Nama : M. Hasan Basri
Kelas : SIB2C
NIM : 2241760139

JOBSHEET W08

ABSTRACT CLASS

1. KOMPETENSI

1. Memahami konsep dasar dan tujuan abstract class
2. Mampu menerapkan abstract class dalam suatu kode program
3. Mampu membuat subclass yang meng-extend abstract class dengan mengimplementasikan seluruh abstract method-nya

2. PENDAHULUAN

Abstract class merupakan class yang **tidak dapat diinstansiasi namun dapat di-extend**. Umumnya abstract class digunakan sebagai **generalisasi** atau **guideline** dari subclass dan hanya bisa digunakan lebih lanjut setelah **di-extend** oleh **concrete class** (class pada umumnya)

Abstract class memiliki karakteristik sebagai berikut:

1. Selalu dideklarasikan dengan menggunakan keyword "**abstract class**"
2. Dapat memiliki atribut dan methods (yang bukan abstract method) seperti concrete class
3. Umumnya memiliki **abstract method**, yaitu method yang hanya dideklarasikan tetapi tidak memiliki implementasi (body)
    - Abstract method mendifinisikan apa saja yang bisa dilakukan oleh sebuah class namun tidak ada detail bagaimana cara melakukannya
    - Untuk membuat abstract method, hanya menuliskan deklarasi method tanpa body dan menggunakan keyword abstract

Untuk mendeklarasikan abstract class:

public abstract class <NamaClass>

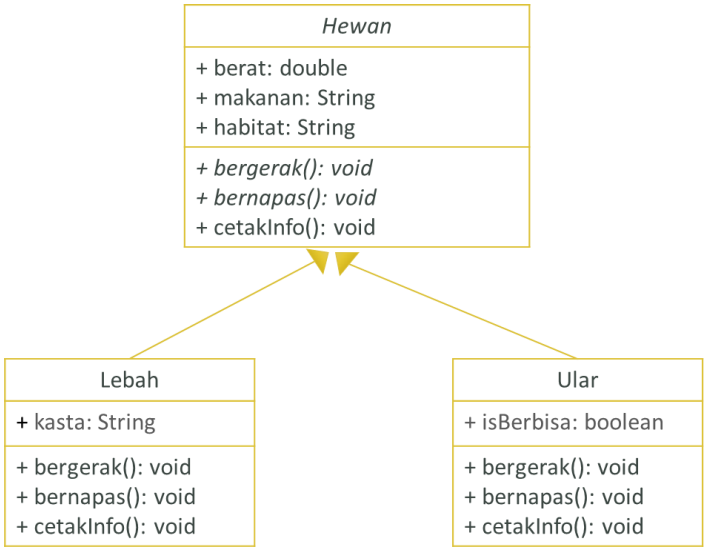
Untuk mendeklarasikan method abstract:

public abstract <return_type> <namaMethod>();

Contoh:

```
public abstract class Hewan {
    public abstract void bergerak();
    public abstract void bernapas();
}
```

Secara umum, notasi class diagram untuk abstract class sama dengan concrete class, namun nama kelas dicetak miring atau ditambah anotasi <<abstract>> di atas nama kelas. Di samping itu abstract method harus dicetak miring juga seperti pada contoh berikut.



Cara menggunaan abstract class:

- Abstract class tidak dapat diinstansiasi (tidak dapat dibuat objectnya). Baris kode berikut akan memunculkan *compilation error* "Hewan is abstract; cannot be instantiated"

  Hewan hewan1 = new Hewan();
- Untuk menggunakan abstract class, dibuat concrete class yang meng-extend abstract class tersebut
  - concrete class menggunakan extends keyword
  - concrete class harus mengimplementasi semua abstract method
- Class yang menge-extend abstract class tetapi tidak mengimplementasi seluruh abstract method nya maka harus dideklarasikan sebagai abstract class juga

Fungsi abstract class:

- Mencegah suatu class diinstansiasi atau dibuat objeknya
- Sebagai generalisasi/superclass pada class hierarki
- Sebagai guideline untuk subclass dengan cara memaksa subclass untuk mengimplementasikan abstract method

3. PERCOBAAN

A. PERCOBAAN 1

1. Buatlah project baru dengan nama Praktikum08 kemudian buat class baru dengan nama Hewan. Method bernapas dan bergerak tidak memiliki statement atau baris kode.

```
public class Hewan {
    public double berat;
    public String makanan;
    public String habitat;

    public Hewan(double berat, String makanan, String habitat) {
        this.berat = berat;
        this.makanan = makanan;
        this.habitat = habitat;
    }

    public void bergerak(){
        //
    }

    public void bernapas(){
        //
    }

    public void cetakInfo(){
        System.out.println("Berat   : " + this.berat);
        System.out.println("Makanan : " + this.makanan);
        System.out.println("Habitat : " + this.habitat);
    }
}
```

2. Buat class Lebah sebagai subclass dari class Hewan sebagai berikut

```
public class Lebah extends Hewan{
    public String kasta;

    public Lebah(String kasta, double berat, String makanan, String habitat) {
        super(berat, makanan, habitat);
        this.kasta = kasta;
    }
}
```

3. Buat class main dengan nama AbstractClassDemo lalu instansiasi objek dari class Hewan dan class Lebah. Run program kemudian amati hasilnya.

```
public class AbstractClassDemo {
    public static void main(String[] args) {
        Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
        hewan1.cetakInfo();
        hewan1.bergerak();
        hewan1.bernapas();

        Lebah lebah1 = new Lebah("Ratu", 0.05, "Nektar", "Hutan");
        lebah1.cetakInfo();
        lebah1.bergerak();
        lebah1.bernapas();
    }
}
```

Output :

```
Berat: 10.0
Makanan: Rumput
Habitat: Savana
Berat: 0.05
Makanan: Nektar
Habitat: Hutan
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum8>
```

B. PERTANYAAN

1. Bagaimana hasil pada langkah 3? Apakah objek hewan1 dan lebah1 berhasil diinstansiasi?

   Jawab : Iya, object dari hewan 1 dan hewan 2 berhasil diinstansiasi.

2. Menurut Anda, mengapa tidak ada baris program pada method bergerak() dan bernapas() pada class Hewan()?

   Jawab : Tidak terdapat baris program karena hanya digunakan untuk penanda bahwa hewan bisa bergerak dan bernafas, namun karena pada baris program pada method bergerak() dan bernapas() masih belum tau apa yang akan dilakukan dimethod tersebut(karena bisa saja proses /cara bernafas dan bergerak setiap hewan berbeda).

3. Class Lebah tidak memiliki method bergerak(), bernapas(), dan cetakInfo(), mengapa tidak terjadi error pada AbstractClassDemo?

   Jawab : Tidak terjadi error pada AbstractClassDemo karena kelas Lebah mewarisi method cetakInfo(), bergerak(), dan bernapas() dari kelas Hewan.

   tidak terjadi error, namun tidak bisa memunculkan output karena tidak ada program dalam method tersebut

C. PERCOBAAN 2

1. Ubah method bergerak dan bernapas menjadi abstract method.

```
public class Hewan {
    public double berat;
    public String makanan;
    public String habitat;

    public Hewan(double berat, String makanan, String habitat) {
        this.berat = berat;
        this.makanan = makanan;
        this.habitat = habitat;
    }

    public abstract void bergerak();
    public abstract void bernapas();

    public void cetakInfo(){
        System.out.println("Berat   : " + this.berat);
        System.out.println("Makanan : " + this.makanan);
        System.out.println("Habitat : " + this.habitat);
    }
}
```

2. Akan muncul error sebagai berikut

```
public class Hewan {    The type Hewan must be an abstract class to define abstract methods
    public double berat;
```

3. Ubah class Hewan menjadi abstract Class. Jalankan program kemudian amati hasilnya.

```
public abstract class Hewan {
    public double berat;
    public String makanan;
    public String habitat;

    public Hewan(double berat, String makanan, String habitat) {
        this.berat = berat;
        this.makanan = makanan;
        this.habitat = habitat;
    }

    public abstract void bergerak();
    public abstract void bernapas();

    public void cetakInfo(){
        System.out.println("Berat   : " + this.berat);
        System.out.println("Makanan : " + this.makanan);
        System.out.println("Habitat : " + this.habitat);
    }
}
```

4. Ubah class demo sebagai berikut. Run program kemudian amati hasilnya

```
public class AbstractClassDemo {
    public static void main(String[] args) {
        Hewan hewan1 = new Hewan(10, "Rumput", "Savana");
        hewan1.cetakInfo();
        hewan1.bergerak();
        hewan1.bernapas();
    }
}
```

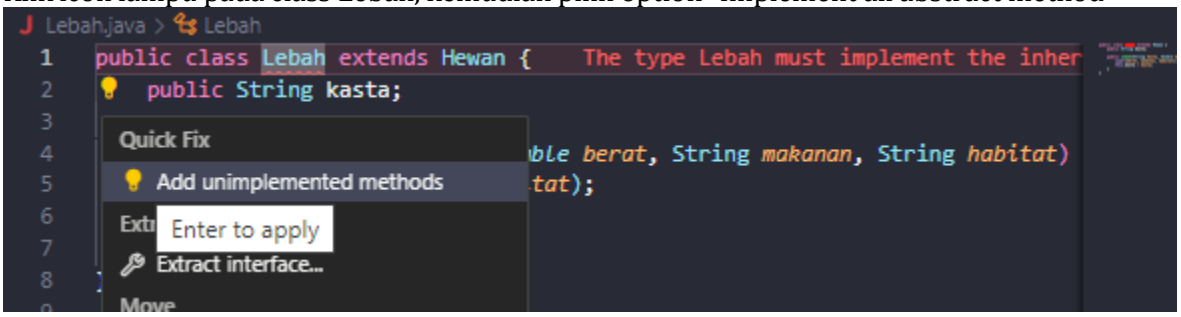
```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Cannot instantiate the type Hewan

        at AbstractClassDemo.main(AbstractClassDemo.java:3)
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum8>
```

5. Ubah class demo sebagai berikut. Run program kemudian amati hasilnya

```
public class AbstractClassDemo {
    public static void main(String[] args) {
        Lebah lebah1 = new Lebah("Ratu", 0.05, "Nektar", "Hutan");
        lebah1.cetakInfo();
        lebah1.bergerak();
        lebah1.bernapas();
    }
}
```

6. Klik icon lampu pada class Lebah, kemudian pilih option "Implement all abstract method"



7. Implementasi method bergerak dan bernapas pada class Lebah sebagai berikut. Run program kemudian amati hasilnya.

```
@Override
public void bernapas(){
    System.out.println("Otot perut mengendur, udara masuk melalui lubang di segmen tubuh");
    System.out.println("Trakea mengirimkan oksigen");
    System.out.println("Otot perut berkontraksi, udara dikeluarkan");
}

@Override
public void bergerak(){
    System.out.println("Mengepakkan sayap ke depan");
    System.out.println("Memutar sayap hampir 90 derajat");
    System.out.println("Mengepakkan sayap ke belakang");
}
```

8. Tambahkan method cetakInfo() pada class Lebah. Run program kemudian amati hasilnya.

```
@Override
public void cetakInfo() {
    super.cetakInfo();
    System.out.println("Kasta    :" + this.kasta);
}
```

```
Berat: 0.05
Makanan: Nektar
Habitat: Hutan
Kasta: Ratu
Mengepakkan sayap ke depan
Memutar sayap hampir 90 derajat
Mengepakkan sayap ke belakang
Otot perut mengendur, udara masuk melalui lubang di segmen tubuh
Trakea mengirimkan oksigen
Otot perut berkontraksi, udara dikeluarkan
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum8>
```

9. Buat class Ular kemudian sebagai berikut. Instansiasi objek bertipe Ular pada class AbstractClassDemo. Ekseksusi ketiga method untuk object tersebut.

```
public class Ular extends Hewan{
    public boolean isBerbisa;

    public Ular(boolean isBerbisa, double berat, String makanan, String habitat) {
        super(berat, makanan, habitat);
        this.isBerbisa = isBerbisa;
    }

    @Override
    public void bergerak(){
        System.out.println("Mengerutkan otot dari segala sisi hingga membentuk lengkungan");
        System.out.println("Menemukan titik penahan seperti batu atau pohon");
        System.out.println("Menggunakan sisik untuk mendorong tubah ke depan");
    }

    @Override
    public void bernapas(){
        System.out.println("Otot tulang rusuk kontraksi, udara masuk lewat hidung");
        System.out.println("Trakea mengirimkan udara ke paru-paru");
        System.out.println("Otot tulang rusuk relaksasi, udara dikeluarkan lewat hidung");
    }

    @Override
    public void cetakInfo() {
        super.cetakInfo();
        System.out.println("Berbisa   :" + (this.isBerbisa ? "Ya" : "Tidak"));
    }
}
```

```java
public class AbstractClassDemo {
    Run | Debug
    public static void main(String[] args) {
        Lebah lebah1 = new Lebah(kasta:"Ratu", berat:0.05, makanan:"Nektar", habitat:"Hutan");
        lebah1.cetakInfo();
        lebah1.bergerak();
        lebah1.bernapas();
System.out.println(x:"======================================================================");
        Ular ular1 = new Ular(isBerbisa:true, berat:5.5, makanan:"Daging", habitat:"Hutan");
        ular1.cetakInfo();
        ular1.bergerak();
        ular1.bernapas();
    }
}
```

Output :

```
Berat: 0.05
Makanan: Nektar
Habitat: Hutan
Kasta: Ratu
Mengepakkan sayap ke depan
Memutar sayap hampir 90 derajat
Mengepakkan sayap ke belakang
Otot perut mengendur, udara masuk melalui lubang di segmen tubuh
Trakea mengirimkan oksigen
Otot perut berkontraksi, udara dikeluarkan
======================================================================
Berat: 5.5
Makanan: Daging
Habitat: Hutan
Berbisa: Ya
Mengerutkan otot dari segala sisi hingga membentuk lengkungan
Menemukan titik penahan seperti batu atau pohon
Menggunakan sisik untuk mendorong tubuh ke depan
Otot tulang rusuk kontraksi, udara masuk lewat hidung
Trakea mengirimkan udara ke paru-paru
Otot tulang rusuk relaksasi, udara dikeluarkan lewat hidung
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum8>
```

D. PERTANYAAN

1. Pada langkah 1, mengapa sebaiknya method bergerak() dan bernapas() dideklarasikan sebagai abstract method?
   Jawab : Karena keduanya merupakan perilaku yang berbeda-beda tergantung pada jenis hewan yang diturunkan. Dengan mendeklarasikan method tersebut sebagai abstract, kelas turunan yang mewarisi kelas Hewan wajib mengimplementasikan method tersebut sesuai dengan perilaku spesifik hewan tersebut.

2. Mengapa pada langkah 2 muncul error?
   Jawab : Pada langkah 2, muncul error karena kelas Hewan tidak dideklarasikan sebagai abstract class, tetapi memiliki abstract method bernapas(). Hal ini bertentangan dengan aturan dalam Java dimana sebuah class yang memiliki abstract method harus dideklarasikan sebagai abstract class.

3. Apakah sebuah class yang memiliki abstract method harus dideklarasikan sebagai abstract class?
   Jawab : Ya, sebuah class yang memiliki abstract method harus dideklarasikan sebagai abstract class.

4. Sebaliknya, apakah abstract class harus memiliki abstract method?
   Jawab : Tidak, sebuah abstract class tidak harus memiliki abstract method. Abstract class dapat memiliki implementasi lengkap dari semua methodnya dan bahkan tidak memiliki method abstract sama sekali.

5. Mengapa muncul error pada langkah 4?
   Jawab : Pada langkah 4, muncul error karena class Hewan dideklarasikan sebagai abstract class, dan objek tidak dapat diinstansiasi dari class abstract.

6. Apakah abstract class dapat memiliki constructor?
   Jawab : Ya, abstract class dapat memiliki constructor, tetapi constructor tersebut tidak dapat digunakan untuk membuat objek dari abstract class. Constructor abstract class biasanya digunakan untuk melakukan inisialisasi atau setup yang diperlukan oleh class turunannya.

7. Apakah constructor abstract class dapat dipanggil?
   Jawab : Tidak, constructor abstract class tidak dapat dipanggil secara langsung. Constructor abstract class hanya akan dipanggil secara tidak langsung melalui constructor dari class turunannya.

8. Pada langkah 6-8, mengapa method bergerak() dan bernapas() **harus** di-override, namun method cetakInfo() **tidak harus** di-override?
   Jawab : Karena method bergerak() dan bernapas() dideklarasikan sebagai abstract method di class Hewan, sehingga class turunan yang meng-extend Hewan wajib mengimplementasikan method tersebut.
   Namun, method cetakInfo() memiliki implementasi lengkap di class Hewan (tidak di deklarasikan sebagai abstract method), sehingga tidak wajib di-override di class turunannya.

9. Simpulkan kegunaan dari abstract method
   Jawab : Abstract method digunakan untuk mendefinisikan perilaku yang harus diimplementasikan oleh class turunan. Dengan menggunakan abstract method, kita dapat menetapkan pola perilaku yang harus diikuti oleh class turunan, namun implementasi detilnya dibiarkan tergantung pada class turunan tersebut.

10. Simpulkan kegunaan dari abstract class
    Jawab : Abstract class digunakan untuk menyediakan kerangka kerja untuk class turunannya. Abstract class dapat mendefinisikan method abstract yang harus diimplementasikan oleh class turunan, serta method non-abstract yang dapat digunakan secara langsung oleh class turunan. Abstract class juga dapat memiliki implementasi default untuk method-method tertentu yang dapat digunakan oleh semua class turunannya.

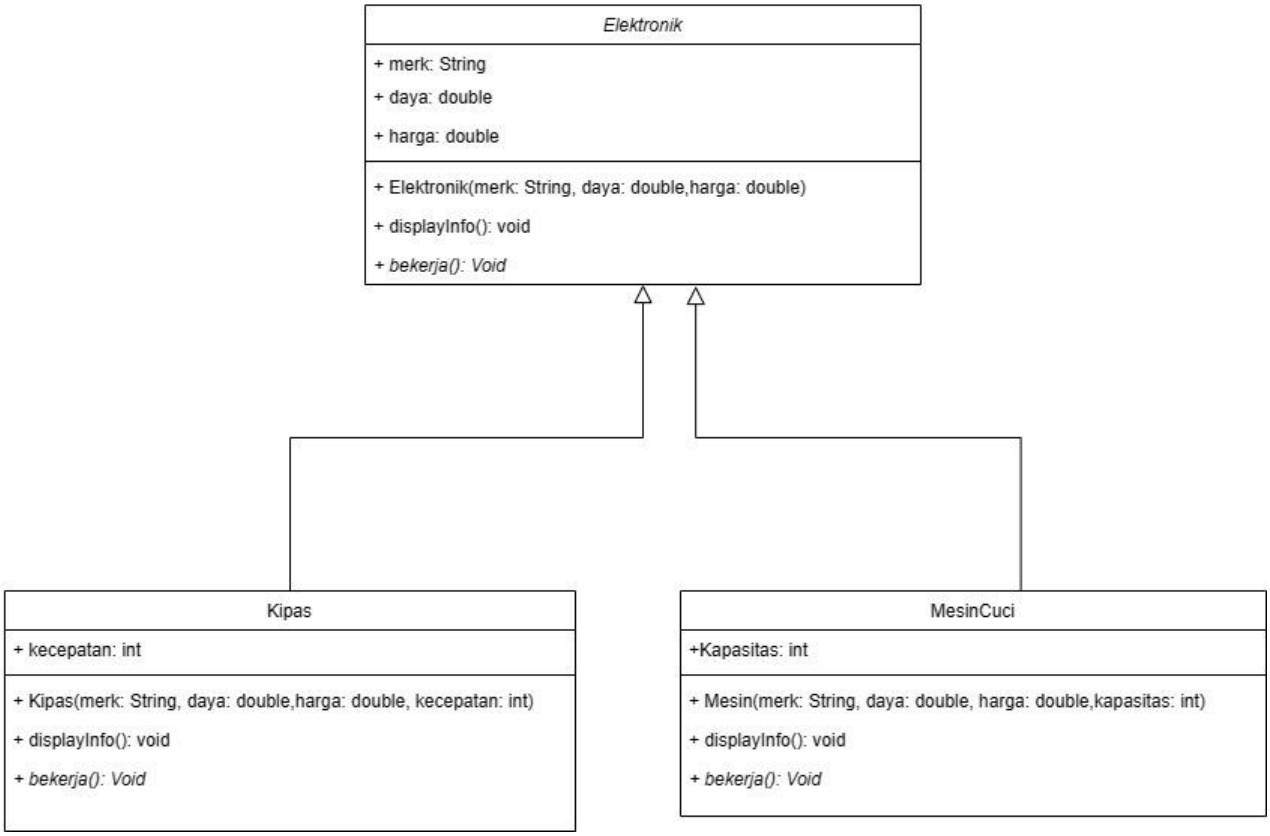
4. TUGAS

Implementasikan class diagram yang telah dirancang pada tugas PBO Teori ke dalam kode program. Selanjutnya buatlah instansiasi objek dari masing-masing subclass kemudian coba eksekusi method-method yang dimiliki.

TUGAS ABSTRACT CLASS

Nama : M. Hasan Basri
Kelas : SIB 2C
NIM : 2241760139

Macam Elektronic



Output setelah implementasi:

```
Informasi Kipas:
Merk: Kipas angin polytron
Daya: 50.0 Watt
Harga: Rp 250000.0
Kecepatan: 1200 RPM
Kipas mulai bekerja...
Pisau kipas berputar dengan kecepatan tinggi.
Angin dibelokkan dan ditiup keluar dari kipas.
Kipas memberikan pendinginan di sekitarnya.
======================================================================
Informasi Mesin Cuci:
Merk: Mesin Cuci Sharp
Daya: 500.0 Watt
Harga: Rp 1500000.0
Kapasitas: 7 kg
Mesin cuci mulai mencuci...
Air dimasukkan ke dalam tabung pencucian.
Deterjen ditambahkan ke dalam air.
Pakaian diaduk dalam air yang berisi deterjen.
Air kotor dibuang dan dilakukan pembilasan.
Proses mencuci selesai.
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum8\Kode classDiagram>
```