RAFI MAULANA
0620101032

JAWABAN UAS

1.

a. Access Modifier adalah sebuah “hak akses” yang diberikan kepada sebuah variabel/method/class dengan tujuan untuk menjaga integritas dari data tersebut ketika ingin diakses object lain. Hak akses sendiri diberikan oleh pembuat program sebagai salah satu bentuk implementasi dari OOP itu sendiri.
b. Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat 'menurunkan' property dan method yang dimilikinya kepada class lain. Konsep inheritance digunakan untuk memanfaatkan fitur 'code reuse' untuk menghindari duplikasi kode program.
c. Polimorfisme terbagi menjadi dua suku kata yaitu, Poly yang berarti banyak dan Morfisme yang berarti bentuk. Dalam ilmu sains, Polimorfisme (polymorphism) adalah sebuah prinsip dalam biologi di mana organisme atau spesies memiliki banyak bentuk serta tahapan (stages).
d. User Interface adalah tampilan visual sebuah produk yang menjembatani sistem dengan pengguna (user). Tampilan UI dapat berupa bentuk, warna, dan tulisan yang didesain semenarik mungkin. ... Tampilan UI ini diterapkan pada sistem operasi, aplikasi, website, maupun blog.
e. Exception Handling merupakan mekanisme yang paling diperlukan dalam menangani error yang terjadi pada saat runtime (program berjalan) atau yang lebih dikenal dengan sebutan runtime error.

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());
       }
}
```