Nama : Restu Lestari Mulianingrum

NIM : A11.2022.14668

Kelompok : A11.4415

1. Jelaskan dan berikan contoh apa yang dimaksud dengan :
   1. **Konstruktor**

Konstruktor adalah method khusus dimana nama methodnya menggunakan nama class itu sendiri yang akan dieksekusi pada saat pembuatan objek (instance). Setiap kali sebuah objek dibuat menggunakan kata kunci new(), setidaknya satu konstruktor dipanggil. Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

|  |
| --- |
| **Contoh tanpa parameter** |
| public class MyClass {      // Konstruktor tanpa parameter      public MyClass() {          // Logika inisialisasi jika diperlukan      }  } |
| **Contoh dengan parameter** |
| public class Person {      private String name;      private int age;      // Konstruktor dengan parameter      public Person(String name, int age) {          this.name = name;          this.age = age;      } |

* 1. **Class, Object, Attribut, Method**
* Class

Class adalah blueprint atau cetakan untuk membuat objek. Class mendefinisikan atribut dan metode yang akan dimiliki oleh objek.

Contoh penulisan class dalam bahasa Java:

class Laptop {

    // Isi dari class Laptop...

}

* Object

Objek adalah sebuah variabel yang merupakan instance atau perwujudan dari Class. Jika menggunakan analogi class Laptop, maka objek dari class Laptop bisa berupa “laptopRestu”

public class Laptop {

    public static void main(String[] args) {

        // menciptakan object dengan nama 'laptopRestu'

        Laptop laptopRestu = new Laptop();

        laptopRestu.cetakNama("Grey");

    }

}

* Attribut

Attribut adalah variabel yang terdapat dalam class. Atribut menyimpan data atau keadaan dari sebuah objek.

Contoh attribut dari class Laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain. Contoh penulisan class dengan attribut:

class Laptop {

    String pemilik;

    String merk;

    double ukuranLayar;

}

* Method

Method adalah fungsi yang terdapat dalam class. Method mendefinisikan perilaku atau aksi yang bisa dilakukan oleh objek.

Contoh method pada class Laptop bisa berupa hidupkanLaptop() untuk menghidupkan laptop, matikanLaptop(), gantiCover().

class Laptop {

    String merk;

    String warna;

    int tahunProduksi;

    void hidupkanLaptop() {

        System.out.println("Laptop dinyalakan.");

    }

}

public class Main {

    public static void main(String[] args) {

        // Membuat objek laptop

        Laptop laptopRestu = new Laptop();

        laptopRestu.merk = "Lenovo";

        laptopRestu.warna = "Grey";

        laptopRestu.tahunProduksi = 2022;

        // Menggunakan metode pada objek

        laptopRestu.hidupkanLaptop();

    }

}

* 1. **Encapsulation**

Encapsulation adalah konsep yang menggabungkan data (attribut) dan metode (method) yang beroperasi pada data tersebut menjadi satu kesatuan. Tujuannya adalah untuk menyembunyikan detail implementasi dan membatasi akses langsung ke data internal suatu objek. Encapsulation bisa dicapai dengan menggunakan modifier seperti private, protected, dan public.

Berikut adalah contoh penggunaan encapsulation dalam sebuah kelas Person:

public class Person {

    private String name;

    private int age;

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

}

public class Main {

    public static void main(String[] args) {

        Person person = new Person();

        person.setName("John");

        person.setAge(30);

        System.out.println("Name: " + person.getName());

        System.out.println("Age: " + person.getAge());

    }

}

Dalam contoh di atas, kita menggunakan encapsulation dengan mendeklarasikan variabel instance kelas Person sebagai private, sehingga hanya dapat diakses di dalam kelas tersebut. Kemudian kita menggunakan getter dan setter untuk mengakses dan mengubah nilai variabel tersebut dari luar kelas

* 1. **Inheritance**

Inheritance (pewarisan) adalah konsep di mana sebuah class dapat mewarisi atribut dan metode dari class lain. Class yang mewarisi disebut subclass atau child class, sedangkan class yang memberikan warisan disebut superclass atau parent class. Contoh:

// Superclass (Parent Class)

    public static class Animal {

        private String name;

        public Animal(String name) {

            this.name = name;

        }

        public String getName() {

            return name;

        }

    }

    // Subclass Dog

    public static class Dog extends Animal {

        public Dog(String name) {

            super(name);

        }

    }

    // Main method untuk menguji pewarisan

    public static void main(String[] args) {

        Dog dog = new Dog("Molly");

        System.out.println(dog.getName());

    }

* 1. **Overriding :** konsep di mana sebuah method dalam class anak (subclass) menggantikan (meng-"override") method dengan nama yang sama dari class induknya (superclass).

Contoh:

class Animal {

    void sound() {

        System.out.println("Animal makes a sound");

    }

}

class Dog extends Animal {

    @Override

    void sound() {

        System.out.println("Dog barks");

    }

}

**Overloading :** adalah kemampuan untuk mendefinisikan beberapa metode dengan nama yang sama di dalam sebuah class. Metode-metode ini memiliki parameter yang berbeda. Java membedakan metode berdasarkan jumlah dan tipe parameter. Contoh:

class Kalkulator {

    int tambah(int a, int b) {

        return a + b;

    }

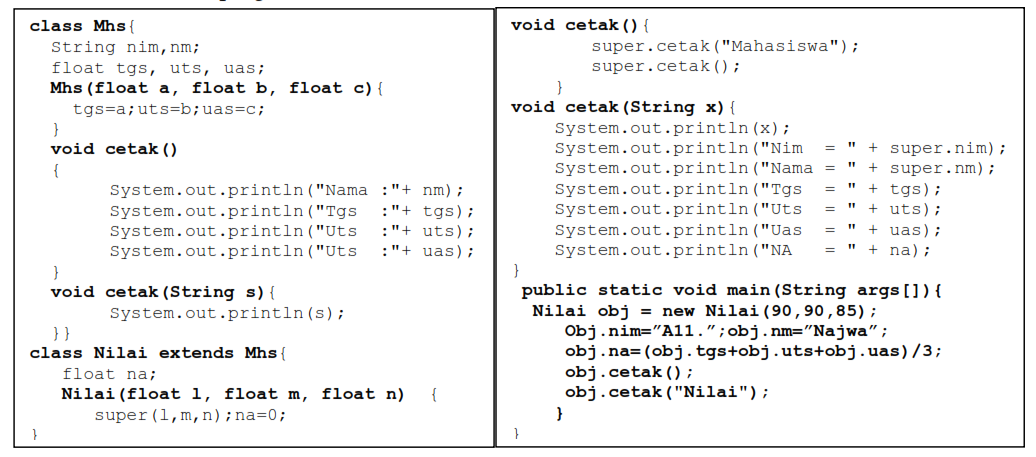
    double tambah(double a, double b) {

        return a + b;

    }

}

1. Perhatikan code program berikut :



Tentukan :

1. Konstruktor

* Kelas Mhs memiliki konstruktor Mhs(float a, float b, float c). Konstruktor ini menerima tiga parameter float a, b, dan c, yang digunakan untuk menginisialisasi atribut tgs, uts, dan uas.
* Kelas Nilai memiliki konstruktor Nilai(float l, float m, float n) yang memanggil konstruktor kelas induk Mhs menggunakan super(l, m, n).

1. Inheritance

Kelas Nilai meng-extend (mewarisi) dari kelas Mhs. Ini berarti bahwa kelas Nilai mewarisi semua properti (field dan method) dari kelas Mhs. Inheritance memungkinkan kelas Nilai menggunakan kembali kode yang didefinisikan di kelas Mhs tanpa menggandakannya.

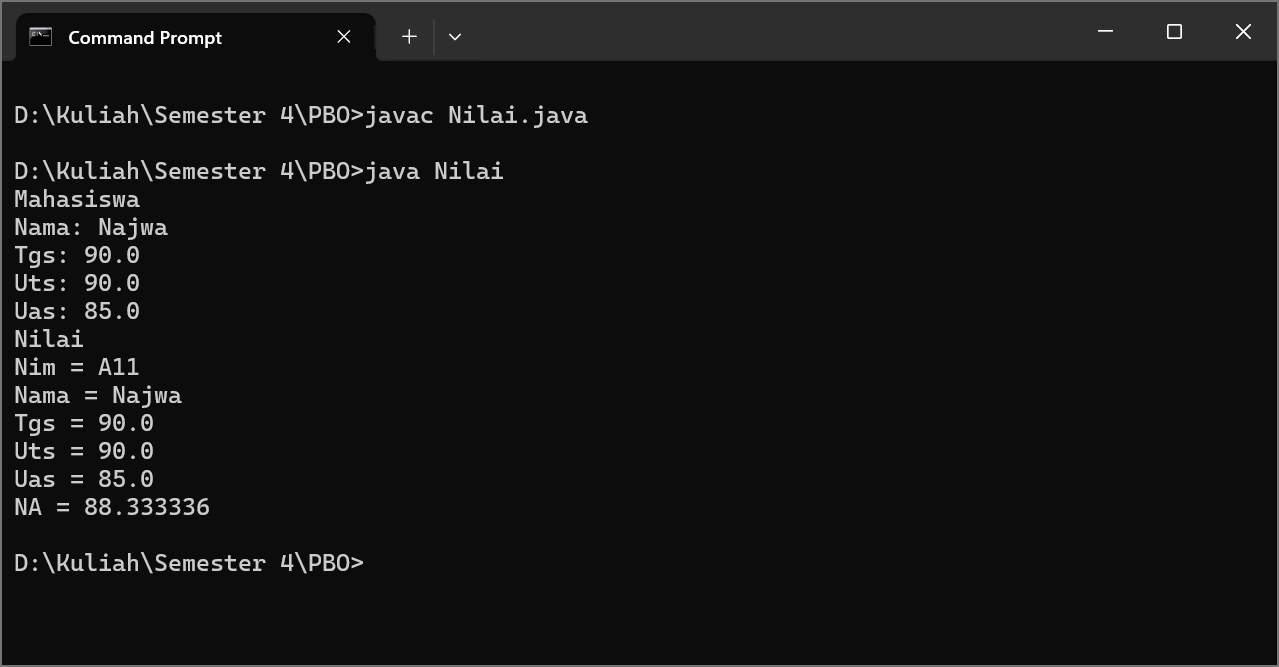
1. Overloading dan Overriding:

* Method Overloading:

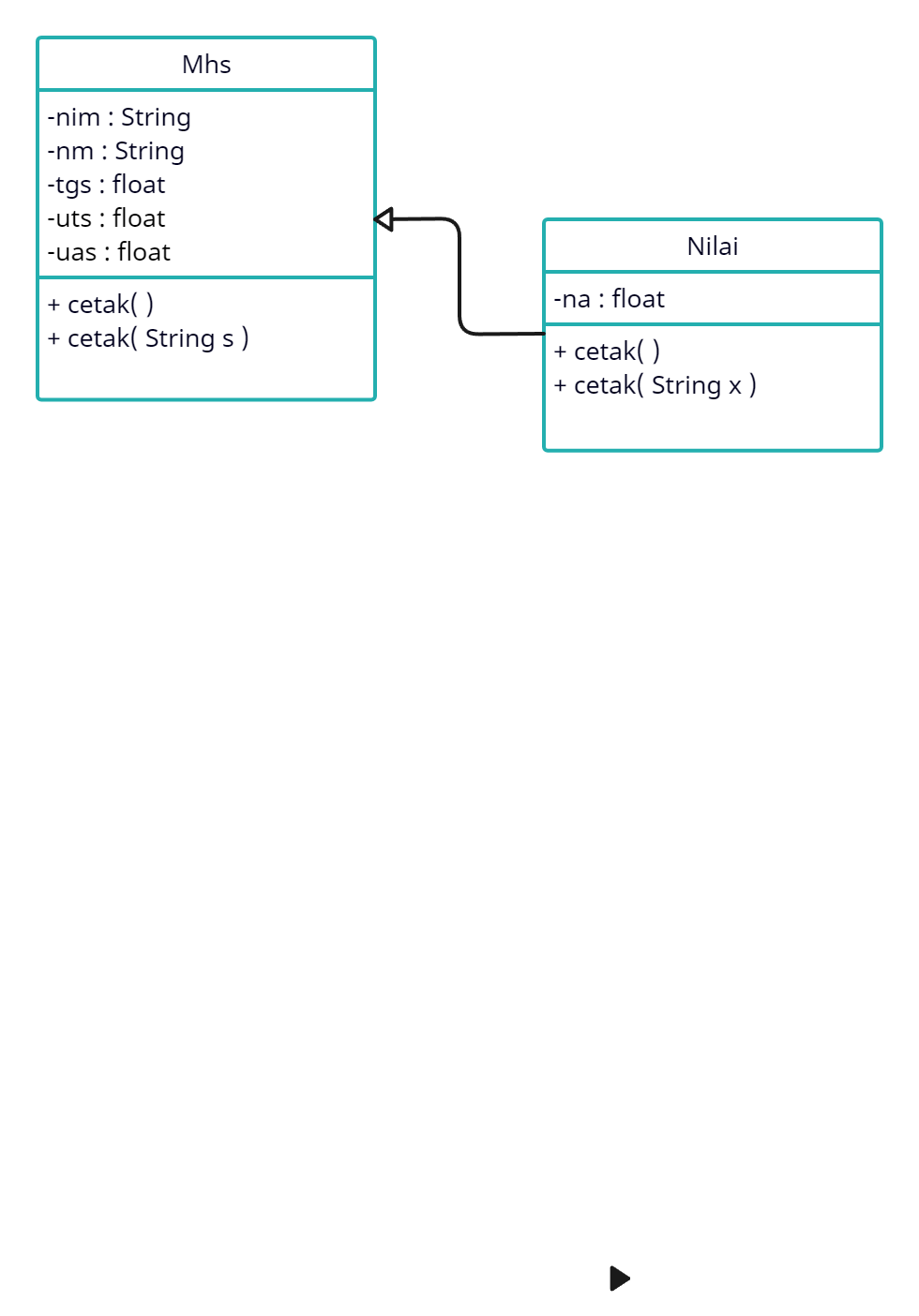
1. Terdapat dua metode cetak() dalam kelas Mhs. Yang pertama tidak memiliki parameter dan yang kedua memiliki parameter String cetak(String s).
2. Pada kelas Nilai, terdapat dua metode cetak(), yang merupakan metode overriding dari kelas Mhs: Metode cetak() tanpa parameter dan Metode cetak() dengan parameter String x.

* Method Overriding

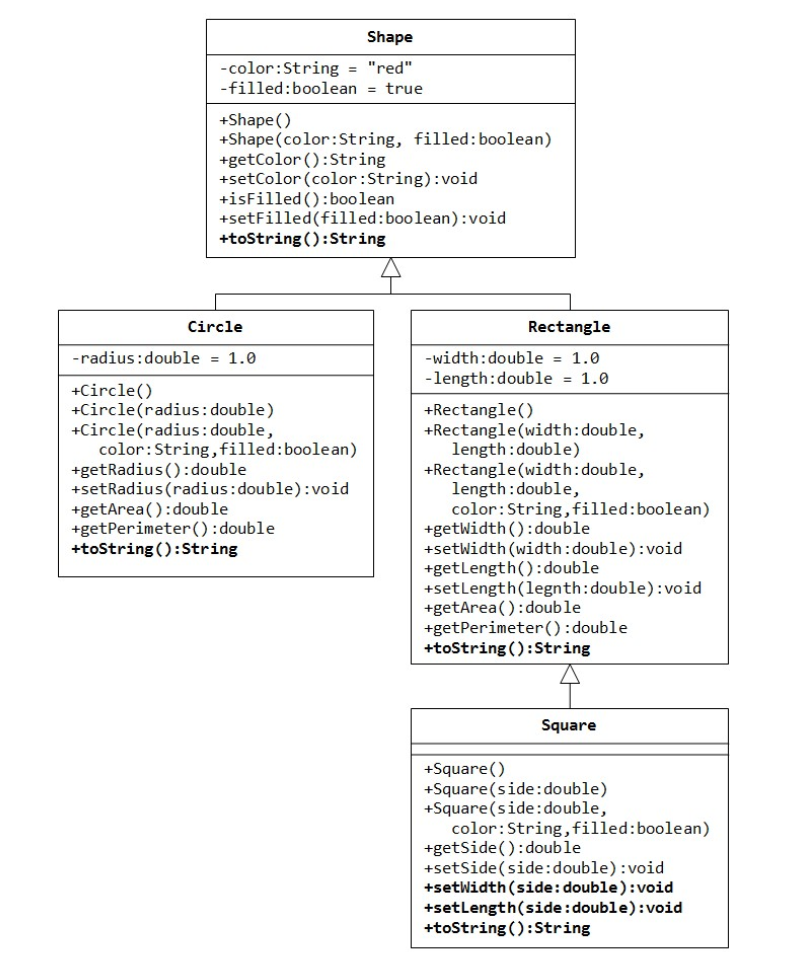
1. Metode cetak() di kelas Nilai mengoverride metode cetak() dari kelas Mhs. Ini memanggil metode cetak(String) dengan argumen "Mahasiswa" dan memanggil metode cetak() dari kelas Mhs sendiri.
2. Metode cetak(String) di kelas Nilai mengoverride metode cetak(String) dari kelas Mhs. Ini mencetak string yang diberikan (x) dan juga mencetak informasi nama, NIM, tugas, UTS, UAS, dan nilai akhir (na) dari mahasiswa.Hasil Program :



1. Gambar Class Diagram



1. Perhatikan class diagram berikut :



1. Buatlah design class dari class diagram di atas!

|  |
| --- |
| **Class Shape** |
| class Shape {      private String color = "red";      private boolean filled = true;      public Shape(){      };      public Shape(String color, boolean filled){          this.color = color;          this.filled = filled;      }      public String getColor(){          return color;      }        public void setColor (String color){          this.color = color;      }      public boolean isFilled(){          return filled;      }      public void setFilled (boolean filled){          this.filled = filled;      }      public String toString() {          return "Shape[color=" + color + ", filled=" + filled + "]";      }  } |

|  |
| --- |
| **Class Circle** |
| class Circle extends Shape{      private double radius = 1.0;      public Circle(){      }      public Circle(double radius){          this.radius = radius;      }      public Circle(double radius, String color, boolean filled){          super(color, filled);          this.radius = radius;      }      public double getRadius(){          return radius;      }      public void setRadius(double radius){          this.radius = radius;      }      public double getArea(){          return Math.PI \* radius \* radius;      }      public double getPerimeter() {          return 2 \* Math.PI \* radius;      }      public String toString() {          return "Circle[Shape" + super.toString() + ", radius=" + radius + "]";      }  } |

|  |
| --- |
| **Class Rectangle** |
| class Rectangle extends Shape {      private double width = 1.0;      private double length = 1.0;      public Rectangle() {      }      public Rectangle(double width, double length) {          this.width = width;          this.length = length;      }      public Rectangle(double width, double length, String color, boolean filled) {          super(color, filled);          this.width = width;          this.length = length;      }      public double getWidth() {          return width;      }      public void setWidth(double width) {          this.width = width;      }      public double getLength() {          return length;      }      public void setLength(double length) {          this.length = length;      }      public double getArea() {          return this.width \* this.length;      }      public double getPerimeter() {          return 2 \* (this.width + this.length);      }      @Override      public String toString() {          return "Rectangle[Shape" + super.toString() + ", width=" + width + ", length=" + length + "]";      }  } |

|  |
| --- |
| **Class Square** |
| class Square extends Rectangle {      public Square() {      }      public Square(double side) {          super(side, side);      }      public Square(double side, String color, boolean filled) {          super(side, side, color, filled);      }      public double getSide(){          return super.getLength();      }      public void setSide(double side){          super.setLength(side);          super.setWidth(side);      }      public void setWidth(double side){          super.setWidth(side);      }      public void setLength(double side){          super.setLength(side);      }      public String toString(){          return "Square[" + super.toString() + "]";      }  } |

1. Berikan contoh penerapan untuk mencetak Rectangle yang ada di design class pada soal (a) dengan output :

Warna :

Tinggi :

Lebar :

Luas :

|  |
| --- |
| **testRectangle.java** |
| public class testRectangle {      public static void main(String[] args) {          Rectangle rectangle = new Rectangle(4.0, 6.0, "Merah", true);          System.out.println("Warna: " + rectangle.getColor());          System.out.println("Tinggi: " + rectangle.getLength());          System.out.println("Lebar: " + rectangle.getWidth());          System.out.println("Luas: " + rectangle.getArea());      }  } |
| **Output** |
|  |