PRAKTIKUM OOP

Polymorphism

Latihan 1: Inheritance dan Overriding method

```
class Karyawan{

    // instance variabel
    private String nama;
    private double gaji;

    // class variabel
    private static double persennaikgaji=0.10;

    Karyawan(String nm, double gj){
        this.setNama(nm);
        this.setGaji(gj);
    }

    void setNama(String nm){
        nama = nm;
    }

    void setGaji(double gj){
        gaji = gj;
    }

    static void setPersentase(double persen){
        persennaikgaji = persen;
    }

    String getNama(){
        return nama;
    }
    double getGaji(){
        return gaji;
    }

    static double getPersentase(){
        return persennaikgaji;
    }

    void naikkanGaji(){
        gaji+=(gaji*persennaikgaji);
    }
}
```

Karyawan.java

```
public class Manager extends karyawan{
    private static double bonus = 500;

    Manager(String nm, double gj){
        super(nm,gj);
    }

    double getBonus(){
        return bonus;
    }

    void setBonus(double bns){
        bonus = bns;
    }

    double getGaji()
    {
        double gajidasar = super.getGaji();
        return (gajidasar+bonus);
    }
}
```

Manager.java

Fungsi getGaji pada kelas Manager
Akan mengoverride fungsi getGaji
Yg dimiliki oleh kelas parent yaitu
Kelas karyawan

TesManager.java

```
public class TesManager{
    public static void main(String args[]){
        Manager mng = new Manager("Andi",300);

        System.out.println(mng.getNama());
        System.out.println(mng.getBonus());
        System.out.println(mng.getGaji());
    }
}
```

Latihan 2 :Polymorphism dan constructor overriding. Pemanggilan parent constructor.

Employee.java

```
class Employee{
    String nama;
    Employee(){
        System.out.println("Konstruktor Employee()dijalankan");
    }
    Employee(String n){
        this.nama = n;
        System.out.println("Konstruktor Employee(String n) dijalankan");
    }
    void getDetails(){
        System.out.println(nama);
    }
}
```

Manager2.java

```
class Manager2 extends Employee{
    String department;
    Manager2(){
        this("sales");  // memanggil konstruktor parent dengan satu argumen
        System.out.println("Konstruktor Manager() dijalankan");
    }
    Manager2(String dept){
        department = dept;
        System.out.println("Konstruktor Manager(String dept) dijalankan");
    }
    void getDetails(){
        System.out.println(department);
    }
    public static void main(String args[]){
        Manager2 e = new Manager2();
        e.getDetails();
    }
}
```

- Jelaskan proses pemanggilan konstruktor pada latihan 2

Latihan 3 : Virtual Method Invocation dan Polymorphic Arguments.

Employee.java

```
public class Employee{
    String nama;
    int Salary;

    Employee(){
    }

    Employee(String nm){
        this.nama = nm;
        System.out.println("Employee");
    }

    public int salary(){
            return 0;
    }
}
```

```
class Manager extends Employee{
    private static final int mgrSal = 40000;
    private static final int tunjangan = 40000;

    public int salary(){
            return mgrSal;
    }

    public int tunjangan(){
            return tunjangan;
    }
}
```

```
class Programmer extends Employee{
    private static final int prgSal = 50000;
    private static final int prgBonus = 10000;

    public int salary (){
        return prgSal;
    }

    public int bonus (){
        return prgBonus;
    }
}
```

```
class Payroll{
    public int calcPayroll(Employee emp){
        int money = emp.salary();
        if (emp instanceof Manager)
           money += ((Manager) emp).tunjangan();
        if (emp instanceof Programmer)
           money += ((Programmer) emp).bonus();
        return money;
    }

    public static void main(String [] args){
        Payroll pr = new Payroll();
        Programmer prg = new Programmer();
        Manager mgr = new Manager();
        System.out.println("Payroll untuk Programmer : "+ pr.calcPayroll(prg));
        System.out.println("Payroll untuk Manager : "+ pr.calcPayroll(mgr));
    }
}
```

Latihan 4 : Polymorphism - Method overloading

```
import java.awt.Point;

class MyRect {
    int x1 = 0;
    int y1 = 0;
    int x2 = 0;
    int y2 = 0;

    MyRect buildRect(int x1, int y1, int x2, int y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
        return this;
    }
    MyRect buildRect(Point topLeft, Point bottomRight) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = bottomRight.x;
        y2 = bottomRight.y;
        return this;
   }

   MyRect buildRect(Point topLeft, int w, int h) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = (x1 + w);
        y2 = (y1 + h);
        return this;
   }
```

```
    void printRect(){
         System.out.print("MyRect: <" + x1 + ", " + y1);
         System.out.println(", " + x2 + ", " + y2 + ">");
    }

    public static void main(String[] arguments) {
         MyRect rect = new MyRect();
         System.out.println("Calling buildRect with coordinates 25,25, 50,50:");
         rect.buildRect(25, 25, 50, 50);
         rect.printRect();
         System.out.println("***");
         System.out.println("Calling buildRect with points (10,10), (20,20):");
         rect.buildRect(new Point(10,10), new Point(20,20));
         rect.printRect();
         System.out.println("***");
         System.out.print("Calling buildRect with 1 point (10,10),");
         System.out.println(" width (50) and height (50):");
         rect.buildRect(new Point(10,10), 50, 50);
         rect.printRect();
         System.out.println("***");
     }
}
```

Latihan 5 : Polymorphism - Upcasting

- Contoh upcasting yaitu mengkonversi suatu kelas turunan ke kelas parent, simpan semua program latihan 5 dalam file Music.java

```
class Note {
    private int value;
    private Note(int val) { value = val; }

    public static final Note
        middleC = new Note(0),
        cSharp = new Note(1),
        cFlat = new Note(2);
} // Etc.

class Instrument {
    public void play(Note n) {
        System.out.println("Instrument.play()");
    }
}
```

```
// Wind objects are instruments
// because they have the same interface:
class Wind extends Instrument {
    // Redefine interface method:
    public void play(Note n) {
        System.out.println("Wind.play()");
    }
}

public class Music {
    public static void tune(Instrument i) {
        // ...
        i.play(Note.middleC);
    }

    public static void main(String[] args) {
        Wind flute = new Wind();
        tune(flute); // Upcasting
    }
} ///:~
```