja yang detail dari alat-alat mobil yang bekerja dengan setir mobil tersebut.

Modul ini akan digunakan sebagai pembelajaran mengenai abstraksi dengan cara memanage type Interface dan abstract pada pemrograman Java. Bagaimana menggunakan dan seperti apa perbedaan interface dengan abstract type.

Syarat Kompetensi

1. Membuat sebuah Aktivitas.
2. Menambahkan TextView ke layout untuk aktivitas tersebut.
3. Mendapatkan id untuk TextView dan menyetel kontennya secara terprogram.
4. Menggunakan tampilan Button dan fungsionalitas onClick.

Standar Kompetensi

1. Membuat aplikasi sederhana yang mengeksekusi tugas backgroundmenggunakan AsyncTask.
2. Menjalankan aplikasi dan melihat apa yang terjadi saat memutar layar.
3. Membuat subkelas BroadcastReceiver untuk menampilkan Toast saat siaran diterima.
4. Mendaftarkan penerima untuk mendengarkan siaran sistem.
5. Mengirimkan dan menerima intent siaran khusus.

Dasar Teori

Interface adalah jenis khusus dari blok yang hanya berisi method signature (atau constant). Interface mendefinisikan sebuah(signature) dari sebuah kumpulan method tanpa tubuh. Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-class, tanpa memperhatikan lokasinya dalam hirarki class, untuk mengimplementasikan sifat-sifat yang umum. Dengan catatan bahwa interface-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.

Berbeda dengan Class biasa, Interface tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

Sebuah kelas dapat mengimplementasikan lebih dari satu interface. Kelas ini akan mendeklarasikan metode pada interface yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program.

Metode pada interface yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada interface tersebut. Property/Field di interface akan menjadi static final atau konstanta. Method dan field di interface akan selalu bersifat public.

Tugas Pendahuluan

- Jelaskan apa pengertian Interface
- Jelaskan perbedaan sehingga sebuah class adalah abstract class atau sebuah class adalah Interface

Langkah-Langkah Praktikum

Tugas 1: Implementasi Abstraction dengan Interface

1.1 Buatlah proyek baru dengan nama Interface

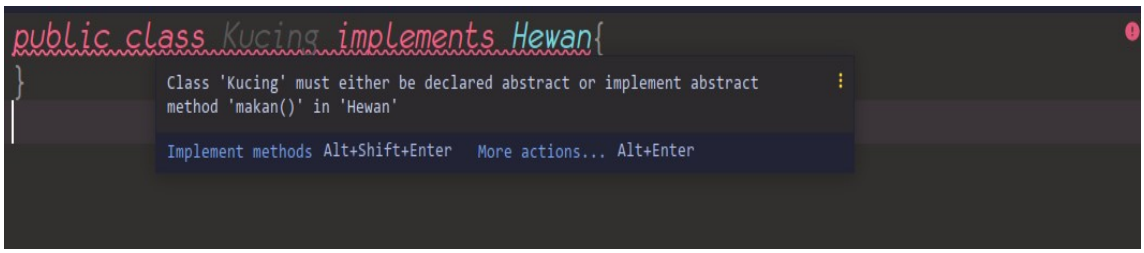
1.2 Buat interface di package yang baru saja Anda buat dengan cara : **Klik kanan** di package tersebut. Pilih **New - Java Class**. Name: Hewan, Kind: Interface, lalu tekan OK.

Lalu isi body dengan seperti berikut

```
public interface Hewan {

    String RESPIRASI = "oksigen";
    void makan() ;
}
```

1.3 Buatlah sebuah kelas baru dengan nama Kucing. Lalu implementasikan dengan kelas Hewan seperti ini:



Terlihat pesan error, yang mengatakan kita harus memilih untuk menjadikan kelas Kucing sebagai abstract class **atau** mendefinisikan isi kode semua deklarasi metode dari interface Hewan.

Jika kita ingin menjadikan kelas Kucing sebagai abstract class, cukup tambahkan keyword abstract sebelum keyword classseperti berikut.

```
public abstract class Kucing implements Hewan {

}
```

Jika kita ingin mengimplementasikan interface maka kita harus membuat method signature (nama, parameter, return) yang sama disertai isi kode program untuk metodenya. Berikut contoh implementasi metode makan.

```
public class Kucing implements Hewan {
    @Override
    public void makan() {

    }
}
```

Isikan print out di dalam **makan()** sesuai karakter dari objek.

1.4 Buat pemanggilan method seperti berikut

```
public class Main {

    public static void main(String[] args) {
        Kucing kucing = new Kucing();
        kucing.makan();
    }
}
```

dan lihat hasil output.

```
Main ×
/usr/local/java/java-8-lama/bin/java ...
Whiskas

Process finished with exit code 0
```

1.5 Buatlah sebuah kelas baru dengan nama Ayam. Lalu implementasikan dengan kelas Hewan seperti ini:

```
public class Ayam implements Hewan{
    @Override
    public void makan() {
        System.out.println("beras ji kodong");
    }
}
```

Isikan print out di dalam **makan()** sesuai karakter dari objek.

1.6 Ulangi dari langkah 1.3 untuk objek **Ayam**. Lalu bandingkan hasilnya

```java
public class Main {

    public static void main(String[] args) {
        Kucing kucing = new Kucing();
        kucing.makan();
        Ayam ayam = new Ayam();
        ayam.makan();
    }
}
```

Tugas Laporan

[Diberikan saat praktikum]

Modul Praktikum 8:

Representasi Tipe Data : Java Colllections

Deskripsi Pertemuan

Dalam praktik ini Anda akan membahas Java Collections Framework yang paling sering digunakan yaitu ArrayList, HashSet, dan HashMap.

Syarat Kompetensi

1. Mengimplementasikan metode onClick() untuk Button.
2. Membuat Intent Implisit.
3. Mengirim Intent Siaran Khusus.
4. Menggunakan Penerima Siaran.

Standar Kompetensi

1. Membuat Notifikasi menggunakan Builder Notifikasi.
2. Menggunakan Intent yang Tertunda untuk merespons tindakan Notifikasi.
3. Memperbarui atau membatalkan Notifikasi yang ada.

Dasar Teori

Pengertian

- Objek yang mengandung objek-objek lain.

  Bayangkan seperti tas yang berisi berbagai macam barang

- Collection digunakan untuk menyimpan, mendapatkan, memanipulasi dan mengkomunikasikan data kumpulan.

- Collection merepresentasikan item-item data yang membentuk grup,

Collection adalah secara teknis merupakan kumpulan *interface* yang digunakan sebagai wadah untuk mengumpulkan beberapa elemen menjadi satu kesatuan. Dari definisi tersebut bisa ditafsirkan Collection mirip dengan Array karena keduanya merepresentasikan struktur data.

Java Collections Framework terdiri dari tiga komponen, yaitu:

- Interface: Tipe abstrak dari Collection yang membentuk suatu hierarki.
- Implementation: Tipe konkrit yang mengimplementasikan Interface dari Collection serta dan merupakan suatu struktur data yang bisa langsung digunakan.
- Algorithm: Ini adalah method yang melakukan komputasi berguna, misal sorting atau searching.

Terdapat perbedaan Collection dengan Array. Misalnya Array tidak mempunyai method untuk *sorting* atau jumlah elemen di Array tidak bisa dinamis. Sementara Collection bisa dinamis.

Tugas Pendahuluan

- Jelaskan apa perbedaan Array dengan ArrayList
- Jelaskan perbedaan ArrayList dan LinkedList

Implementasi

Berikut adalah hirarki dan implmentasi dari Collection yang sering dipakai.

List	Set	Map
ArrayList LinkedList	TreeSet HashSet LinkedHashSet	TreeMap HashMap LinkedHashMap

Set

Set mewakili collection yang tidak boleh ada unsur duplikasi. Yang artinya jika objek yang sama dimasukkan beberapa kali ke dalam Set maka Set hanya akan menyimpan objek tersebut satu kali saja.

Contoh penggunaan:

- Permainan kartu
- Kursus pada jadwal
- Proses pada mesin

List

- Merupakan interface berupa rentetan/rangkaian
- Unsur pertama masuk akan tetap pada urutan pertama, urutan kedua masuk akan tetap pada urutan kedua, dst
- List membolehkan unsur yang sama/duplikat

Map

Map adalah struktur data dalam bentuk pasangan key-value. Map sama dengan "associative array " dalam bahasa PHP. Objek disimpan di Map sebagai value menggunakan key yang harus unik dan harus berupa objek juga. Salah satu implementasi dari interface Map adalah class HashMap.

Tugas 1: Membuat aplikasi penerapan ArrayList

1.1 Buatlah proyek baru atau pada proyek existing buat sebuah Class ListPLanet

1.2 Masukkan kode berikut ke dalamnya:

Lalu isi body main dengan seperti berikut

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class ListPlanet {

    public static void main(String[] args) {
        // deklarasi Array
        String[] heroes = new String[2];
        heroes[0] = "riki";
        heroes[1] = "sven";
        //heroes[2] = "zeus";
        //error ArrayIndexOutOfBoundsException karena ukuran Array tetap

        // menggunakan ArrayList
        List<String> planets = new ArrayList<>();
        // method add() untuk menambahkan objek ke List
        planets.add("mercury");
        planets.add("venus");
        planets.add("earth");
        planets.add("mars"); //objek masih bisa terus ditambahkan ke List
        System.out.println("List planets awal:");
        // method size() untuk mendapatkan ukuran List
        for (int i = 0; i < planets.size(); i++) {
            // method get() untuk melihat isi List pada index i
            System.out.println("\t index-"+ i +" = " + planets.get(i));
        }

        // method remove() untuk mengeluarkan objek dari List
        planets.remove( o: "venus");
        System.out.println("List planets akhir:");
        for (int i = 0; i < planets.size(); i++) {
            System.out.println("\t index-"+ i +" = " + planets.get(i));
        }
    }

}
```

Pastikan output berikut

```
List planets awal:
      index-0 = mercury
      index-1 = venus
      index-2 = earth
      index-3 = mars
List planets akhir:
      index-0 = mercury
      index-1 = earth
      index-2 = mars
```

1.3 Array vs ArrayList

Uncomment pada baris 12 → *heroes[2] = "zeus";*

```
public class ListPlanet {

    public static void main(String[] args) {
        // deklarasi Array
        String[] heroes = new String[2];
        heroes[0] = "riki";
        heroes[1] = "sven";
        heroes[2] = "zeus";
                Array index is out of bounds

        //error ArrayIndexOutOfBoundsException karena uk

        // menggunakan ArrayList
        List<String> planets = new ArrayList<>();
```

terlihat bahwa error *Array index is out of bonds;*

Ini terjadi karena pada Array tidak dapat menambahkan elemen lebih dari yang sudah dideklarasikan. Berbeda dengan ArrayList yang lebih dinamis

Tugas 2: Membuat aplikasi penerapan HashSet

2.1 Buatlah kelas baru dalam sebuah proyek baru dengan nama SetPlanet dan masukkan kode berikut

```
public class ListPlanet {

    public static void main(String[] args) {
        // menggunakan HashSet
        Set<String> planets = new HashSet<>();
        planets.add("mercury"); // method add() untuk menambahkan objek ke Set
        planets.add("venus");
        planets.add("earth");
        planets.add("earth"); // menambahkan objek "earth" beberapa kali
        planets.add("earth");
        planets.add("mars"); // objek bisa terus ditambahkan ke Set

        // method size() untuk mendapatkan ukuran Set
        System.out.println("Set planets awal: (size = " + planets.size() + ")");
        for (String planet : planets) {
            System.out.println("\t " + planet);
        }

        planets.remove("venus"); // method remove() untuk mengeluarkan objek dari Set

        System.out.println("Set planets akhir: (size = " + planets.size() + ")");
        for (Iterator iterator = planets.iterator(); iterator.hasNext(); ) {
            // looping menggunakan Iterator
            System.out.println("\t " + iterator.next());
        }
    }

}
```

- Iterator digunakan untuk scanning setiap index dalam sebuah var Set
- import java.util.* untuk menggunakan Iterator.
- Memilliki perintah add & remove

2.2 Running dan pastikan output sesuai .

```
Map planets awal: (size = 4)
     key-1 : Planet Mercury, mass: 0.06
     key-4 : Planet Mars-X, mass: 0.11
     key-3 : Planet Earth, mass: 1.0
     key-2 : Planet Venus, mass: 0.82
Map planets akhir: (size = 3)
     Planet Mercury, mass: 0.06
     Planet Mars-X, mass: 0.11
     Planet Earth, mass: 1.0
```

Tugas 3: Membuat aplikasi penerapan HashMap

2.1 Salin class code dari gambar berikut ke sebuah proyek Java baru.

```
public class Planet {

    public static void main(String[] args) {

    }

    private String name;
    private double mass;

    public Planet(String name, double mass) {
        this.name = name;
        this.mass = mass;
    }


    @Override
    public String toString() {
        return "Planet " + name + ", mass: " + mass;
    }
}
```

2.2 Lakukan pemanggilan dari class lain

```
import java.util.*;

public class ListPlanet{

    public static void main(String[] args) {
        // menggunakan HashMap
        Map<String, Planet> planets = new HashMap();

        // method put() untuk menambahkan objek ke Map
        planets.put("key-1", new Planet( name: "Mercury",  mass: 0.06));
        // menambahkan ke Map dengan key yang sama
        planets.put("key-2", new Planet( name: "Venus",  mass: 0.82));
        planets.put("key-3", new Planet( name: "Earth",  mass: 1.00));
        planets.put("key-4", new Planet( name: "Mars",  mass: 0.11));
        planets.put("key-4", new Planet( name: "Mars-X",  mass: 0.11));


        // method size() untuk mendapatkan ukuran Set
        System.out.println("Map planets awal: (size = "+ planets.size() +")");
        for (String key : planets.keySet()) { // looping key dari Map
            // method get() untuk melihat isi Map berdasarkan key
            System.out.println("\t " + key + " : " + planets.get(key));
        }

        // method remove()
        planets.remove( key: "key-2");
        // untuk mengeluarkan objek dari Map

        System.out.println("Map planets akhir: (size = "+ planets.size() +")");
        for (Planet planet : planets.values()) { // looping value dari Map
            System.out.println("\t " + planet);
        }
    }
}
```

Terdapat 3 perintah dalam sebuah HashMap

a) put

b) size

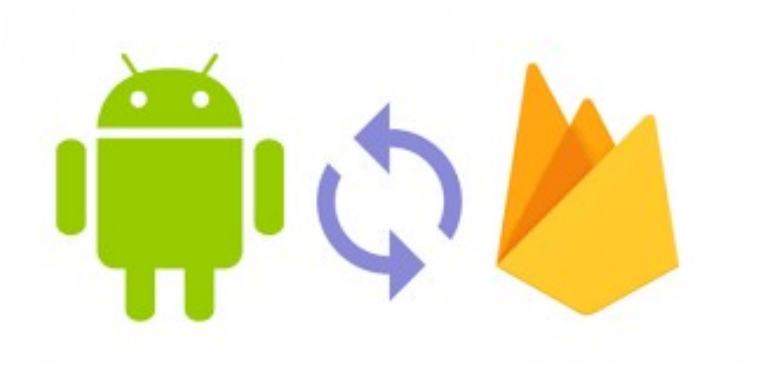
c) remove

2.3 Running dan pastikan output sesuai .

```
Map planets awal: (size = 4)
     key-1 : Planet Mercury, mass: 0.06
     key-4 : Planet Mars-X, mass: 0.11
     key-3 : Planet Earth, mass: 1.0
     key-2 : Planet Venus, mass: 0.82
Map planets akhir: (size = 3)
     Planet Mercury, mass: 0.06
     Planet Mars-X, mass: 0.11
     Planet Earth, mass: 1.0
```

Tugas Laporan

[Diberikan saat praktikum]



Modul Praktikum 9: Generic

Deskripsi Pertemuan

Modul ini akan membahas penggunaan Generic. Tipe generik sering ditemui dengan symbol <...> dan <T>.

Syarat Kompetensi

1. Mengimplementasikan list untuk menampilkan data
2. Membuat aktivitas dengan Intent
3. Mampu membuat dan merancang UI.
4. Mengerti konsep JSON sebagai data.

Standar Kompetensi

1. Menghasilkan tampilan data dari database (View data)
2. Mengolah data dari database (Edit data)
3. Merespon kesalahan pengguna terhadap database.

Dasar Teori

Generik berarti tipe berparameter. Idenya adalah untuk mengizinkan tipe (Integer, String, ... dll., bahkan tipe yang ditentukan pengguna) menjadi parameter untuk method, class, dan interface. Dengan Generik, dimungkinkan untuk membuat kelas yang bekerja dengan tipe data yang berbeda. Entitas seperti kelas, interface, atau metode yang beroperasi pada tipe berparameter adalah entitas generik.

Tugas Pendahuluan

1. Apa kegunaan dari Generics
2. Jelaskan mengenai Generic Class
3. Jelaskan mengenai Generic Method

Langkah-Langkah Praktikum

1. Buatlah proyek baru dengan nama Generics

2. Buatlah sebuah kelas baru di dalamnya dengan nama Planet, kemudian tambahkan kode berikut

```
class Planet {
    private String name;
    private double mass;

    public Planet(String name, double mass) {
        this.name = name;
        this.mass = mass;
    }

    public void print() {
        System.out.println("Planet " + name + ", mass: " + mass);
    }
}
```

3. Selanjutnya buat kelas Pemanggil, dan isi seperti berikut:

```java
import java.util.List;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {

        List lo = new ArrayList(); // List tanpa type-parameter
        lo.add("lo - String 1"); // lo menampung objek String

        // lo menampung objek Planet
        lo.add(new Planet( name: "Mercury",  mass: 0.06));

        // List dengan type-parameter Planet
        List<Planet> lp = new ArrayList();
        // lp menampung objek Planet
        lp.add(new Planet( name: "Mercury",  mass: 0.06));
        // baris berikut compile-error,
        // lp tidak diijinkan menampung objek String
        lp.add("lp - String 1");
    }
}
```

terlihat List<Planet> lp tidak diizinkan menampung objek selain Planet. Dalam kasus ini List<Planet> lp dilindungi compile-time type safety di mana jika ada objek lain yang dimasukkan ke List<Planet> lp . Tetapi dengan tipe selain Planet , seketika compile error, Artinya deteksilah lebih dahulu sebelum runtime (program dijalankan). Bandingkan dengan List lo yang bisa menampung objek String ataupun Planet (bahkan semua jenis objek). Sebagai contoh, kita ingin loop kedua List tersebut untuk memanggil method print dari class Planet.

Wildcards

wildcard type, ditulis dengan syntax Collection<?> yang artinya collection of unknown. Kita bisa menuliskan tipe data apa saja yang akan dilewatkan jika menggunakan <?>.

4. buat Main class baru atau pakai yang sudah ada, dan isi method main seperti berikut:

```
import java.util.List;
import java.util.ArrayList;
import java.util.Collection;

public class Main {

    public static void main(String[] args) {

        List<String> ls = new ArrayList();
        ls.add("String1");
        ls.add("String2");

        print(ls); // Apakah baris ini valid?

        Collection<Planet> cp = new ArrayList();
        cp.add(new Planet( name: "Mercury",  mass: 0.06));
        cp.add(new Planet( name: "Venus",  mass: 0.82));

        print(cp); // Apakah baris ini valid?
    }

    public static void print(Collection<Object> collection) {
        for (Object o : collection) {
            System.out.println(o);
        }
    }

}
```

Ubah tipe parameter Collection menjadi seperti berikut

```
    public static void print(Collection<?> collection)
```

Sekarang kode kita sebelumnya menjadi valid. Perhatikan sekarang method print() menjadi bisa dipanggil dari kedua tipe Collection yang berbeda yaitu List<String> dan Collection<Planet>.

Generic Methods

Bentuk generics methods dengan menuliskan huruf T dalam tag <>, sehingga menjadi <T>.

<T> berarti type yang dapat dilewatkan tidak ditentukan.

5. Buat sebuah class baru GenericsMethod, dan isi seperti berikut

```java
import java.util.ArrayList;
import java.util.Collection;

public class GenericsMethod {

//    private static <T> void arrayToCollection(T[] a, Collection<T> c) {
//        for (T o : a) {
//            c.add(o); // baris ini valid
//        }
//    }

    private static void arrayToCollection(Object[] a, Collection<?> c) {
        for (Object o : a) {
            c.add(o); // baris ini tidak valid
        }
    }

    public static void main(String[] args) {
        Object[] oa = new Object[100];
        Collection<Object> co = new ArrayList();

        // T inferred to be Object
        arrayToCollection(oa, co);

        String[] sa = new String[100];
        Collection<String> cs = new ArrayList();

        // T inferred to be String
        arrayToCollection(sa, cs);

        // T inferred to be Object
        arrayToCollection(sa, co);

        Integer[] ia = new Integer[100];
        Float[] fa = new Float[100];
        Number[] na = new Number[100];
        Collection<Number> cn = new ArrayList();

        // T inferred to be Number
        arrayToCollection(ia, cn);

        // T inferred to be Number
        arrayToCollection(fa, cn);

        // T inferred to be Number
        arrayToCollection(na, cn);

        // T inferred to be Object
        arrayToCollection(na, co);

        // compile-error
        //arrayToCollection(na, cs);
    }
}
```

Kode di atas tidak valid karena Collection<?> c adalah *collection of unknown type* dan kita menambahkan tipe Object o.

6. Ganti method arrayToCollection dengan Uncomment pada row 6 s/s 10, dan nonaktifkan row 12 s/s 16.

Wildcards → ketika balikan (return-type) dari suatu method tidak bergantung kepada tipe parameter (parameter-type). Atau kita hanya ingin memanfaatkan fitur polymorphism untuk tipe parameter method tersebut.

generic methods → Jika ada keterkaitan antara return-type dan parameter-type .

Tugas Laporan

[Diberikan saat praktikum]

Modul Praktikum 10: Exception handling

Deskripsi Pertemuan

Exception adalah event (kejadian) yang mengacaukan jalannya suatu program. Worst case scenario ketika suatu program mengalami exception adalah termination. Termination (penutupan) program adalah hal yang harus dihindari. Untuk itu kita harus menangani exception yang terjadi di program, atau yang biasa disebut sebagai handle exception.

Syarat Kompetensi

1. Membuat, membuat, dan menjalankan aplikasi di Android Studio.
2. Merancang tata letak dengan tombol dan tampilan teks.
3. Menggunakan gaya dan tema.
4. Menyimpan dan memulihkan keadaan instance aktivitas.

Standar Kompetensi

1. Membuat file preferensi bersama untuk aplikasi Anda.
2. Menyimpan data ke preferensi bersama, dan membaca kembali preferensi itu.
3. Memperbarui aplikasi sehingga dapat menyimpan, mengambil, dan mengatur ulang preferensi bersama.

Dasar Teori

Terdapat tiga keyword pada Exception handling :

- try
- catch

- finally

Program akan menjalankan kode yang ada di dalam block try. Jika tidak ada kesalahan yang terjadi, maka program akan berjalan normal sampai baris kode di dalam try selesai. Jika ketika terjadi eror di dalam try, maka program akan keluar dari block try dan menjalankan block catch.

Tugas Pendahuluan

1. Sebutkan 3 jenis exception berdasarkan kategorinya.
2. Jelaskan mengenai ***finally , throw dan throws***
3. Buat salah satu nya No.2 dalam implementasi code.

Langkah-Langkah Praktikum

1.1 Buka sebuah method main

1. tuliskan code seperti berikut

```
public static void main(String[] args) {
    int[] a = new int[5];
    try {
        System.out.println("Akses elemen ke 5 :" + a[5]);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception thrown :" + e);
    } finally {
        a[4] = 10;
        System.out.println("Nilai elemen terakhir: " + a[4]);
    }

}
```

2. Pastikan output seperti berikut :

```
/usr/local/java/java-8-lama/bin/java ...
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 5
Nilai elemen terakhir: 10

Process finished with exit code 0
```

Sekarang analisis apa yang terjadi atau apa tugas masing masing try – catch dan finally.

Tugas Laporan

[Diberikan saat praktikum]

Lampiran-Lampiran

Format Tugas Pendahuluan

TUGAS PENDAHULUAN PRAKTIKUM 1

Nama :

Nim :

Soal :

Penyelesaian:

a. Algoritma Deskripsi (jika ada)

……

……

……

b. Flowchart (Jika ada)

……

……

……

c. Program

……

……

……

d. Penjelasan kode program

……

……

……

Tanda Tangan Asistensi

Asisten

Format Laporan Praktikum

LAPORAN PRAKTIKUM 1

Nama :

Nim :

Soal :

Penyelesaian:

a. Algoritma Deskripsi (jika ada)

……

……

……

b. Flowchart (Jika ada)

……

……

……

c. Program

……

……

……

d. Penjelasan kode program

……

……

……

Tanda Tangan Asistensi

Asisten