Nama :Hlda Deliana
Kelas : TI22H
NIM : 20220040112

Percobaan 1:

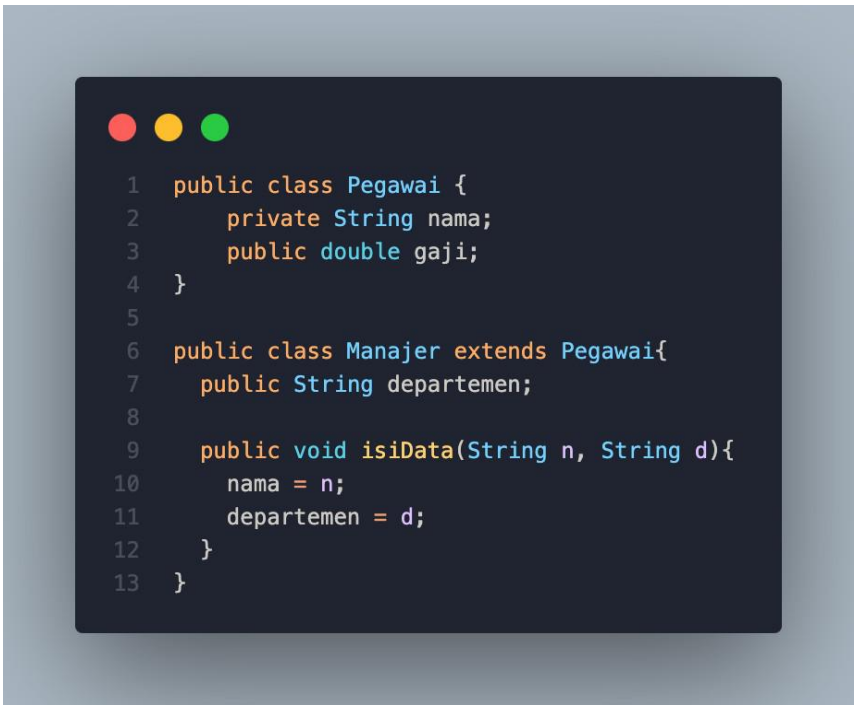
```java
class Parent {
    public int x = 5;
}

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(20);
    }
}
```

- Kelas Parent memiliki satu anggota data publik x yang diinisialisasi dengan nilai 5.
- Kelas Child mewarisi kelas Parent, yang berarti Child memiliki semua anggota data dan metode yang dimiliki oleh Parent.
- Kelas Child menambahkan anggota data x baru yang diinisialisasi dengan nilai 10.
- Kelas Child juga memiliki metode Info yang menerima satu parameter x.
- Metode Info mencetak nilai x yang diterima sebagai parameter, nilai x dari instans Child menggunakan this.x, dan nilai x dari kelas Parent menggunakan super.x.
- Dalam metode main dari kelas NilaiX, sebuah objek test dari kelas Child dibuat dan metode Info dipanggil dengan nilai 20 sebagai argumen.

Hasil output yang diharapkan dari kode tersebut adalah:

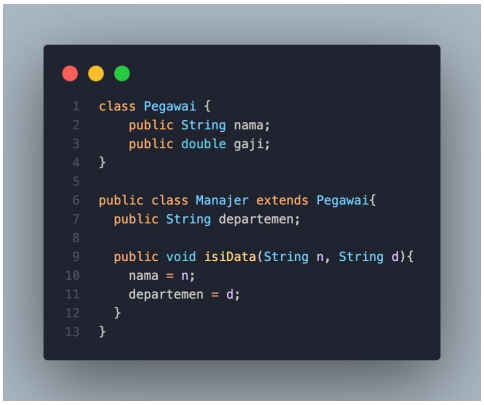
```
Nilai x sebagai parameter: 20
Data member x di class Child = 10
Data member x di class Parent = 5
```

Percobaan 2:



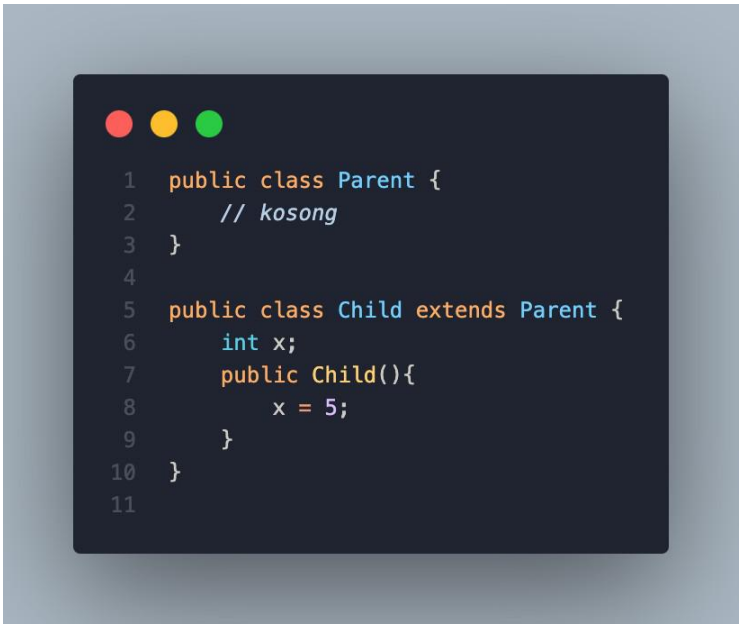
```
1  public class Pegawai {
2      private String nama;
3      public double gaji;
4  }
5
6  public class Manajer extends Pegawai{
7    public String departemen;
8
9    public void isiData(String n, String d){
10     nama = n;
11     departemen = d;
12   }
13 }
```

- Kelas Pegawai memiliki dua anggota data: nama, yang bersifat private dan hanya dapat diakses dalam kelas itu sendiri, dan gaji, yang bersifat public dan dapat diakses dari mana saja di luar kelas Pegawai.
- Kelas Manajer merupakan subkelas dari Pegawai, yang berarti Manajer mewarisi semua anggota data dan metode yang dimiliki oleh Pegawai.
- Kelas Manajer menambahkan satu anggota data baru yaitu departemen.
- Kelas Manajer memiliki sebuah metode isiData yang menerima dua parameter string n dan d, yang digunakan untuk mengatur nilai nama dan departemen.
- Kode diatas terjadi error, karena dua hal
    - Setiap public class harus ditulis pada filenya masing-masing.
    - atribut nama yang dituliskan di kelas pegawai tidak dapat diakses pada kelas lain, sehingga terjadi error pada baris 10 karena atribut nama tidak dapat diakses dari kelas turunannya sekalipun, dalam hal ini kelas Manajer
- Cara memperbaikinya adalah:
    - Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada class Pegawai.
    - Mengganti modifier atribut nama menjadi public



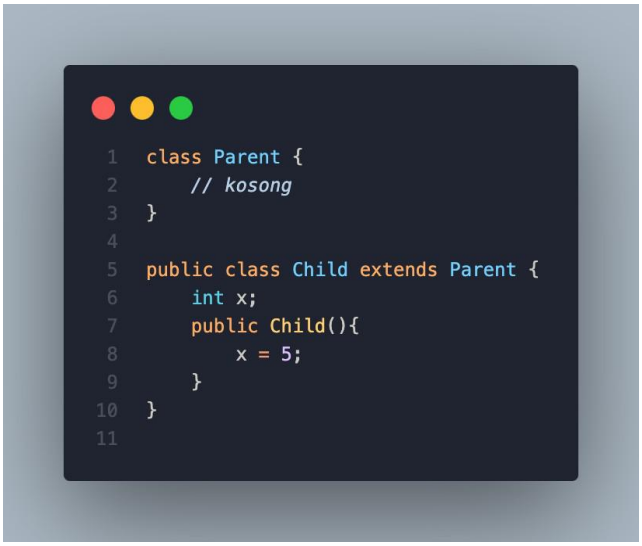
```
1  class Pegawai {
2      public String nama;
3      public double gaji;
4  }
5
6  public class Manajer extends Pegawai{
7    public String departemen;
8
9    public void isiData(String n, String d){
10     nama = n;
11     departemen = d;
12   }
13 }
```

Percobaan 3:



```java
public 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 apa pun yang didefinisikan di dalamnya.
- Kelas Child adalah subkelas dari Parent, yang berarti Child mewarisi semua anggota dan metode yang ada dalam Parent.
- Kelas Child memiliki satu anggota data x yang tidak dideklarasikan secara eksplisit, sehingga memiliki 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 x dengan 5.
- Kode diatas terjadi error, karena satu hal
  - Setiap public class harus ditulis pada filenya masing-masing.
- Cara memperbaikinya adalah:
  - Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada class Parent.
- Tidak ada error yang berkaitan dengan konstruktor



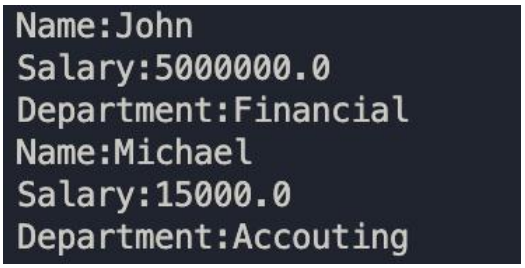
```java
class Parent {
    // kosong
}

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

Percobaan 4:

```java
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("John", 5000000, "Financial");
        System.out.println("Name:" + Utama.getName());
        System.out.println("Salary:" + Utama.getSalary());
        System.out.println("Department:" + Utama.getDept());

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

- Kelas Employee mendefinisikan beberapa atribut, termasuk Name (nama karyawan), Salary (gaji karyawan), dan birthDate (tanggal lahir karyawan). Atribut BASE_SALARY adalah konstanta yang merupakan gaji dasar yang diberikan kepada karyawan.
- Kelas Employee memiliki beberapa konstruktor yang memungkinkan objek Employee dibuat dengan berbagai kombinasi parameter, termasuk nama, gaji, dan tanggal lahir. Jika parameter tanggal lahir tidak disediakan, maka akan dianggap null.

- Kelas Manager adalah subkelas dari Employee yang menambahkan atribut department (departemen) dan memiliki konstruktor yang memungkinkan objek Manager dibuat dengan berbagai kombinasi parameter, termasuk nama, gaji, dan departemen.
- Metode getDept() digunakan untuk mendapatkan nilai departemen dari objek Manager.
- Kelas TestManager memiliki metode main yang membuat objek Manager dengan berbagai kombinasi parameter dan mencetak nama, gaji, dan departemen dari setiap objek yang dibuat.
- Dalam metode main, terdapat dua objek Manager yang dibuat.
    - Pertama, objek Utama dengan nama "John", gaji 5000000, dan departemen "Financial".
    - Kedua, objek Utama dengan nama "Michael" dan departemen "Accounting". Gaji objek Utama kedua akan menggunakan nilai BASE_SALARY karena gaji tidak disediakan dalam konstruktor yang digunakan.
- Hanya terdapat satu error pada kode tersebut, yaitu harus menambahkan import java.util.Date pada awal baris kode.
- Jika sudah diperbaiki, maka output dari kode tersebut adalah:

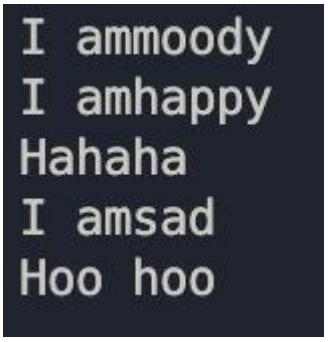


Percobaan 5:

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

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

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

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

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

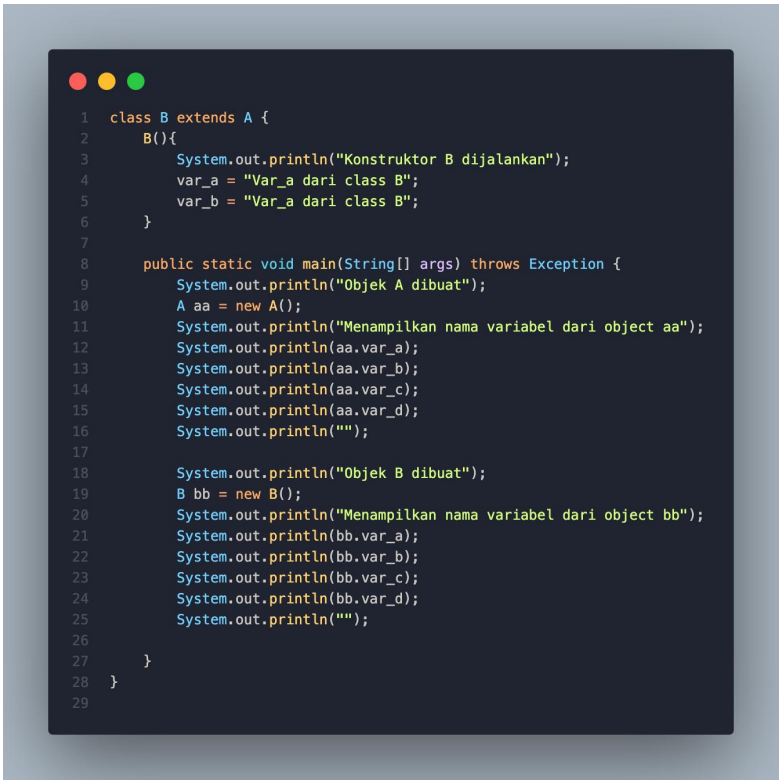
- Kelas MoodyObject adalah kelas dasar yang memiliki metode getMood() yang mengembalikan string "moody" dan metode speak() yang mencetak "I am" diikuti oleh mood yang diperoleh dari metode getMood(). Kelas ini juga memiliki dua metode tanpa pengimplementasian, yaitu laugh() dan cry().

- Kelas SadObject adalah subkelas dari MoodyObject yang mengganti implementasi metode getMood() sehingga mengembalikan string "sad" dan mengimplementasikan metode cry() untuk mencetak "Hoo hoo".
- Kelas HappyObject juga adalah subkelas dari MoodyObject yang mengganti implementasi metode getMood() sehingga mengembalikan string "happy" dan mengimplementasikan metode laugh() untuk mencetak "Hahaha".
- Kelas MoodyTest berisi metode main() yang digunakan untuk menguji polimorfisme. Di dalam metode main(), pertama-tama, objek m dari kelas MoodyObject dibuat. Kemudian, dipanggil metode speak() dari objek m. Setelah itu, objek m dialokasikan kembali sebagai objek HappyObject dan metode speak() dipanggil lagi, diikuti oleh metode laugh(). Terakhir, objek m dialokasikan kembali sebagai objek SadObject dan metode speak() dipanggil kembali, diikuti oleh metode cry().
- Kode diatas terjadi error, karena satu hal
  - Setiap public class harus ditulis pada filenya masing-masing.
- Cara memperbaikinya adalah:
  - Memisahkan setiap kelas pada masing-masing file atau menghapus modifier public pada class MoodyObject, SadObject, dan HappyObject.
- Jika sudah diperbaiki, maka output dari kode tersebut adalah:



Percobaan 6:

```java
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("Konstruktor A dijalankan");
  }
}
```



```java
class B extends A {
    B(){
        System.out.println("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("Objek A dibuat");
        A aa = new A();
        System.out.println("Menampilkan nama variabel dari object aa");
        System.out.println(aa.var_a);
        System.out.println(aa.var_b);
        System.out.println(aa.var_c);
        System.out.println(aa.var_d);
        System.out.println("");

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

    }
}
```

- 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 dalam konstruktor B, nilai dari var_a dan var_b diubah.
- Metode main terdapat di dalam kelas B dan digunakan untuk menguji pembuatan objek dari kedua 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 data var_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 tetap sesuai dengan nilai awal yang diinisialisasi di dalam kelas A.

- Output dari kode tersebut jika dijalankan, maka:

```
Objek A dibuat
Konstruktor A dijalankan
Menampilkan nama variabel dari object aa
Variable A
Variable B
Variable C
Variable D

Objek B dibuat
Konstruktor A dijalankan
Konstruktor B dijalankan
Menampilkan nama variabel dari object bb
Var_a dari class B
Var_a dari class B
Variable C
Variable D
```

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() {
        System.out.println("Nilai a=" + a);
        System.out.println("Nilai b=" + b);
        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 dua anggota data a dan b, serta sebuah metode show_variabel() yang mencetak nilai dari kedua anggota data tersebut.
- Kelas Anak adalah subkelas dari Bapak yang menambahkan satu anggota data c dan meng-override metode show_variabel() untuk mencetak nilai dari semua anggota data a, b, dan c.

- Kelas InheritExample berisi metode main() yang digunakan untuk menguji pewarisan. Di dalam metode main(), dibuat objek objectBapak dari kelas Bapak dan objek objectAnak dari kelas Anak. Nilai dari anggota data a dan b dari objek objectBapak diatur, kemudian metode show_variabel() dipanggil untuk mencetak nilai anggota data tersebut. Nilai dari anggota data c dari objek objectAnak juga diatur, kemudian metode show_variabel() dari objectAnak dipanggil untuk mencetak nilai anggota data dari kedua kelas, Bapak dan Anak.
- Saat metode show_variabel() dipanggil pada objek objectBapak, ia mencetak nilai a dan b yang sudah diatur sebelumnya (nilai 1).
- Saat metode show_variabel() dipanggil pada objek objectAnak, ia mencetak nilai a dan b dari kelas Bapak yang memiliki nilai default (nilai 0), dan nilai c yang sudah diatur sebelumnya (nilai 5).
- Output dari kode tersebut sebagai berikut:

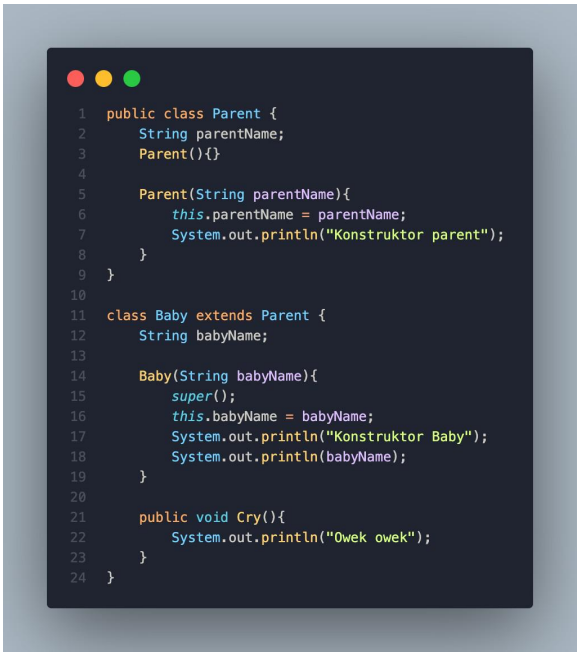
```
Object Bapak (Superclass):
Nilai a=1
Nilai b=1
Object anak (Superclass dari anak)
Nilai a=0
Nilai b=0
Nilai c=5
```

- Kemudian dilakukan modifikasi pada method show_variabel pada class Anak, dengan mengganti print a dan b menjadi super.show_variabel();. Ini mempersingkat penulisan, dan hasil kode akan sama seperti sebelumnya.

```
1  void show_variabel() {
2      super.show_variabel();
3      System.out.println("Nilai c=" + c);
4  }
```

```
Object Bapak (Superclass):
Nilai a=1
Nilai b=1
Object anak (Superclass dari anak)
Nilai a=0
Nilai b=0
Nilai c=5
```

Percobaan 8:



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

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

class Baby extends Parent {
    String babyName;

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

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

- Kelas Parent memiliki satu anggota data parentName yang tidak diinisialisasi secara langsung. Kelas ini memiliki dua konstruktor. Konstruktor pertama adalah konstruktor default yang tidak memiliki parameter dan tidak melakukan operasi apapun. Konstruktor kedua menerima satu parameter parentName dan menginisialisasi anggota data parentName dengan nilai dari parameter tersebut.
- Kelas Baby adalah subkelas dari Parent. Kelas ini memiliki satu anggota data babyName yang tidak diinisialisasi secara langsung. Konstruktor Baby menerima satu parameter babyName. Di dalam konstruktor tersebut, konstruktor tanpa parameter dari kelas Parent dipanggil menggunakan super(), kemudian anggota data babyName diinisialisasi dengan nilai dari parameter babyName. Selain itu, konstruktor mencetak pesan "Konstruktor Baby" dan nilai dari babyName.
- Kelas Baby memiliki metode Cry() yang mencetak pesan "Owek owek".
- Saat objek Baby dibuat, konstruktor Baby akan dipanggil. Konstruktor ini akan memanggil konstruktor Parent menggunakan super(), yang kemudian akan mencetak pesan "Konstruktor parent". Setelah itu, konstruktor Baby akan mencetak pesan "Konstruktor Baby" dan nilai babyName.