Name : Faricha Aulia
NIM : 2141720155
Class : 2I / D-IV Informatics Engineering

Jobsheet 10
Abstract Class

I. Competence

After finishing this sheet work students are expected able :

1. Explain intent and purpose use of Abstract Class
2. Implementing Abstract Class in programming

II. Introduction

Abstract Class

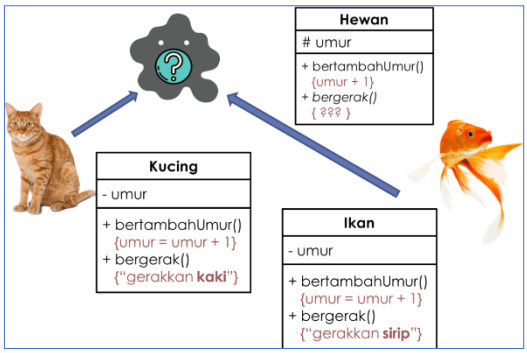
Abstract Class is a class that is not could instantiated but can be *extended* . *New abstract class* could utilized when it is extended .

Characteristics :

a. Could have *properties* and *methods s* like normal classes .
b. Always have *methods* that don't have body (only the declaration only), also called *abstract method* .
c. Always declared with use keywords abstract class .

Uses :

Describe something that is general , which is only can working after he described to in more shape _ specific .

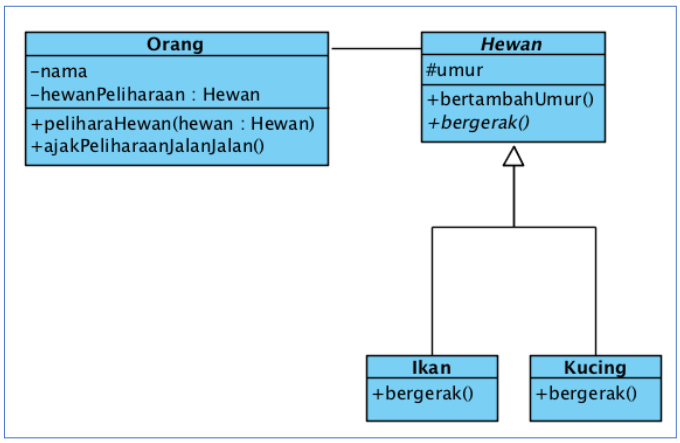


III. Practice

Experiment 1: Abstract Class

In this world there is many type animal . All animal have a number of the same characteristics , such as for example all animal have age , animal whatever that , his age will increase same amount every year .

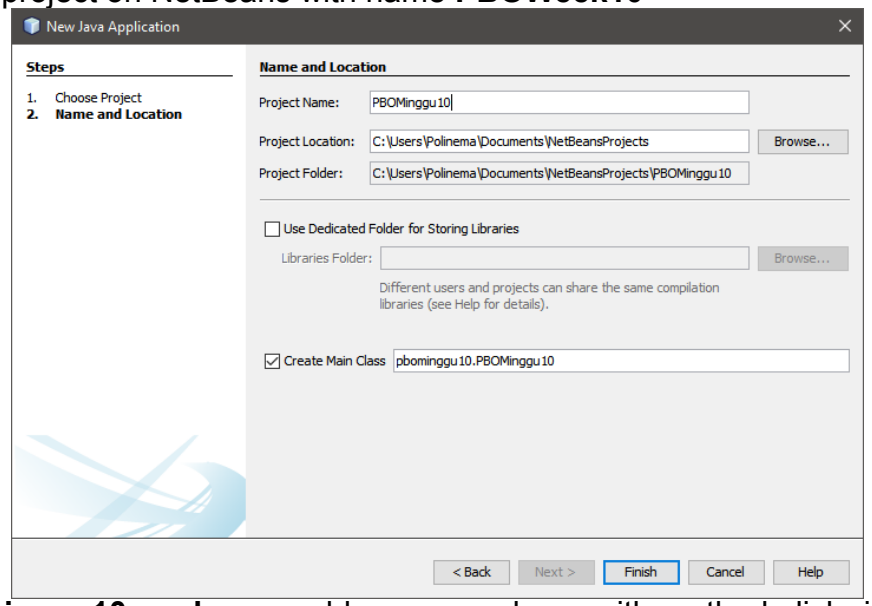
Besides the same characteristics , each animal also has different characteristics _ one with the others . For example in Thing **move** . cat way move different with the way the fish move Cat move with method put their feet while the fish move with method move the fin .



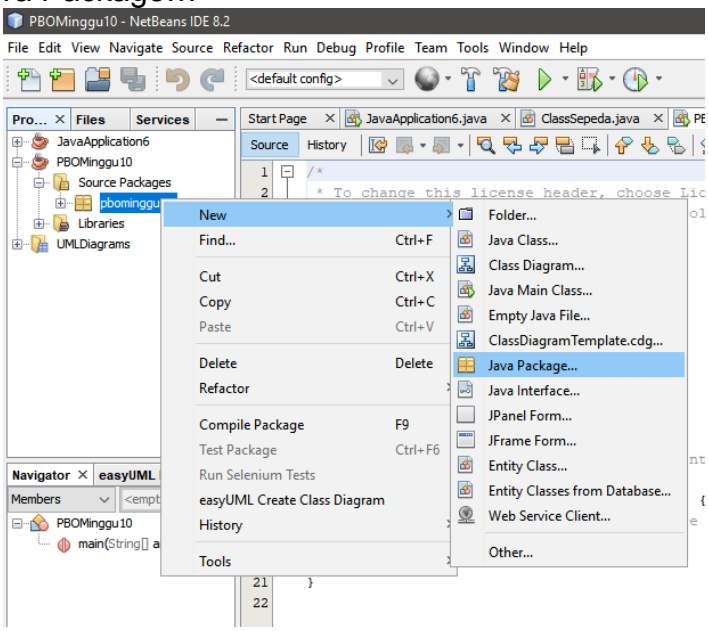
Everyone who maintains animal could invite animal his pet walk (make the animal his pet move). But the one who keeps different animals , will _ different way _ animal his pet in move .

On trial first this our will make a program that describes the scenario in on with utilise **abstract class.**

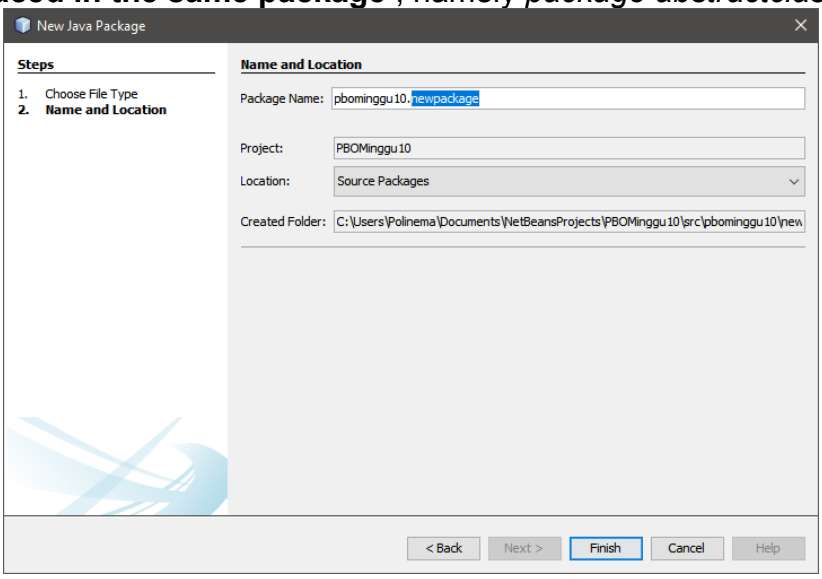
1. make a new project on NetBeans with name **PBOWeek10**



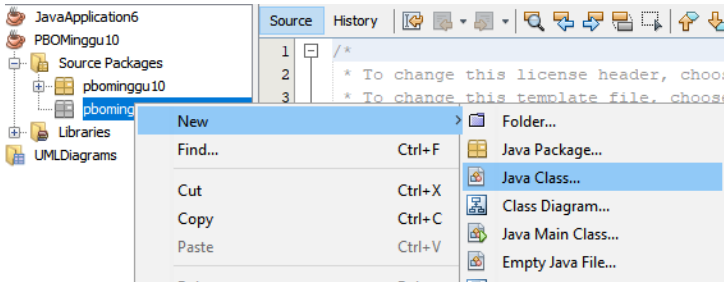
2. In **the pbominggu10 package** , add a new package with method click right package →name New →Java Package…



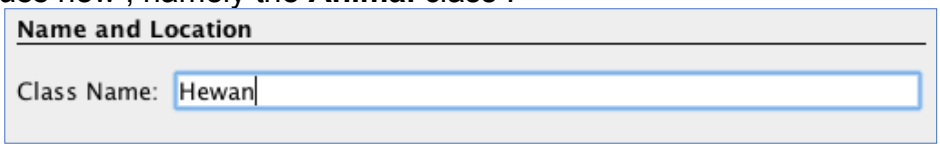
3. Name the package _ with name **abstractclass** . All classes created in this 1st experiment **placed in the same package** , namely *package abstractclass* this ,



4. new package the add *class* new .



5. Name _ *class* new , namely the **Animal** class .



6. In *class* Animal it , type code following this .

```
public abstract class Hewan
{
    private int umur;

    protected Hewan()
    {
        this.umur = 0;
    }

    public void bertambahUmur()
    {
        this.umur += 1;
    }

    public abstract void bergerak();
}
```

Animal Class the is an abstract class containing ordinary properties and methods , plus a *abstract method* named **move ( )** . *Method* the in front of him there are keywords **abstract** and not has a function body . this method later will be *overridden* by whichever *class* becomes _ _ *class* derivative from the Animal class that .

7. With the same way , create a class with name **A cat** that *extends* the Animal class . Inside the Cat class , after you write _ code like in down , then will light icon appears warning . Click lamp that and then choose **implement all abstract methods** .

```
    public class Kucing extends Hewan{
13
14  }
```

8. So will by automatic made *overriding* function _ function *abstract* **move ( )** in the animal class .

```
public class Kucing extends Hewan{

    @Override
    public void bergerak() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

}
```

9. Change the function body the with replace code inside it Becomes like following .

```
@Override
public void bergerak()
{
    System.out.println("Berjalan dengan KAKI, \"Tap..tap..\"");
}
```

10. With same way _ like when you create a Cat class , create an Animal class new named **Fish** and make the code like in the picture below .

```
public class Ikan extends Hewan{

    @Override
    public void bergerak() {
        System.out.println("Berenang dengan SIRIP, \"wush..wush..\"");
    }

}
```

11. Next , create a regular class new named class **Person** . _ This class is the class that becomes user from *abstract class* Animals that have made before . Type in the Person class , these lines of code as below . _

```
public class Orang
{
    private String nama;
    private Hewan hewanPeliharaan;

    public Orang(String nama)
    {
        this.nama = nama;
    }

    public void peliharaHewan(Hewan hewanPeliharaan)
    {
        this.hewanPeliharaan = hewanPeliharaan;
    }

    public void ajakPeliharaanJalanJalan()
    {
        System.out.println("Namaku " + this.nama);
        System.out.println("Hewan peliharaanku berjalan dengan cara: ");
        this.hewanPeliharaan.bergerak();
        System.out.println("-----------------------------------");
    }
}
```

12. Lastly , do it a *Play Class* new inside _ the same *packages* . Name the new class the with **program** class name . Type inside it like the code in below .

```
public class Program
{
    public static void main(String[] args)
    {
        Kucing kucingKampung = new Kucing();
        Ikan lumbaLumba = new Ikan();

        Orang ani = new Orang("Ani");
        Orang budi = new Orang("Budi");

        ani.peliharaHewan(kucingKampung);
        budi.peliharaHewan(lumbaLumba);

        ani.ajakPeliharaanJalanJalan();
        budi.ajakPeliharaanJalanJalan();
    }
}
```

Code Program:

```
public abstract class Hewan {
    private int umur;

    protected Hewan() {
        this.umur=0;
    }
    public void bertambahUmur() {
        this.umur+=1;
    }
    public abstract void bergerak();
}
public class Ikan extends Hewan{
    @Override
    public void bergerak() {
        System.out.println("Berjalan dengan SIRIP, \"Wush.. Wush..\"");
    }
}
public class Kucing extends Hewan {
    @Override
    public void bergerak() {
        System.out.println("Berjalan dengan KAKI, \"Tap.. Tap..\"");
    }
}
public class Orang {
    private String nama;
    private Hewan hewanPeliharaan;

    public Orang(String nama) {
        this.nama=nama;
    }

    public void peliharaanHewan(Hewan hewanPeliharaan) {
        this.hewanPeliharaan=hewanPeliharaan;
    }

    public void ajakPeliharaanBerjalan() {
        System.out.println("Namaku "+this.nama);
        System.out.println("Hewan peliharaanku berjalan dengan cara :");S
        this.hewanPeliharaan.bergerak();
        System.out.println("-----------------------------------------------");
    }
```

```
}
public class Program {
    public static void main(String[] args) {
        Kucing kucingKampung=new Kucing();
        Ikan lumbaLumba=new Ikan();

        Orang ani=new Orang("Ani");
        Orang budi=new Orang("Budi");

        ani.peliharaanHewan(kucingKampung);
        budi.peliharaanHewan(lumbaLumba);

        ani.ajakPeliharaanBerjalan();
        budi.ajakPeliharaanBerjalan();

    }
}
```

13. Run the class with method click right in the Program class then choose **Run File** (Shift + F6).
14. Watch and observe the result !

Output:

```
Namaku Ani
Hewan peliharaanku berjalan dengan cara :
Berjalan dengan KAKI, "Tap.. Tap.."
--------------------------------------------------
Namaku Budi
Hewan peliharaanku berjalan dengan cara :
Berjalan dengan SIRIP, "Wush.. Wush.."
--------------------------------------------------
```

15. **Question discussion:**
    may I if a class that *extends* something *abstract class* no implement *abstract* method in class _ parent ? Prove it!
    Answer: If a class that extends an abstract class does not implement the abstract method in its parent class, it will not work because an error will occur. Because the class that extends this is a concrete class which must implement all the abstract methods of the superclass or its parent class. So the class that extends must also have an abstract method contained in the abstract superclass.
    Proving if abstract method is not used on extend parent class would be an error

Question

1. Give explanation related about the above program
   Answer:

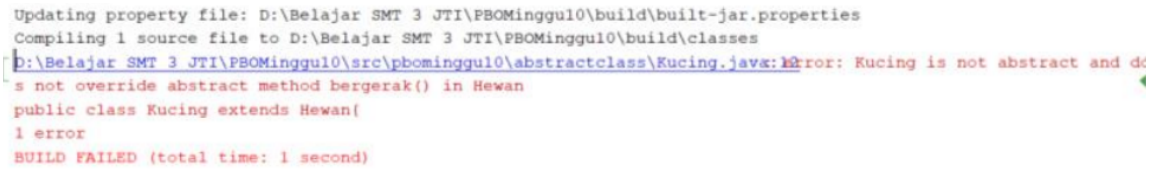
- The program starts from creating an abstract class Animal as the superclass of the Cat class and Fish class. Within the Animal class there is an Animal constructor, where this.animal = 0;. Then there is the void method increaseAge where this.age += 1;. And there is also a moving abstract void method.
- Then enter the Cat class. Where in the Cat class extends into the Animal class by overriding the void method move, which will print "Move Using FEET" Tap... Tap...".
- Then enter the Fish class. Where in the Fish class extends into the Animal class by overriding the move void method, which will print "Swim Using FIN, \"wush... wush...\".

- The last in the Person class has a private attribute Pets and a private String name. Where in the Person constructor is the name parameter. Then in the void pet() method
- used to link the Animal class with the Person class. And lastly, the invitePeliharaanJalanJalan() method, which functions to display the names of people, and also the movements that animals make when walk

2. Show results compile program and provide explanation short if *method* **move ( )** changed Becomes *abstract methods* !
   Answer:

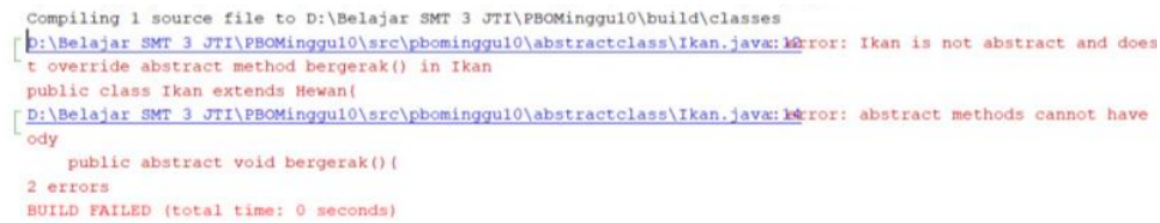
   An error will occur because the abstract method can only be used in abstract classes. Another reason for the abstract method cannot be a body.

3. Show results compile program and provide explanation short if no conducted *overriding* to **move ( )** *method*
   Answer:



   Programs in the Cat class that do not do Overriding will cause an error. The error occurs because, the class that implements the move() method and extends the abstract class must declare its abstract method as well.

4. Show results compile program and provide explanation short if *abstract method* **move ( )** declared _ in Fish Class
   Answer:



   An error occurs when the program code is executed. This error occurs because the move() method on the derived class and not from the abstract class itself cannot create an abstract method in the Fish class which is not abstract.