MODUL PRAKTIKUM

Review Konsep Dasar Dalam JAVA

Versi 1.3

1. Tujuan

- Mengetahui dan menggunakan konsep dasar beorientasi object.
  - class
  - object
  - atribut
  - method
  - konstruktor
- Mengetahui dengan jelas tentang konsep lanjutan berorientasi object dan menggunakannya dengan baik
  - package
  - enkapsulasi
  - abstraksi
  - pewarisan
  - polimorfisme
  - interface
- Mengetahui dengan jelas penggunaan kata kunci *this, super, final* dan *static*
- Membedakan antara method *overloading* dan method *overridding*

2. Latar Belakang

Sebelum melangkah pada fitur-fitur menarik yang ada pada Java, mari kita meninjau beberapa hal yang telah Anda pelajari pada pelajaran pemograman pertama Anda. Pelajaran ini menyajikan diskusi tentang perbedaan konsep-konsep berorientasi object dalam Java.

Desain berorientasi object adalah sebuah teknik yang memfokuskan desain pada object dan class berdasarkan pada skenario dunia nyata. Hal ini menegaskan keadaan(*state*), *behaviour* dan interaksi dari object. Selain itu juga menyediakan manfaat akan kebebasan pengembangan, meningkatkan kualitas, mempermudah pemeliharaan, mempertinggi kemampuan dalam modifikasi dan meningkatkan penggunaan kembali software.

3. Percobaan

Percobaan 1 Class SuperHero:

```
public class SuperHero {
    String superPower[];
    void setSuperPower(String superPower[]){
        this.superPower = superPower;
    }
    void printSuperPower(){
        for(int i=0;i<superPower.length;i++){
            System.out.println(superPower[i]);
        }
    }
}
```

Percobaan 2 Class Atribut Demo :

```
public class StudentRecordExample {
    public static void main( String[] args ){
    //membuat 3 object StudentRecord
    StudentRecord annaRecord = new StudentRecord();
    StudentRecord beahRecord = new StudentRecord();
    StudentRecord crisRecord = new StudentRecord();
    //Memberi nama siswa
    annaRecord.setName("Anna");
    beahRecord.setName("Beah");
    crisRecord.setName("Cris");
    //Menampilkan nama siswa "Anna"
    System.out.println( annaRecord.getName() );
    //Menampilkan jumlah siswa
    System.out.println("Count="+StudentRecord.getStudentCount());
    }
}
```

Percobaan 3 Class method Demo:

```
public class MethodDemo {
    int data;
    int getData(){
        return data;
    }
    void setData(int data){
        this.data = data;
    }
    void setMaxData(int data1,int data2){
        data = (data1>data2)? data1 : data2;
    }
}
```

Percobaan 4 Class Construstor Demo :

```
public class ConstructorDemo {
    private int data;
    public ConstructorDemo(){
        data = 100;
    }
    ConstructorDemo(int data){
        this.data = data;
    }
}
```

Percobaan 5 Instantiate sebuah Class:

```
public class ConstructObj {
    int data;
    ConstructObj(){}
    void setData(int data){
        this.data = data;
    }
    public static void main(String[] args) {
            ConstructObj obj = new ConstructObj();
    }
}
```

Percobaan 6 Mengakses sebuah object :

```
public class ConstructObj {
    int data;
    ConstructObj(){}
    void setData(int data){
        this.data = data;
    }
    public static void main(String[] args) {
        ConstructObj obj = new ConstructObj();
        obj.setData(10);
        System.out.println(obj.data);
    }
}
```

Percobaan 7 Package :

```
package registration.reports;

import registration.processing.*;
import java.util.List;
import java.lang.*;

class MyClass {
/*rincian dari MyClass*/
}
```

Percobaan 8 Class Enkapsulasi :

```
public class encapsulation {
    private int secret;
    public boolean setSecret(int secret){
        if(secret <1 || secret >100){
            return false;
        }
        this.secret = secret;
        return true;
    }
    public int getSecret(){
        return secret;
    }
}
```

Percobaan 9 Class Override Demo :

```
class Superclass {
    void display(int n){
        System.out.println("super : "+n);
    }
}
class Subclass extends Superclass {
    void display(int k){
        System.out.println("sub : "+k);
    }
}
public class OverrideDemo {
    public static void main(String[] args) {
        Subclass subObj = new Subclass();
        Superclass SuperObj = subObj;
        subObj.display(3);
        ((Superclass)subObj).display(4);
    }
}
```

Percobaan 10 Class Abstract dan method :

```
abstract class SuperHero {
    String superPower[];
    void setSuperPower(String superPower[]){
        this.superPower = superPower;
    }
    void printSuperPower(){
        for(int i=0;i<superPower.length;i++){
            System.out.println(superPower[i]);
        }
    }
    abstract void displayPower();
}
public class UnderwaterSuperHero extends SuperHero {
    void displayPower(){
        System.out.println("Communicate with sea creatures...");
        System.out.println("Fast swimming ability...");
    }
}
class FlyingSuperHero extends SuperHero {
    void displayPower(){
        System.out.println("Fly...");
    }
}
```

Percobaan 11 Class Interface Demo :

```
interface MyInterface {
    void iMethod();
}
class Myclass1 implements MyInterface{
    public void iMethod(){
        System.out.println("Interface Method.");
    }
    void MyMethod(){
        System.out.println("Another Method");
    }
}
class Myclass2 implements MyInterface{
    public void iMethod(){
        System.out.println("Another implementasion");
    }
}
public class interfaceDemo {
    public static void main(String[] args) {
        Myclass1 mc1 = new Myclass1();
        Myclass2 mc2 = new Myclass2();
        mc1.iMethod();
        mc1.MyMethod();
        mc2.iMethod();
    }
}
```

Percobaan 12 Kata Kunci This :

```
public class thisDemo1 {
    int data;
    void method(int data){
        this.data = data;
    /*
    * this. data menunjuk ke atribut dan data menunjuk ke variabel lokal
    */
    }
}
public class thisDemo2 {
    int data;
    void method(){
        System.out.println(data); //this.data
    }
    void method2(){
        method(); //this.method()
    }
}
public class thisDemo3 {
    int data;
    thisDemo3(int data){
        this.data = data;
    }
}
```

```
class Person{
    String firstName;
    String lastName;
    Person(String fName,String lName){
        firstName = fName;
        lastName = lName;
    }
}
public class student extends Person {
    String studNum;
    student(String fName,String lName,String sNum){
        super(fName,lName);
        studNum = sNum;
    }
}
```

Percobaan 13 Kata Kunci Super :

```
Contoh lain : class SuperDemo

class SuperClass{
    int a;
    void display_a(){
        System.out.println("a = "+a);
    }
}
class ZubClass extends SuperClass{
    int a;
    void display_a(){
        System.out.println("a = "+a);
    }
    void set_super_a(int n){
        super.a = n;
    }
    void display_super_a(){
        super.display_a();
    }
}
public class SuperDemo {
    public static void main(String[] args) {
        SuperClass superObj = new SuperClass();
        SubClass SubObj = new SubClass();
        superObj.a = 1;
        SubObj.a = 2;
        SubObj.set_super_a(3);
        superObj.display_a();
        SubObj.display_a();
        SubObj.display_super_a();
        System.out.print(SubObj.a);
    }
}
```

Percobaan 14 Kata Kunci Static :

```
class Demo {
    static int a =0;
    static void staticMethod(int i){
        System.out.println(i);
    }
    static {
        System.out.println("This is a block static");
        a+=1;
    }
}
public class StaticDemo {
    public static void main(String[] args) {
        System.out.println(Demo.a);
        Demo.staticMethod(5);
        Demo d = new Demo();
        System.out.println(d.a);
        d.staticMethod(0);
        Demo e = new Demo();
        System.out.println(e.a);
        d.a +=3;
        System.out.println(Demo.a+", "+d.a+", "+e.a);
    }
}
```

Percobaan 15 Outer Class :

```
public class OuterClass {
    int data = 5;
    class InnerClass{
        int data2 = 10;
        void method(){
            System.out.println(data);
            System.out.println(data2);
        }
    }
    public static void main(String[] args) {
        OuterClass oc = new OuterClass();
        InnerClass ic = oc.new InnerClass();
        System.out.println(oc.data);
        System.out.println(ic.data2);
        ic.method();
    }
}
```

4. Latihan

4.1 Tabel Perkalian

Tulis program yang mempunyai masukkan *size* dari user dan mencetak tabel perkalian dengan *size* yang ditetapkan.
Size untuk tabel perkalian : 5
Tabel perkalian dari size 5:

	1	2	3	4	5
1	1				
2	2	4			
3	3	6	9		
4	4	8	12	16	
5	5	10	15	20	25

4.2 Greatest Common Factor(GCF)

Tulis sebuah program yang mempunyai tiga integer dan menghitung nilai GCF dari tiga angka. GCF adalah angka terbesar yang secara rata dibagi ke semua angka yang diberikan.

Input 1: 25	Input 1: 1	Input 1: 9
Input 2: 15	Input 2: 2	Input 2: 27
Input 3: 35	Input 3: 3	Input 3: 12
GCF: 5	GCF: 1	GCF: 3

4.3 Shape

Buatlah class S*hape*. class memiliki dua field *String*: *name* dan *size.* class mempunyai method *printShapeInfo*, dimana hanya mengeluarkan nilai *name* dan field *size* dari object *Shape.* Juga memiliki method *printShapeName* dan *printShapeSize,* dimana mencetak nama dan size dari object, berturut-turut.

Menggunakan pewarisan, buat class *Square* dengan field yang sama dan method seperti itu dari class *Shape*. Class ini mempunyai dua tambahan field integer: *length* dan *width*. Method *printShapeLength* dan *printShapeWidth* yang mencetak panjang dan lebar object yang juga termasuk dalam class ini. Anda juga harus meng-override printShapeInfo untuk mencetak keluaran field tambahan dalam subclass juga.

4.4 Binatang

Buatlah interface *Animal* yang mempunyai tiga method:*eat* dan *move*. Semua method ini tidak punya argumen atau nilai return. Method ini hanya mengeluarkan bagaimana object *Animal* makan dan bergerak. Sebagai contoh, seekor kelinci memakan wortel dan bergerak dengan melompat. Buat class *Fish* dan *Bear* yang menggunakan interface *Animal*. Terserah kepada Anda bagaimana menggunakan method *eat* dan *move.*