	Modul 6-7 **Modifier (Static, Final), Exception Handling, dan Polymorphism** Praktikum Pemrograman Berorientasi Objek Jurusan Teknik Informatika Fakultas Sains dan Teknologi
	Dosen: Rifqi Syamsul Fuadi, ST.

A. Dasar Teori

1. Static Member

Pendefinisian suatu member sebagai static member menghasilkan efek khusus, yaitu perubahan nilai yang dilakukan terhadap static member, akan dirasakan oleh semua instance dari class yang bersangkutan. Jadi static member tidak diasosiasikan dengan setiap instance, melainkan dengan class. Karena itu static member juga disebut sebagai class variable, sedangkan non-static member disebut instance variabel.

Contoh program (Static Member):

```
public class BorgUnite{
public static void main(String args[]){
Borg borg1 = new Borg();
Borg borg2 = new Borg();
System.out.println("Borg 1 intelligence = " +
borg1.getIntelligence());
System.out.println("Borg 2 intelligence = " +
borg2.getIntelligence());
borg1.learn();
System.out.println("Borg 1 intelligence = " +
borg1.getIntelligence());
System.out.println("Borg 2 intelligence = " +
borg2.getIntelligence());
Borg borg3 = new Borg();
System.out.println("Borg 3 intelligence = " +
borg3.getIntelligence());
}
}
class Borg{
private static int intelligence = 0; //Static member
public Borg() {
System.out.println("Spark...Bip...Bip..");
}
```

```
public void learn(){
int electricalSpark = (int)(Math.random()*5);
intelligence += electricalSpark;
System.out.println("Borg Intellgence increase by = " +
electricalSpark);
}
public int getIntelligence( ) {
return intelligence;
}
}
```

Contoh program (Non-Static Member):

```
public class HumanCantUnite{
public static void main(String args[]){
Human Human1 = new Human();
Human Human2 = new Human();
System.out.println("Human 1 IQ = " + Human1.getIQ());
System.out.println("Human 2 IQ = " + Human2.getIQ());
Human1.learn();
System.out.println("Human 1 IQ = " + Human1.getIQ());
System.out.println("Human 2 IQ = " + Human2.getIQ());
}
}
class Human{
private int IQ = 0;
public void learn(){
IQ += (int)(Math.random()*5);
}
public int getIQ(){
return IQ;
}
}
```

2. Static Method

Dengan mendefinisikan suatu field atau method sebagai static. Akses terhadap method tersebut dapat dilakukan tanpa melakukan instansiasi terlebih dahulu.

Cara memanggil method static yang dimiliki suatu class, yakni :
namaclass.namamethod([daftar-argument]);

Contoh Program (Static Method)

```
public class MyGeomUtil {
int iAmNotPopular = 13;
static int iAmCelebrity = 7;
public static double luasSegiempat(float length, float width) {
return length * width;
}
public static double luasSegitiga(float alas, float tinggi) {
return 0.5 * alas * tinggi;
}
public static double luasSegitiga(float A, float B, float gamma) {
// fungsi sin(double angel) menerima masukan sudut
// dalam radian angle not angel, you prevet...!
return 0.5 * A * B * Math.sin(gamma / Math.PI);
}
public static void main(String[] args){
double luassg = luasSegitiga(3, 5);
double luassg2 = luasSegitiga(3, 5, 2);
double luasse = MyGeomUtil.luasSegiempat(3,5);
System.out.println("Luas segitiga ke -1 = "+ luassg+" cm2");
System.out.println("Luas segitiga ke - 2 = "+ luassg2+" cm2");
System.out.println("Luas segiempat = "+ luasse+" cm2");
}
}
```

3. **Final**

Dengan menambahkan modifier final pada deklarasi sebuah method, dapat ditetapkan bahwa method pada class tersebut tidak bisa ditimpa/override pada subclass yang akan dibuat.

Modifier final pada method :

[modifier-modifier] final namamethod() {

//method body

}

Modifier final pada class :

final class namaclass extends parentclass {

// class body

}

Contoh Program (Final)

```
public class FinalKattWorld {
public static void main (String args[ ]) {
Katt k = new Katt();
k.speak();
k.berdoa();
Anggora a = new Anggora();
a.speak();
a.jump();
a.berdoa();
Siam s = new Siam();
s.speak();s.berdoa();
}
}

class Katt {
public Katt() {
System.out.println ("Katt Constructor");
}
public void speak() {
System.out.println ("Miaawww....");
}
public final void berdoa() {
```

```
System.out.println("Murr..Murr..Awright
...!!!");
}
}

class Anggora extends Katt {
public void jump() {
System.out.println
("Crash...BOOM...!!!"); }
//Perintah berikut akan menyebabkan
pesan kesalahan
public void berdoa() {
System.out.println("Murr...Awright...Am
ien...!!!");
}
}

final class Siam extends Katt {
public Siam() {
System.out.println ("Siam
Constructor");
}
public void speak() {
System.out.println
("Mmurrr...Murrrr....");
}
}
//Perintah berikut akan menyebabkan pesan
kesalahan
class Chesire extends Siam{
public Chesire() {
System.out.println("Chesire
Constructor");
}
}
```

4. Exception Handling

Eksepsi dapat diinisialisasikan secara otomatis oleh Java runtime atau secara manual melalui kode yang kita tulis. Oleh karena itu dikenal dengan istilah exception handling atau disebut juga dengan penanganan Eksepsi. Java menyediakan lima buah kata kunci untuk menangani eksepsi, yaitu: *try, catch, throw, throws,*dan *finally*.

Try - Catch

Kata kunci try digunakan untuk membuat blok yang berisi statemenstatemen yang mungkin menimbulkan eksepsi. Jika dalam proses eksekusi runtunan statemen tersebut terjadi sebuah eksepsi, maka eksepsi akan dilempar ke bagian blok penangkap yang dibuat dengan kata kunci catch.

```
class ContohMultiEksepsi {
public static void cobaEksepsi(int pembilang, int penyebut) {
try {
int hasil = pembilang / penyebut;
System.out.println("Hasil bagi: " + hasil);
int[] Arr = {1, 2, 3, 4, 5}; // array dengan 5 elemen
Arr[ 10] = 23; // mengakses indeks ke-10
}
catch (ArithmeticException eksepsi1) {
System.out.println("Terdapat pembagian dengan0");
}
catch (ArrayIndexOutOfBoundsException eksepsi2) {
System.out.println("Indeks di luar rentang");
}
}
public static void main(String[] args) {
cobaEksepsi(4, 0); // menimbulkan rithmeticException
System.out.println();
// menimbulkan ArrayIndexOutOfBoundsException
cobaEksepsi(12, 4);
}
}
```

5. **Polymorphism**

Polymorphism (polimorfi) bermakna sesuatu yang memiliki banyak bentuk. Dalam kehidupan sehari-hari dapat diilustrasikan sebagai berikut, contoh seperti kata “apel”, apel memiliki pengertian yang banyak. Ketika kita mengatakan “Ujang apel ke rumah pacarnya yang halamn rumahnya terdapat banyak pohon apel”. Contoh tersebut mengartikan bahwa dua kata apel tersebut memiliki banyak pengertian didalamnya. Namun istilah dalam dunia pemrograman, polymorphism diartikan sebagai suatu object dapat memiliki berbagai bentuk, sebagai object dari classnya sendiri atau dari object supercalssnya.

a. Overloading → Penggunaan satu nama untuk beberapa method yang berbeda parameter.
b. Overriding → Terjadi ketika deklarasi method subclass sama dengan method dari superclassnya.

Tulis prgram dibawah ini, harap teliti !!!

Contoh program: Polymorphism (Overloading)

```
class Mobil{
  String warna;
  int tahunproduksi

  public Mobil(String warna,int tahunproduksi){
    this.warna=warna;
    thistahunproduksi=tahunproduksi;
  }

  public Mobil(){
  }

  void info(){
    System.out.println("Warna: "+this.warna);
    System.out.println("Tahun: "+this.tahunproduksi);
  }
}

  class MobilKonstruktor{
    public static void main(String[]args){
      Mobil mobilku=new Mobil("Biru,2009);
      mobilku.info;

      Mobil mobilmu=new Mobil();
      mobilmu.info();

  }
```

Contoh program: Polymorphism (Overriding)

```
class Sepeda{
  int kecepatan=0;
  int gir=0

  void ubahGir(int pertambahanGir){
    gir=gir+pertambahanGir;
    System.outprintln("Gir: "+gir);
  }

  void tambahkecepatan(int pertambahankecepatan){
    kecepatan=kecepatan+pertambahankecepatan
    System.out.println("Kecepatan: "+kecepatan);
  }
}

class SepedaGunung extends Sepeda{

  void ubahGir(int pertambahanGir){
    gir=2*(gir+pertambahanGir);
    System.out.println("Gir: "+gir);
  }
}

class SepedaGunungAction{
  public void main(String]args){

    SepedaGunung sepedaku=new SepedaGunung();

    sepedaku.tambahkecepatan(10);
    sepedaku.ubahGir(1);
  }
}
```