1. Jelaskan dan berikan contoh apa yang dimaksud dengan :

   a. 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;
    }
```

   b. 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.");
    }

    void matikanLaptop() {
        System.out.println("Laptop dimatikan.");
    }

    void gantiCover(String warnaBaru) {
```

```
        warna = warnaBaru;
        System.out.println("Cover laptop diganti menjadi " +
warnaBaru + ".");
    }
}

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();
        laptopRestu.gantiCover("Merah");
        laptopRestu.matikanLaptop();
    }
}
```

c. 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

d. 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:

```
class Kendaraan {
    // Atribut dan metode untuk semua kendaraan
```

```
}

class Mobil extends Kendaraan {
    // Atribut dan metode khusus untuk mobil
}
```

e. **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;
    }
}
```

2. Perhatikan code program berikut :

   Tentukan :

   a. 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).

   b. 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.

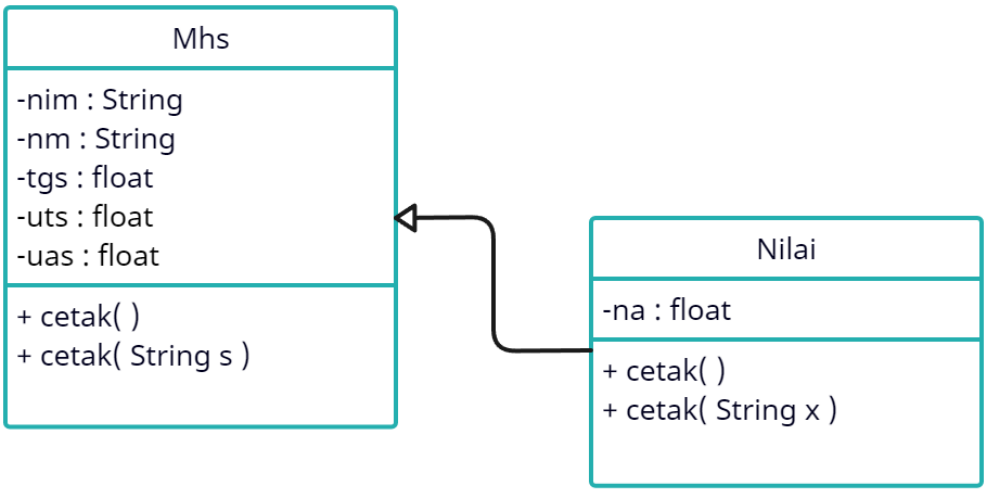
   c. Overloading dan Overriding:
      - Method Overloading:
        1. Terdapat dua metode cetak() dalam kelas Mhs. Yang pertama tidak memiliki parameter dan yang kedua memiliki parameter String cetak(String s).
        2. Pada kelas Nilai, terdapat dua metode cetak(), yang merupakan metode overriding 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.

d. Hasil Program

```
Mahasiswa
Nama: Najwa
Tgs: 95.0
Uts: 95.0
Uas: 90.0
Nilai
Nim = A11
Nama = Najwa
Tgs = 95.0
Uts = 95.0
Uas = 90.0
NA = 93.333336
```

e. Gambar Class Diagram



Ada tambahan atribut bonus di class nilai