Nama : Zidan Mulyana

Npm : 0620101019

UAS PBO

1. Sebutkan dan jelaskan menurut bahasa Anda sendiri istilah - istilah di bawah ini : (Knowledge Test : 30)

a. Access Modifier (public, private, protected)

b. Inheritence

c. Polymorphsm

d. Interface

e. Exception Handling (try catch)

2. Buatlah sebuah program dengan algoritma dibawah ini (Logic Test : 25)

|  |
| --- |
| Input  Array 1 : [1,3,5,7,8,9]  Array 2 : [1,2,4,5,8]  Output : 1,5,8 |

3. Perhatikan potongan code dibawah ini, lengkapi titik titik yang ada sebutkan apa saja konsep OOP yang digunakan diikuti nomor baris. (OOP Test : 45)

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

Jawaban :

1. **Acces modifier** : tujuan nya adalah  membatasi resource-resource mana saja yang dapat diakses oleh object tertentu

* Public : class atau variable yang dapat diakses oleh seluruh class
* Private : class atau variable yang hanya bisa di akses oleh class itu sendiri
* Protected : dapat diakses oleh subclass atau class lain asalkan didalam satu package yang sama

1. **Inheritence :** dalam inheritance dimana entitas dapat mempunyai banyak objek turunan
2. **Polymorphsm :** di mana class memiliki nama yang sama namun dapat memiliki method yang berbeda-beda.
3. **Interface :** digunakan secara metafora untuk perbatasan antara benda
4. **Exception Handling (try catch) :** mekanisme yang paling diperlukan dalam menangani error

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());  }  } |