LAPORAN

PRAKTIKUM PEMROGRAMAN

ABSTRACT CLASS DAN INTERFACES

Oleh:

Muchamad Lutfi Maftuh

NIM. 19537141023

PROGRAM STUDI TEKNOLOGI INFORMASI

JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA DAN INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI YOGYAKARTA

2020

A. LATIHAN

1. Menggunakan abstract class. Terdapat ***class Hewan, class Karnivora, dan TryAbstractClass1.***

```
abstract class Hewan {
    String nama;
    public abstract void habitatHewan();
    public void namaHewan() {
        System.out.println("\nMethod di dalam abstract class Hewan");
        System.out.println("Nama hewan : " + nama);
    }
}
```

```
class Karnivora extends Hewan {
    String habitat;

    public void habitatHewan() {
        System.out.println("\nMethod di dalam abstract class Karnivora");
        System.out.println("Habitat hewan " + habitat);
    }
}
```

```
public class TryAbstractClass1 {
    public static void main(String[] args) {
        System.out.println("\n************************************************");
        System.out.println("\n\tMENERAPKAN PENGGUNAAN ABSTRACT CLASS #1");
        System.out.println("\n************************************************");
        Karnivora singa = new Karnivora();
        singa.nama = "Singa";
        singa.habitat = "Darat";
        singa.namaHewan();
        singa.habitatHewan();
    }
}
```

   a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

```
************************************************

    MENERAPKAN PENGGUNAAN ABSTRACT CLASS #1

************************************************

Method di dalam abstract class Hewan
Nama hewan : Singa

Method di dalam abstract class Karnivora
Habitat hewan Darat
[Finished in 2.9s]
```

   b. Berikan penjelasan terkait jalannya program ini!
      - **Class Hewan** merupakan kelas abstract yang berisi data field String nama, method abstract void habitatHewan(), dan method void namaHewan().
      - **Class Karnivora** meng-extends Hewan yang berarti harus mengoverride method abstract void habitatHewan(). Karnivora memiliki data field habitat.
      - **Class TryAbstractClass1** membuat object Karnivora singa. Kemudian menginisiasi dengan perintah singa.nama (di Class Hewan sesuai polymorphism) dan singa.habitat (di Class Karnivora). Method namaHewan() dan habitatHewan pun dipanggil untuk menampilkan kalimat seperti pada gambar point a.
   c. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika method namaHewan() diubah menjadi method abstract!

```
.\Hewan.java:4: error: abstract methods cannot have a body
    public abstract void namaHewan() {
                         ^
.\Karnivora.java:1: error: Karnivora is not abstract and does not override abstract method
namaHewan() in Hewan
class Karnivora extends Hewan {
^
2 errors
[Finished in 2.5s]
```

Akan terjadi error yang pertama, method namaHewan tidak boleh memiliki body {} karena abstract. Kedua, Karnivora yang meng-extends Hewan tidak meng-override namaHewan().

d. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika tidak dilakukan overriding terhadap abstract method habitatHewan()!

```
.\Karnivora.java:1: error: Karnivora is not abstract and does not override abstract method
habitatHewan() in Hewan
class Karnivora extends Hewan {
^
1 error
[Finished in 2.7s]
```

Terjadi error karena method abstract harus di-override oleh siapapun yang meng-extends-nya

e. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika abstract method habitatHewan() dideklarasikan dalam class Karnivora!

```
.\Karnivora.java:4: error: method habitatHewan() is already defined in class Karnivora
    public void habitatHewan() {
                ^
.\Karnivora.java:1: error: Karnivora is not abstract and does not override abstract method
habitatHewan() in Karnivora
class Karnivora extends Hewan {
^
2 errors
[Finished in 2.6s]
```

Terjadi error karena terdapat dua method yang sama namanya. Juga error karena Karnivora bukan abstract dan abstract method harus di-override.

2. Menggunakan interface. Terdapat ***interface Operasi, class Hitung, class TryHitung (Buat sendiri).***

```
public interface Operasi
{
    double kons_pi = 3.14;
    String kons_panjang = " cm";
    void kelilingLingkaran(double radius) ;
    void kelilingPersegi ();
}
```

```
class Hitung implements Operasi
{
    double lingkaran, persegi;
    double sisi = 5;
    public void kelilingLingkaran(double radius)
    {
        System.out.println("\nMenghitung Keliling Lingkaran");
        System.out.println("Nilai radius = " + radius + kons_panjang) ;
        lingkaran = kons_pi * 2 * radius;
        System.out.println("Keliling Lingkaran = " + lingkaran + kons_pa
    }
    public void kelilingPersegi()
    {
        System.out.println("\nMenghitung Keliling Persegi");
        System.out.println("Nilai sisi = " + sisi + kons_panjang) ;
        persegi = 4 * sisi;
        System.out.println("Keliling Persegi = " + persegi + kons_panjan
    }
}
```

```
public class TryHitung {
    public static void main(String[] args) {
        Hitung hitung = new Hitung();
        hitung.kelilingLingkaran(10);
        hitung.kelilingPersegi();
    }
}
```

a. Buatlah sebuah class baru yang berisi method main untuk menjalankan programtersebut! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

```
Menghitung Keliling Lingkaran
Nilai radius = 10.0 cm
Keliling Lingkaran = 62.800000000000004 cm

Menghitung Keliling Persegi
Nilai sisi = 5.0 cm
Keliling Persegi = 20.0 cm
[Finished in 3.1s]
```

b. Berikan penjelasan terkait jalannya program ini!
   - **Interface Operasi** memiliki data fields konst_pi dan kons_panjang. Juga memiliki method abstract kelilingLingkaran(double radius) dan kelilingPersegi().
   - **Class Hitung** meng-implements Operasi yang berarti harus meng-override method abstract diatas.
   - **Class TryHitung** mencoba membuat Object Hitung dan memanggil method yang sudah di-override.

c. Tunjukkan hasil kompilasi dan eksekusi program kemudian berikan penjelasan singkat jika method kelilingPersegi() dikosongkan!

```
Menghitung Keliling Lingkaran
Nilai radius = 10.0 cm
Keliling Lingkaran = 62.800000000000004 cm
[Finished in 2.8s]
```

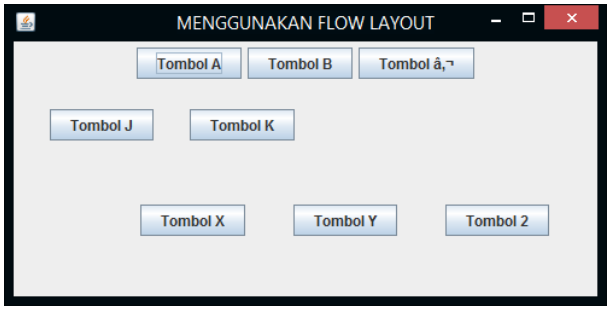
Method kelilingPersegi tetap dipanggil tetapi tidak melakukan apa-apa karena kosong.

d. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika method kelilingPersegi()dihapus dan tidak dipanggil dalam method main!

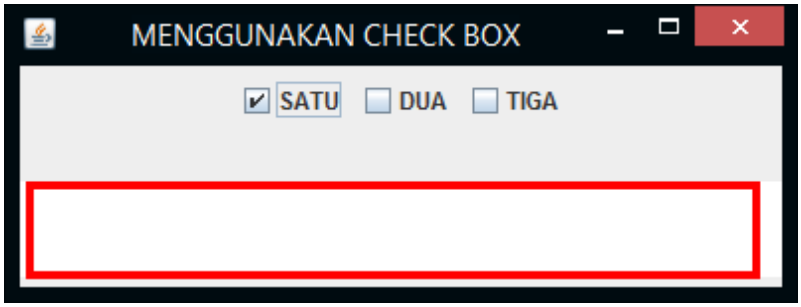
```
.\Hitung.java:1: error: Hitung is not abstract and does not override abstract method
kelilingPersegi() in Operasi
class Hitung implements Operasi
^
1 error
[Finished in 2.6s]
```

Terjadi error dikarenakan method abstract kelilingPersegi() yang harusnya di-override oleh Hitung, tidak di-override.

3. Menggunakan class FlowLayout.
   a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



   b. Berikan penjelasan terkait jalannya program ini!
   Mencoba menggunakan FlowLayout yang akan diset ke sebuah JPanel.
   c. Jelaskan fungsi perintah
      - `p2.setLayout(new FlowLayout (FlowLayout.LEFT,30,20));`
      Perintah diatas digunakan untuk meng-set layout pada p2 (JPanel) dengan FlowLayout berparameter:
        - **FlowLayout.LEFT** : align left.
        - **30** : HorizontalGap
        - **20** : VerticalGap
      - `p3.setLayout(new FlowLayout (FlowLayout.RIGHT,40,50));`
      Perintah diatas digunakan untuk meng-set layout pada p2 (JPanel) dengan FlowLayout berparameter:
        - **FlowLayout.RIGHT** : align right.
        - **40** : HorizontalGap
        - **50** : VerticalGap
4. Menggunakan class JCheckBox.
   a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



b. Berikan penjelasan terkait jalannya program ini!
Program tersebut mencoba membuat checkbox dengan JCheckBox. Ketika terjadi perubahan pada checkbox **harusnya** muncul text pada textArea.

c. Jelaskan fungsi perintah

- `tArea = new JTextArea(3,20);`
  Perintah diatas digunakan menginisiasi object tArea dengan constructor JTextArea. Angka 3 adalah tingginya dan angka 20 adalah panjangnya.
- `tArea.setEditable(false);`
  Perintah diatas digunakan untuk meng-set agar tArea tidak bisa diedit.

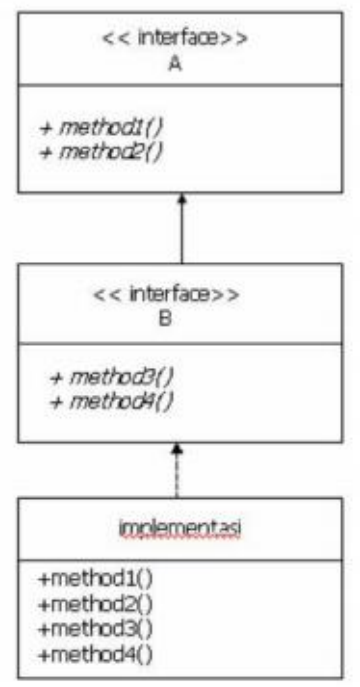
5. Menggunakan class JRadioButton.

a. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!

b. Berikan penjelasan terkait jalannya program ini!
Program diatas menggunakan JRadioButton dan ketika terjadi ActionEvent textarea akan menampilkan pilihan.

B. TUGAS PRAKTIKUM

1. Impementasikan UML class diagram dibawah!




```java
interface InterfaceA {
    void method1();
    void method2();
}

interface InterfaceB {
    void method3();
    void method4();
}

class Implement implements InterfaceA, InterfaceB {
    public void method1() {
        System.out.println("Implementasi method1");
    }

    public void method2() {
        System.out.println("Implementasi method2");
    }

    public void method3() {
        System.out.println("Implementasi method3");
    }

    public void method4() {
        System.out.println("Implementasi method4");
    }
}

public class praktik {
    public static void main(String[] args) {
        Implement impl = new Implement();
        impl.method1();
        impl.method2();
        impl.method3();
        impl.method4();
    }
}
```

Hasil run:

```
Implementasi method1
Implementasi method2
Implementasi method3
Implementasi method4
[Finished in 3.2s]
```

C. TUGAS RUMAH

1. Berikan argumentasi anda tentang perbedaan antara Interface dan Abstract? Sertakan contoh program untuk memperkuat argumen anda!

Abstract Class	Interface
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Modifiersnya harus dituliskan sendiri.	Tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah **public** dan ***abstract***.
Bisa mendeklarasikan **constant** dan **instance variable**.	Hanya bisa mendeklarasikan **constant**. Secara implisit variable yang dideklarasikan di interface bersifat **public**, **static** dan **final**.
Method boleh bersifat **static**.	Method tidak boleh bersifat **static**
Met hod boleh bersifat **final**.	Method tidak boleh bersifat **final**.
Suatu abstact class hanya bisa meng-*extend* satu abstract class lainnya.	Suatu interface bisa meng-*extend* satu atau lebih interface lainnya.
Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-**implement** beberapa interface.	Suatu interface hanya bisa meng-*extend* interface lainnya. Dan tidak bisa meng-*implement* class atau interface lainnya.

2. Jelaskan kondisi yang tepat untuk penggunaan Abstract dan Interface!
   Abstract untuk mendefinisikan secara luas sifat-sifat dari class tertinggi pada hirarki OOP, dan gunakan subclassnya (turunannya/child class) untuk melengkapi deskripsi method dari class abstract.
   Interface untuk mendefinisikan method standar yang sama dalam class-class yang berbeda. Sekali kita telah membuat kumpulan definisi method standar(abstrak), kita

dapat menulis method tunggal(spesifik) untuk memanipulasi semua class-class yang mengimplementasikan interface tsb.

3. Berikan capaian pemahaman anda dalam bentuk persentase (0%-100%) tentang praktikum pertemuan ini! Tambahkan argumentasi singkat mengenai teknik pembelajaran yang telah dilaksanakan selama praktikum!
   Pemahaman 90%.