LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
PERTEMUAN 10

2141762056
Margaretha Violina Putri Purnomo
SIB 2F /11

DAFTAR ISI

1. Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

1. Menjelaskan maksud dan tujuan penggunaan Abstract Class;
2. Menerapkan Abstract Class di dalam pembuatan program.

2. Link Github

https://github.com/MargarethaViolinaPutri/PBO-

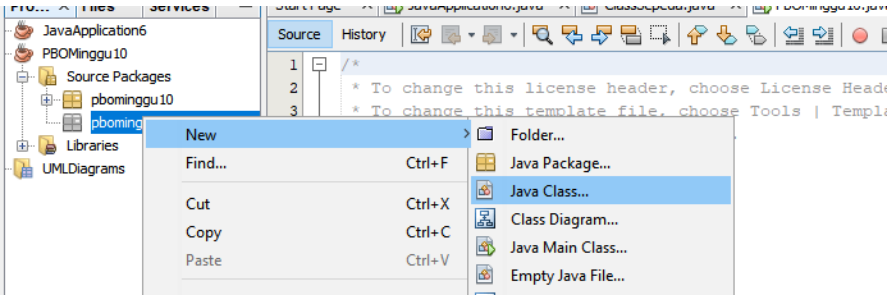
3. Praktikum

1. Buatlah sebuah project baru di NetBeans dengan nama **PBOMinggu10**

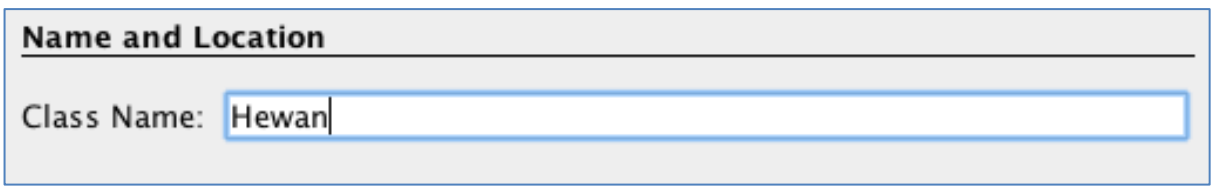
2. Pada package **pbominggu10**, tambahkan package baru dengan cara klik kanan nama package → New → Java Package…

3. Beri nama package tersebut dengan nama **abstractclass**. Semua class yang dibuat pada percobaan 1 ini **diletakkan pada package yang sama**, yaitu *package abstractclass* ini,

4. Pada package baru tersebut tambahkan *class* baru.



5. Beri nama *class* baru tersebut, yaitu class **Hewan**.



6. Pada *class* Hewan tersebut, ketikkan kode berikut ini.

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

    protected Hewan(){
        this.umur=0;
    }
    public void bertambahUmur(){
        this.umur +=1;
    }
    public abstract void bergerak();
}
```

Class Hewan tersebut adalah class abstract berisi property dan method biasa, ditambah sebuah *method abstract* bernama **bergerak()**. *Method* tersebut didepannya terdapat kata kunci **abstract** dan tidak memiliki badan fungsi. Method ini nantinya akan di-*override* oleh *class* mana saja yang menjadi *class* turunan dari class Hewan tersebut.

7. Dengan cara yang sama, buatlah class dengan nama **Kucing** yang meng-*extend* class Hewan. Di dalam class Kucing tersebut, setelah Anda menuliskan kode seperti di bawah, maka akan muncul ikon lampu peringatan. Klik lampu tersebut dan kemudian pilih **implement all abstract methods**.

```
public class Kucing extends Hewan{

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

8. Maka akan secara otomatis dibuatkan fungsi yang meng-*override* fungsi *abstract* **bergerak()** yang ada pada class hewan.

```
public class Kucing extends Hewan{

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

}
```

9. Ubahlah badan fungsi tersebut dengan mengganti kode didalamnya menjadi seperti berikut.

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

10. Dengan cara yang sama seperti ketika Anda membuat class Kucing, buatlah class Hewan baru bernama **Ikan** dan buatlah kodenya seperti pada gambar dibawah.

```
public class Ikan extends Hewan {

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

11. Selanjutnya, buatlah class biasa baru yang bernama class **Orang**. Class ini adalah class yang menjadi pengguna dari *class abstract* Hewan yang sudah dibuat sebelumnya. Ketikkan pada class Orang tersebut, baris-baris kode seperti di bawah.

```
11  public class Orang {
12      private String nama;
13      private Hewan hewanPeliharaan;
14
15      public Orang(String nama){
16          this.nama = nama;
17      }
18      public void peliharaHewan(Hewan hewanPeliharaan){
19          this.hewanPeliharaan = hewanPeliharaan;
20      }
21      public void ajakPeliharaanJalanJalan(){
22          System.out.println("Namaku " + this.nama);
23          System.out.println( x: "Hewan peliharaanku berjalan dengan cara: ");
24          this.hewanPeliharaan.bergerak();
25          System.out.println( x: "----------------------------------------");
26      }
27  }
```

12. Terakhir, buatlah sebuah *Main Class* baru di dalam *package* yang sama. Beri nama class baru tersebut dengan nama class **Program**. Ketikkan didalamnya seperti kode di bawah ini.

```
11    public class Program {
12        public static void main(String[] args){
13            Kucing kucingKampung = new Kucing();
14            Ikan lumbaLumba = new Ikan();
15
16            Orang ani = new Orang( nama: "Ani");
17            Orang budi = new Orang( nama: "Budi");
18
19            ani.peliharaHewan( hewanPeliharaan: kucingKampung);
20            budi.peliharaHewan( hewanPeliharaan: lumbaLumba);
21
22            ani.ajakPeliharaanJalanJalan();
23            budi.ajakPeliharaanJalanJalan();
24        }
25    }
26
```

13. Jalankan class tersebut dengan cara klik kanan pada class Program kemudian pilih **Run File** (Shift + F6).
14. Perhatikan dan amati hasilnya!

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @
Namaku Ani
Hewan peliharaanku berjalan dengan cara:
Berjalan dengan KAKI, "Tap..tap.."
------------------------------------------
Namaku Budi
Hewan peliharaanku berjalan dengan cara:
Berenang dengan SIRIP, "wush..wush.."
------------------------------------------
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
Total time:  2.552 s
Finished at: 2022-11-17T13:56:13+07:00
------------------------------------------------
```

15. Pertanyaan diskusi:

    Bolehkah apabila sebuah class yang meng-*extend* suatu *abstract class* tidak mengimplementasikan *method abstract* yang ada di class induknya? Buktikan!

Jawab :

Hasilnya akan error karena bila suatu class meng-extends haruslah ada implementasi method abstract yanag ada di kelas induk.sekalipun ada method lain di class extends, tapi akan eror juga jika tidak ada method abstract di induk.

Misal kita hapus method bergerak pada class Kucing dan Ikan dimana artinya class Kucing dan Ikan tidak mengimplementasikan method abstract di class induk, hasilnya akan dibawah ini :

```
public class Kucing extends Hewan{
12
13  }
```

```
public class Ikan extends Hewan {
12
13  }
```

Output :

```
-----------------------< com.mycompany:Prak_PBO >-----------------------
Building Prak_PBO 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Prak_PBO ---
Exception in thread "main" java.lang.ExceptionInInitializerError
        at PBOMinggu10.Program.main(Program.java:13)
Caused by: java.lang.RuntimeException: Uncompilable code - PBOMinggu10.Kucing is not abstract and does not override abstract method bergerak()
        at PBOMinggu10.Kucing.<clinit>(Kucing.java:1)
        ... 1 more
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
    at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)
    at org.apache.commons.exec.DefaultExecutor.execute (DefaultExecutor.java:166)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:982)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:929)
    at org.codehaus.mojo.exec.ExecMojo.execute (ExecMojo.java:457)
    at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo (DefaultBuildPluginManager.java:137)
    at org.apache.maven.lifecycle.internal.MojoExecutor.doExecute2 (MojoExecutor.java:370)
    at org.apache.maven.lifecycle.internal.MojoExecutor.doExecute (MojoExecutor.java:351)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:215)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:171)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:163)
    at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject (LifecycleModuleBuilder.java:117)
    at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject (LifecycleModuleBuilder.java:81)
    at org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder.build (SingleThreadedBuilder.java:56)
```

Pertanyaan

1. Berikan penjelasan terkait tentang jalannya program diatas

Jawab : Pada class hewan yang merupakan class abstract memiliki method abstract bergerak dimana method pada class abstract diimplementasikan dalam class yang mengextends class induk yaitu di class Kucing dan Ikan. Terdapat class yang berdiri sendiri yaitu class Orang dimana bertindak sebagai pengguna dari class abstract atau class hewan.

2. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika *method* **bergerak()** diubah menjadi *method abstract*!

Jawab : Karena class Kucing dan class Ikan bukan class abstract dimana mereka hanya kelas biasa yang meng- extends class abstract. Jadi mereka tidak boleh memiliki method abstract.

```
public class Kucing extends Hewan{
12
13      @Override
15      public abstract void bergerak(){
16          System.out.println("Berjalan dengan KAKI, \"Tap..tap..\"");
17      }
18  }
```

```
public class Ikan extends Hewan {
12
13      @Override
15      public abstract void bergerak(){
16          System.out.println("Berenang dengan SIRIP, \"wush..wush..\"");
17      }
18  }
```

```
----------------------< com.mycompany:Prak_PBO >----------------------
Building Prak_PBO 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Prak_PBO ---
Error: Unable to initialize main class PBOMinggu10.Program
Caused by: java.lang.ClassFormatError: Code attribute in native or abstract methods in class file PBOMinggu10/Kucing
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
    at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)
    at org.apache.commons.exec.DefaultExecutor.execute (DefaultExecutor.java:166)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:982)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:929)
    at org.codehaus.mojo.exec.ExecMojo.execute (ExecMojo.java:457)
    at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo (DefaultBuildPluginManager.java:137)
```

3. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika tidak dilakukan *overriding* terhadap *method* **bergerak()**

Jawab : Jika tidak dilakukan overriding maka class Kucing dan Ikan akan error. Karena kedua class ini adalah sub class dari hewan (super classnya). Karena class yang mengexstends harus melakukan overriding.

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

```
12  public class Ikan extends Hewan {
13
    }
```

Output :

```
----------------------< com.mycompany:Prak_PBO >----------------------
Building Prak_PBO 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Prak_PBO ---
Exception in thread "main" java.lang.ExceptionInInitializerError
        at PBOMinggu10.Program.main(Program.java:13)
Caused by: java.lang.RuntimeException: Uncompilable code - PBOMinggu10.Kucing is not abstract and does not override abstract method bergerak()
        at PBOMinggu10.Kucing.<clinit>(Kucing.java:1)
        ... 1 more
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
    at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)
    at org.apache.commons.exec.DefaultExecutor.execute (DefaultExecutor.java:166)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:982)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:929)
    at org.codehaus.mojo.exec.ExecMojo.execute (ExecMojo.java:457)
    at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo (DefaultBuildPluginManager.java:137)
    at org.apache.maven.lifecycle.internal.MojoExecutor.doExecute2 (MojoExecutor.java:370)
    at org.apache.maven.lifecycle.internal.MojoExecutor.doExecute (MojoExecutor.java:351)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:215)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:171)
    at org.apache.maven.lifecycle.internal.MojoExecutor.execute (MojoExecutor.java:163)
    at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject (LifecycleModuleBuilder.java:117)
    at org.apache.maven.lifecycle.internal.LifecycleModuleBuilder.buildProject (LifecycleModuleBuilder.java:81)
    at org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder.build (SingleThreadedBuilder.java:56)
```

4. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika *abstract method* **bergerak()**yang dideklarasikan dalam Class Ikan

Jawab : Terjadi error karena tidak boleh mendeklarasikan method abstract dibukan class abstract. Deklarasi abstrack method hanyalah boleh di class abstract bukan class yang mengxtends abstract class.

```java
public class Kucing extends Hewan{

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

```java
10  */
    public class Ikan extends Hewan {
12
13      @Override
        public abstract void bergerak(){
15          System.out.println("Berenang dengan SIRIP, \"wush..wush..\"");
16      }
17  }
```

Output :

```
-----------------------< com.mycompany:Prak_PBO >-----------------------
Building Prak_PBO 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Prak_PBO ---
Error: Unable to initialize main class PBOMinggu10.Program
Caused by: java.lang.ClassFormatError: Code attribute in native or abstract methods in class file PBOMinggu10/Ikan
Command execution failed.
org.apache.commons.exec.ExecuteException: Process exited with an error: 1 (Exit value: 1)
    at org.apache.commons.exec.DefaultExecutor.executeInternal (DefaultExecutor.java:404)
    at org.apache.commons.exec.DefaultExecutor.execute (DefaultExecutor.java:166)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:982)
    at org.codehaus.mojo.exec.ExecMojo.executeCommandLine (ExecMojo.java:929)
    at org.codehaus.mojo.exec.ExecMojo.execute (ExecMojo.java:457)
    at org.apache.maven.plugin.DefaultBuildPluginManager.executeMojo (DefaultBuildPluginManager.java:137)
```