

# P 10 Abstract Clas

## A. Tujuan

1. Mahasiswa mampu menerapkan konsep dasar, mendeklarasikan dan menggunakan *abstract class*.
2. Mahasiswa mampu menerapkan konsep dasar, mendeklarasikan, menggunakan, dan mewariskan *interface*.
3. Menggunakan *class* `FlowLayout`, `JCheckBox`, dan `JRadioButton`.

## B. Dasar Teori

1. Konsep *abstract class*.

*Abstract class* atau kelas abstrak adalah kelas yang terletak di posisi tertinggi dalam hierarki *class*. *Class* ini tidak dapat diinstansiasi karena masih bersifat abstrak. *Class* ini hanya berisi variabel umum dan deklarasi *method* tanpa detail penggunaannya (*abstract method*). Selanjutnya *class* yang menjadi turunan dari *abstract class* ini yang akan menentukan detail penggunaan *method* tersebut.

2. Deklarasi dan penggunaan *abstract class*.

*Abstract Class* dideklarasikan dengan cara sebagai berikut:

```
public abstract class NamaKelasAbstrak {  
    //deklarasi variabel dan abstract method  
    //definisi method tidak abstrak  
}
```

*Abstract Class* digunakan dengan cara sebagai berikut:

```
class NamaKelasPengguna extends NamaKelasAbstrak {  
    //penggunaan variabel dan method tidak abstrak  
    //pendefinisian abstract method  
}
```

3. Konsep *interface*.

*Interface* merupakan suatu mekanisme dalam Java yang memungkinkan untuk berbagi konstanta atau menentukan bentuk *method* yang dapat digunakan oleh sejumlah *class*. Sebuah *class* dapat mengimplementasikan lebih dari satu *interface*. Di dalam

*interface*, penentu akses untuk definisi konstanta adalah *public static final*. Sedangkan penentu akses untuk deklarasi *abstract method* adalah *public abstract*. Kedua penentu akses ini tidak harus dituliskan secara eksplisit dalam kode program karena Java akan menggunakan kedua penentu akses ini sebagai penentu akses *default* bila sekiranya tidak ditulis secara eksplisit.

#### 4. Deklarasi, penggunaan, dan pewarisan *interface*.

*Interface* dideklarasikan dengan cara sebagai berikut:

```
public interface NamaInterface {  
    //definisi konstanta  
    //deklarasi abstract method  
}
```

*Interface* digunakan dengan cara sebagai berikut:

```
class NamaKelasPengguna implements NamaInterface {  
    //penggunaan konstanta  
    //pendefinisian abstract method  
}
```

*Interface* diwariskan dengan cara sebagai berikut:

```
public interface Interface_Y extends Interface_X {  
    //definisi konstanta  
    //deklarasi abstract method  
}
```

#### 5. Perbedaan antara *abstract class* dan *interface*.

Saat deklarasi *abstract class* dan *interface* digunakan *modifier public*. Hal ini bertujuan agar *abstract class* dan *interface* tersebut dapat digunakan secara bebas oleh *class* lain yang membutuhkannya. Sekilas *interface* tampak mirip dengan *abstract class* karena *abstract class* juga menentukan bentuk *method* untuk *subclass*, tetapi ada beberapa perbedaan antara *interface* dan *abstract class*, yaitu:

- a. *Abstract class* dapat mengandung *abstract method* maupun *method* tidak abstrak. Sedangkan *interface* hanya boleh mengandung *abstract method*.
- b. *Abstract class* dapat mendeklarasikan variabel *instant*. Sedangkan *interface* hanya dapat mendeklarasikan konstanta.
- c. *Abstract class* digunakan oleh *class* lain melalui pewarisan dengan kata kunci *extends*. Sedangkan sebuah *interface* diimplementasikan ke dalam suatu *class* dengan menggunakan kata kunci *implements*.

## 6. Pengenalan *class* *FlowLayout*, *JCheckBox*, dan *JRadioButton*

*Class FlowLayout* merupakan *class* dari *package* *java.awt* yang digunakan untuk mengatur letak komponen dengan urutan dari kiri ke kanan dan dari atas ke bawah. Pengaturan rata kiri, rata kanan dan rata tengah dapat dilakukan dengan melibatkan konstanta *LEFT*, *RIGHT* dan *CENTER*.

*Class JCheckBox* merupakan *class* dari *package* *javax.swing* yang digunakan untuk membuat komponen berupa *check box* atau kotak pilihan yang dapat dipilih oleh *user*. Pada umumnya, saat menggunakan *check box*, *user* dapat memilih lebih dari satu pilihan yang telah disediakan.

*Class JRadioButton* merupakan *class* dari *package* *javax.swing* yang digunakan untuk membuat komponen berupa *radio button* atau tombol radio yang dapat dipilih oleh *user*. Pada umumnya, saat menggunakan *radio button*, *user* hanya dapat memilih satu dari pilihan yang telah disediakan. Hal ini dapat diatur dengan bantuan *class* *ButtonGroup* dari *package* *javax.swing*.

## C. Latihan

### 1. Menggunakan *abstract class*.

Tulislah dan simpan kode program di bawah ini:

Kode program *abstract class* Hewan:

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

Kode program *class* Karnivora:

```
class Karnivora extends Hewan
{
    String habitat;
    public void habitatHewan() //pendefinisian abstract method
    {
        System.out.println("\n    Method di dalam class
Karnivora");
        System.out.println("Habitat hewan : "+habitat);
    }
}
```

Kode program *class* TryAbstractClass1:

```
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();

    }
}
```

- Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!
- Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika *method* `namaHewan()` diubah menjadi *method abstract*!
- Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika tidak dilakukan *overriding* terhadap *abstract method* `habitatHewan()`!
- Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika *abstract method* `habitatHewan()` dideklarasikan dalam *class* `Karnivora`!

## 2. Menggunakan *interface*.

Tulislah dan simpan kode program di bawah ini:

Kode program *interface* Operasi:

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

Kode program *class* Hitung:

```
class Hitung implements Operasi
{
    double lingkaran, persegi;
    double sisi=5;

    public void kelilingLingkaran(double radius)
    {
        System.out.println("\n    Menghitung Keliling Lingkaran");
        System.out.println("Nilai radius      = "
        +radius+kons_panjang);
        lingkaran = kons_pi*2*radius;
        System.out.println("Keliling Lingkaran = "
        +lingkaran+kons_panjang);
    }

    public void kelilingPersegi()
    {
        System.out.println("\n    Menghitung Keliling Persegi");
        System.out.println("Nilai sisi      = "
        +sisi+kons_panjang);
        persegi = 4*sisi;
        System.out.println("Keliling Persegi  = "
        +persegi+kons_panjang);
    }
}
```

- Buatlah sebuah *class* baru yang berisi *method main* untuk menjalankan program tersebut! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!
- Tunjukkan hasil kompilasi dan eksekusi program kemudian berikan penjelasan singkat jika *method* `kelilingPersegi ()` dikosongkan!
- Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika *method* `kelilingPersegi ()` dihapus dan tidak dipanggil dalam *method main*!

### 3. Menggunakan *class* FlowLayout.

Tulislah dan simpan kode program di bawah ini:

```
import javax.swing.*;
import java.awt.*;
public class TryFlowLayout extends JFrame
{
    public TryFlowLayout()
    {
        super("MENGGUNAKAN FLOW LAYOUT");
        setSize(500,250);

        JPanel p1 = new JPanel();
        p1.setLayout(new FlowLayout());

        //meletakkan button pada panel
        p1.add(new JButton("Tombol A"));
        p1.add(new JButton("Tombol B"));
        p1.add(new JButton("Tombol C"));

        JPanel p2 = new JPanel();
        p2.setLayout(new FlowLayout(FlowLayout.LEFT,30,20));

        //meletakkan button pada panel
        p2.add(new JButton("Tombol J"));
        p2.add(new JButton("Tombol K"));

        JPanel p3 = new JPanel();
        p3.setLayout(new FlowLayout(FlowLayout.RIGHT,40,50));

        //meletakkan button pada panel
        p3.add(new JButton("Tombol X"));
        p3.add(new JButton("Tombol Y"));
        p3.add(new JButton("Tombol Z"));

        //meletakkan dan mengatur posisi panel pada frame
        add("North", p1);
        add("Center", p2);
        add("South", p3);

        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args)
    {
        TryFlowLayout frame = new TryFlowLayout();
    }
}
```

- Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!



c. Jelaskan fungsi perintah `p2.setLayout(new FlowLayout`  
`(FlowLayout.LEFT,30,20));` dan `p3.setLayout(new FlowLayout`  
`(FlowLayout.RIGHT,40,50));` !

4. Menggunakan *class* JCheckBox.

Tulislah dan simpan kode program di bawah ini:

```
import javax.swing.*;
import java.awt.event.*;

public class TryCheckBox extends JFrame implements ActionListener
{
    //membuat objek yang akan digunakan
    private JCheckBox cb1, cb2, cb3;
    private JTextArea tArea;

    public TryCheckBox()
    {
        super("MENGGUNAKAN CHECK BOX");
        setSize(400,150);

        //menugaskan objek yang dibuat dari class JCheckBox
        cb1 = new JCheckBox("SATU", false);
        cb2 = new JCheckBox("DUA", false);
        cb3 = new JCheckBox("TIGA", false);

        tArea = new JTextArea(3,20);
        tArea.setEditable(false);

        JPanel p1 = new JPanel();
        p1.add(cb1);
        p1.add(cb2);
        p1.add(cb3);

        JPanel p2 = new JPanel();
        p2.add(tArea);
    }
}
```

```

        add("North", p1);
        add("South", p2);

        //memberikan mekanisme event handling pada check box
        cb1.addActionListener(this);
        cb2.addActionListener(this);
        cb3.addActionListener(this);

        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    //method untuk menampilkan status check box
    public void tampilkanInfo()
    {
        String sCB1, sCB2, sCB3;
        sCB1 = "Status SATU : "+cb1.isSelected();
        sCB2 = "\nStatus DUA : "+cb2.isSelected();
        sCB3 = "\nStatus TIGA : "+cb3.isSelected();

        tArea.setText(sCB1 + sCB2 + sCB3);
    }

    //method untuk memberikan aksi bila status check box berubah
    public void actionPerformed(ActionEvent e)
    {
        tampilkanInfo(); //memanggil method tampilkanInfo()
    }

    public static void main(String[] args)
    {
        TryCheckBox frame = new TryCheckBox();
    }
}

```

- Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!
- Jelaskan fungsi perintah `tArea = new JTextArea(3,20);` dan `tArea.setEditable(false);`!



5. Menggunakan *class* JRadioButton.

Tulislah dan simpan kode program di bawah ini:

```
import javax.swing.*;
import java.awt.event.*;

public class TryRadioButton extends JFrame implements
ActionListener
{
    //membuat objek yang akan digunakan
    private JRadioButton rb1, rb2, rb3;
    private ButtonGroup bg;
    private JTextArea tArea;

    public TryRadioButton()
    {
        super("MENGGUNAKAN RADIO BUTTON");
        setSize(400,150);

        //menugaskan objek yang dibuat dari class JRadioButton
        rb1 = new JRadioButton("SATU", false);
        rb2 = new JRadioButton("DUA", false);
        rb3 = new JRadioButton("TIGA", false);

        //membuat hanya bisa memilih satu radio button
        bg = new ButtonGroup();
        bg.add(rb1);
        bg.add(rb2);
        bg.add(rb3);

        tArea = new JTextArea(3,20);
        tArea.setEditable(false);

        JPanel p1 = new JPanel();
        p1.add(rb1);
        p1.add(rb2);
        p1.add(rb3);

        JPanel p2 = new JPanel();
        p2.add(tArea);

        add("North", p1);
        add("South", p2);
    }
}
```

```

//memberikan mekanisme event handling pada radio button
rb1.addActionListener(this);
rb2.addActionListener(this);
rb3.addActionListener(this);

setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

//method untuk memberikan aksi bila radio button dipilih
public void actionPerformed(ActionEvent e)
{
    tArea.setText("Pilihan Anda : "+e.getActionCommand());
}

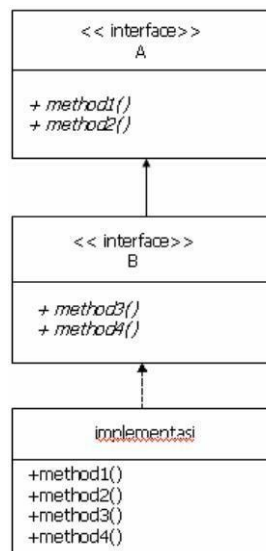
public static void main(String[] args)
{
    TryRadioButton frame = new TryRadioButton();
}
}

```

- Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!
- Berikan penjelasan terkait jalannya program ini!

#### D. Tugas Praktikum

- Impementasikan UML *class diagram* dibawah!



Berikut merupakan contoh member kelas implementasi:

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

Tampilan yang diharapkan(contoh):

```
Implementasi method 1.....
Implementasi method 2.....
Implementasi method 3.....
Implementasi method 4.....
```

## 2. *Let's make something useful!*

```
import java.awt.*; import
java.awt.event.*;

class ImplementsDemo extends Frame implements WindowListener {
    Label label;
    ImplementsDemo(String title)
    { super(title);
      label = new Label("Close the frame.");
      this.addWindowListener(this);
    }

    void launchFrame() {
        setSize(300,300);
        setVisible(true);
    }

    /* Method dibawah merupakan implementasi dari
    Interface 'WindowListener' */

    public void windowActivated(WindowEvent e) {
    }
    public void windowClosed(WindowEvent e) {
    }
    public void windowClosing(WindowEvent e)
    { setVisible(false);
      System.exit(0);
    }

    public void windowDeactivated(WindowEvent e) {
```

```

    }
    public void windowDeiconified(WindowEvent e) {
    }
    public void windowIconified(WindowEvent e) {
    }
    public void windowOpened(WindowEvent e) {
    }

    public static void main(String args[]) {
        ImplementsDemo cf = new ImplementsDemo("Close Window
Example");
        cf.launchFrame();
    }
}

```

3. Dari percobaan praktikum 2 tambahkan ucapan selamat datang dengan menggunakan kotak dialog (misal: JOptionPane) ketika Window dibuka!

### E. Tugas Rumah

1. Berikan argumentasi anda tentang perbedaan antara *Interface* dan *Abstract*? Sertakan contoh program untuk memperkuat argumen anda!
2. Jelaskan kondisi yang tepat untuk penggunaan *Abstract* dan *Interface*!
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!