Praktikum

Nama : Siti Alfiyyatuz Zakiyyah Alawiyyah
NIM : 20220040136
Kelas : TI 22 A

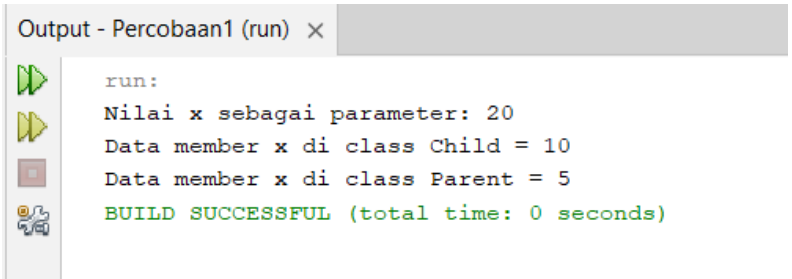
Percobaan 1

```
class Child extends Parent {
    public int x = 10;
    public void Info(int x){
        System.out.println("Nilai x sebagai parameter: " + x);
        System.out.println("Data member x di class Child = " + this.x);
        System.out.println("Data member x di class Parent = " + super.x);
    }
}

public class NilaiX {
    public static void main(String[] args) throws Exception {
        Child test = new Child();
        test.Info(x: 20);
    }
}
```

- Ada dua kelas yang dideklarasikan, yaitu **Parent** dan **Child**.
- Kelas **Child** merupakan turunan dari kelas **Parent**.
- Kelas **Parent** memiliki sebuah variabel anggota **x** dengan nilai 5.
- Kelas **Child** memiliki variabel anggota **x** yang menutupi variabel **x** dari kelas **Parent**, dengannilai 10.
- Kelas **Child** memiliki metode **Info()** yang menerima satu parameter **x**.
- Metode **Info()** mencetak nilai parameter **x**, nilai variabel **x** di kelas **Child**, dan nilai variabel **x**

  di kelas **Parent** menggunakan kata kunci super.
- Pada metode **main(),** sebuah objek **Child** (**test**) dibuat.
- Metode **Info()** dipanggil pada objek test

dengan argumen 20Output



Percobaan 2

```
class Pegawai {
    public String nama;
    public double gaji;
}

public class Manajer extends Pegawai{
  public String departemen;

  public void isiData(String n, String d){
    nama = n;
    departemen = d;
  }
}
```

- Kelas **Pegawai** memiliki dua variabel anggota (**nama** dan **gaji**) yang bersifat publik (public),yang berarti dapat diakses dari luar kelas.
- Kelas **Manajer** merupakan turunan dari kelas **Pegawai**.
- Kelas **Manajer** memiliki satu variabel anggota tambahan yaitu **departemen**, yangmenunjukkan departemen tempat manajer bekerja.
- Kelas **Manajer** memiliki metode **isiData()** yang menerima dua parameter, yaitu **n** (nama) dan **d** (departemen).
- Metode **isiData()** mengisi nilai variabel **nama** dan **departemen** dari objek **Manajer**.

Error :

- Setiap public class harus ditulis pada filenya masing-masing.
- atribut nama yang dituliskan di kelas pegawai tidak dapat diakses pada kelas lain, sehinggaterjadi error pada baris 10 karena atribut nama tidak dapat diakses dari kelas turunannya sekalipun, dalam hal ini kelas Manajer

Cara Memperbaiki :

- Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada classPegawai.
- Mengganti modifier atribut nama menjadi public

Percobaan 3

```
class Parent {
    // kosong
}

public class Child extends Parent {
    int x;
    public Child(){
        x = 5;
    }
}
```

- Kelas Parent adalah kelas dasar atau induk yang tidak memiliki anggota data atau metode apapun yang didefinisikan di dalamnya.
- Kelas Child adalah subkelas dari Parent, yang berarti Child mewarisi semua anggota danmetode yang ada dalam Parent.
- Kelas Child memiliki satu anggota data x yang tidak dideklarasikan secara eksplisit, sehinggamemiliki akses pakai default (default access modifier), yang berarti dapat diakses dari kelas dalam paket yang sama.
- Kelas Child memiliki sebuah konstruktor tanpa parameter yang menginisialisasi nilai xdengan 5.

Error:

- Setiap public class harus ditulis pada filenya masing-masing.

Cara Memperbaiki :

- Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada classParent.

Percobaan 4

```
import java.util.Date;

class Employee {
    private static final double BASE_SALARY = 15000.00;
    private String Name = "";
    private double Salary = 0.0;
    private Date birthDate;

    public Employee(){}
    public Employee(String name, double salary, Date DoB){
        this.Name = name;
        this.Salary = salary;
        this.birthDate = DoB;
    }
    public Employee(String name, double salary){
        this(name, salary, null);
    }
    public Employee(String name, Date DoB){
        this(name, BASE_SALARY, DoB);
    }
    public Employee(String name){
        this(name, BASE_SALARY);
    }

    public String getName() {
      return Name;
    }

    public double getSalary() {
      return Salary;
    }
}

class Manager extends Employee{
    // tambahan attribute untuk kelas manager;
    private String department;

    public Manager(String name, double salary, String dept){
        super(name, salary);
        department = dept;
    }

    public Manager(String n, String dept){
        super(n);
        department = dept;
    }

    public Manager(String dept){
        super();
        department = dept;
    }

    public String getDept() {
      return department;
    }
}

public class TestManager {
    public static void main(String[] args) throws Exception {
        Manager Utama = new Manager(name:"John", salary:5 000000, dept:"Financial");
        System.out.println("Name:" + Utama.getName());
        System.out.println("Salary:" + Utama.getSalary());
        System.out.println("Department:" + Utama.getDept());

        Utama = new Manager(n:"Michael", dept:"Accouting");
        System.out.println("Name:" + Utama.getName());
        System.out.println("Salary:" + Utama.getSalary());
        System.out.println("Department:" + Utama.getDept());
    }
}
```

- Memiliki dua kelas utama, **Employee** dan **Manager**, yang merepresentasikan karyawan dan manajer. **Manager** adalah subclassdari **Employee**. Ada beberapa konstruktor di kedua kelas untuk inisialisasi objek dengan berbagai argumen.

Error:

Harus menambahkan import java.util.Date pada awal baris kode

Output

```
Output - Percobaan4 (run)
    run:
    Name:John
    Salary:5000000.0
    Department:Financial
    Name:Michael
    Salary:15000.0
    Department:Accouting
    BUILD SUCCESSFUL (total time: 0 seconds)
```

Percobaan 5

```
class MoodyObject {
 protected String getMood(){
   return "moody";
 }

 public void speak(){
   System.out.println("I am" + getMood());
 }

 void laugh(){}
 void cry(){}
}

class SadObject extends MoodyObject {
 protected String getMood(){
   return "sad";
 }
 void cry() {
   System.out.println(x:"Hoo hoo");
 }
}

class HappyObject extends MoodyObject {
 protected String getMood() {
   return "happy";
 }
 void laugh() {
   System.out.println(x:"Hahaha");
 }
}

public class MoodyTest {
    public static void main(String[] args) {
      MoodyObject m = new MoodyObject();

      // test parent class
      m.speak();

      // test inheritance class
      m = new HappyObject();
      m.speak();
      m.laugh();

      // test inheritance class
      m = new SadObject();
      m.speak();
      m.cry();
    }
}
```

- **MoodyObject**: Kelas ini adalah kelas dasar yang memiliki metode **speak**, **laugh**, dan **cry**.Metode **getMood** yang dilindungi (protected) mengembalikan string "moody".
- **SadObject**: Kelas ini mewarisi **MoodyObject** dan meng-override metode **getMood** untukmengembalikan string "sad". Ini juga memiliki metode **cry**, yang mencetak "Hoo hoo".
- **HappyObject**: Kelas ini juga mewarisi **MoodyObject** dan meng-override metode **getMood**

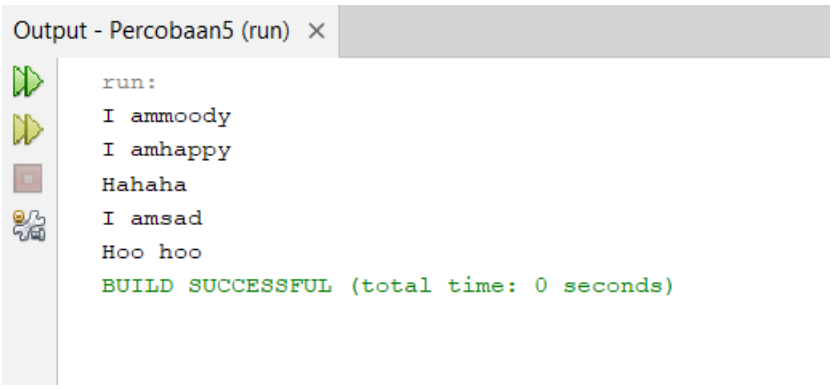
  untuk mengembalikan string "happy". Ini memiliki metode **laugh**, yang mencetak "Hahaha".

Error:

- Setiap public class harus ditulis pada filenya masing-masing.

Cara memperbaiki:

- Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada classMoodyObject, SadObject, dan HappyObject.



Percobaan 6

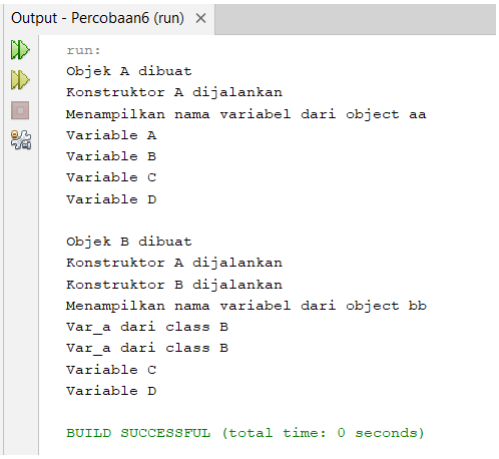
```
class A {
  String var_a = "Variable A";
  String var_b = "Variable B";
  String var_c = "Variable C";
  String var_d = "Variable D";

  A(){
    System.out.println(x: "Konstruktor A dijalankan");
  }
}
```

```
class B extends A {
    B(){
        System.out.println(x: "Konstruktor B dijalankan");
        var_a = "Var_a dari class B";
        var_b = "Var_a dari class B";
    }

    public static void main(String[] args) throws Exception {
        System.out.println(x: "Objek A dibuat");
        A aa = new A();
        System.out.println(x: "Menampilkan nama variabel dari object aa");
        System.out.println(x: aa.var_a);
        System.out.println(x: aa.var_b);
        System.out.println(x: aa.var_c);
        System.out.println(x: aa.var_d);
        System.out.println(x: "");

        System.out.println(x: "Objek B dibuat");
        B bb = new B();
        System.out.println(x: "Menampilkan nama variabel dari object bb");
        System.out.println(x: bb.var_a);
        System.out.println(x: bb.var_b);
        System.out.println(x: bb.var_c);
        System.out.println(x: bb.var_d);
        System.out.println(x: "");

    }
}
```

- Kelas A memiliki empat anggota data bertipe string: var_a, var_b, var_c, dan var_d, yang diinisialisasi dengan nilai string tertentu. Kelas A juga memiliki konstruktor yang mencetak pesan"Konstruktor A dijalankan" ketika dipanggil.
- Kelas B adalah subkelas dari A yang mewarisi semua anggota data dan metode dari kelas A. Kelas B memiliki konstruktor sendiri yang mencetak pesan "Konstruktor B dijalankan". Di dalamkonstruktor B, nilai dari var_a dan var_b diubah.
- Metode main terdapat di dalam kelas B dan digunakan untuk menguji pembuatan objek darikedua kelas A dan B, serta untuk mencetak nilai anggota data dari objek-objek tersebut.
- Ketika objek A (objek aa) dibuat, hanya konstruktor kelas A yang dijalankan. Nilai anggota datavar_a, var_b, var_c, dan var_d dari objek tersebut tetap sesuai dengan nilai awal yang diinisialisasi di dalam kelas A.
- Ketika objek B (objek bb) dibuat, terlebih dahulu konstruktor kelas A dijalankan, kemudian konstruktor kelas B dijalankan. Dalam konstruktor

kelas B, nilai dari var_a dan var_b diubah. Oleh karena itu, saat mencetak nilai anggota data dari objek bb, nilai var_a dan var_b mengikuti

perubahan yang dilakukan dalam konstruktor kelas B, sedangkan nilai var_c dan var_d tetapsesuai dengan nilai awal yang diinisialisasi di dalam kelas A.



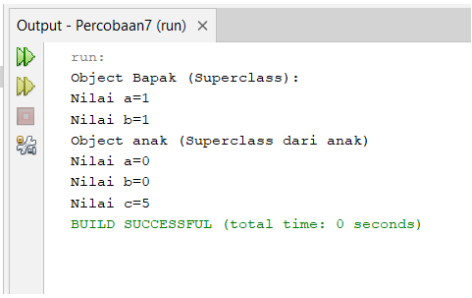
Percobaan 7

```
class Bapak {
    int a;
    int b;

    void show_variabel(){
        System.out.println("Nilai a=" + a);
        System.out.println("Nilai b=" + b);
    }
}

class Anak extends Bapak {
    int c;

    void show_variabel() {
        super.show_variabel();
        System.out.println("Nilai c=" + c);
    }
}

public class InheritExample {
    public static void main(String[] args) throws Exception {
        Bapak objectBapak = new Bapak();
        Anak objectAnak = new Anak();

        objectBapak.a = 1;
        objectBapak.b = 1;
        System.out.println("Object Bapak (Superclass):");

        objectBapak.show_variabel();
        objectAnak.c = 5;
        System.out.println("Object anak (Superclass dari anak)");
        objectAnak.show_variabel();
    }
}
```

Kelas **Bapak** memiliki variabel **a** dan **b**, serta metode **show_variabel()** untuk mencetak nilai **a** dan **b**.

Kelas **Anak** mewarisi variabel dan metode dari kelas **Bapak**, dengan menambahkan variabel **c**. Metode **show_variabel()** juga dioverride untuk mencetak nilai **a**, **b**, dan **c**.

Kelas **InheritExample** membuat objek dari kelas **Bapak** dan **Anak**, menginisialisasi nilai, dan memanggil metode.


Output - Percobaan7 (run)
run:
Object Bapak (Superclass):
Nilai a=1
Nilai b=1
Object anak (Superclass dari anak)
Nilai a=0
Nilai b=0
Nilai c=5
BUILD SUCCESSFUL (total time: 0 seconds)

Percobaan 8

```java
public class Parent {
    String parentName;
    Parent(){}

    Parent(String parentName){
        this.parentName = parentName;
        System.out.println(x: "Konstruktor parent");
    }
}

class Baby extends Parent {
    String babyName;

    Baby(String babyName){
        super();
        this.babyName = babyName;
        System.out.println(x: "Konstruktor Baby");
        System.out.println(x: babyName);
    }

    public void Cry(){
        System.out.println(x: "Owek owek");
    }
}
```

Kelas **Baby** mewarisi dari kelas **Parent** dan menambahkan atribut **babyName** serta metode **Cry()**. Saat objek dari kelas **Baby** dibuat, konstruktor kelas **Parent** dan **Baby** dipanggil secara berurutan. Metode **Cry()** dapat dipanggil untuk mencetak "Owek owek" ke konsol.