**Nama : Elvin Eireni Silambi**

**Npm : 0620101027**

**UAS PBO**

**1.**

**a. Access Modifier (public, private, protected)**

- public : menentukan sifat akses ke semua member yang mengikutinya

- private : berfungsi untuk menentukan sifat akses ke semua member yang mengikutinya menjadi memiliki sifat yang tidak dapat di akses dari manapun

- protected : berfungsi untuk menentukan sifat akses semua member yang mengikutinya, sehingga memiliki sifat yang tidak dapat diakses dari luar class

**b. Inheritence**

mewarisi atribut dan metode dari satu kelas ke kelas lainnya. mengelompokkan "konsep pewarisan" ke dalam dua kategori:

subclass (anak) - kelas yang mewarisi dari kelas lain

superclass (induk) - kelas yang diwarisi

**c. Polymorphsm**

metode untuk melakukan tugas yang berbeda. Ini memungkinkan kita untuk melakukan satu tindakan dengan cara yang berbeda.

**d. Interface**

Cara lain untuk mencapai abstraksi di Java, adalah dengan antarmuka.

Antarmuka adalah "kelas abstrak" sepenuhnya yang digunakan untuk mengelompokkan metode terkait dengan badan kosong:

**e. Exception Handling (try catch**

mekanisme paling sederhana untuk menangani pengecualian.

untuk menentukan blok kode yang akan diuji untuk kesalahan saat sedang dieksekusi

**2.**

import java.util.Scanner;

public class himp{

Scanner input = new Scanner(System.in);

int[] Array1= new int[100];

int[] Array2= new int[100];

int JumlahArray1;

int JumlahArray2;

public void dataAnggota() {

System.out.print("Masukkan Jumlah Array 1 : ");

JumlahArray1 = input.nextInt();

System.out.println("Anggota Array 1");

for(int i=1;i<=JumlahArray1;i++){

System.out.print("Anggota "+i+" : ");

Array1[i]=input.nextInt();

}

System.out.print("Masukkan Jumlah Array 2 : ");

JumlahArray2 = input.nextInt();

System.out.println("Anggota Array 2");

for(int i=1;i<=JumlahArray2;i++){

System.out.print("Anggota "+i+" : ");

Array2[i]=input.nextInt();

}

}

public void tampilAnggota(){

System.out.print("Anggota Array 1 = { ");

for(int i=1;i<=JumlahArray1;i++){

System.out.print(Array1[i]+" ");

}

System.out.println("}");

System.out.print("Anggota Array 2 = { ");

for(int i=1;i<=JumlahArray2;i++){

System.out.print(Array2[i]+ " ");

}

System.out.println("}");

}

public void himpunanIrisan(){

System.out.print("Himpunan Irisan Array 1 dan 2 : { ");

for( int i=1;i<=JumlahArray2;i++){

boolean adaSama = false;

for(int j=1;j<=JumlahArray1;j++){

if(Array2[i]==Array1[j])

adaSama=true;

}

if(adaSama==true){

System.out.print(Array2[i]+" ");

}

}

System.out.println(" }");

}

public static void main(String args[]){

himp bilangan = new himp();

bilangan.dataAnggota();

bilangan.tampilAnggota();

bilangan.himpunanIrisan();

}

}

**3.**

|  |
| --- |
| public class DataClass {  int numberData; String wordsData;  public DataClass(int numberData, String **wordsData** ){  this.numberData = **numberData**;  **this.wordsData** = wordsData;  }  Public int getNumberData(){**int[]numberData**;}  public String getWordsData(){**String[]wordsData**;}  }  public interface ParentInterface {  public DataClass initData(int number, String words);  **return new DataClass(number, words);**  }  public class ParentClass implements **ParentInterface**{  public DataClass initData(int number, String words){  return new DataClass(number, words);  }  public String getDataString() { return “” ; }  }  public class childClass extends **ParentClass**{  DataClass childData ;    public void setData() {  childData = initData(1, “firstChild”);  }  @override  public String getDataString() { return childData.getWordsData(); }  }  public class secondChild extends **ParentClass**{  DataClass secondChildData ;    public void setData() {  secondChildData = initData(2, “secondChild”);  }  @override  public String getDataString() { return **secondChildData**.getWordsData(); }  }    public MainClass{  public static void main (String [] args){  // do polymorphsm here  **childClass**.objek ;  objek = new childClass();  objek.setData();  System.out.println(objek.getDataString());  objek = **new secondChild();** // change to secondChild  objek.setData();  System.out.println(objek.getDataString());  }  } |