Soal Latihan UAS OOP

Nama : Bernardus Ignasio

NIM : 2602063913

Kelas : LB01

Essay

1. What is the difference between Overloading and Overriding? Provide example code for them.

   Jawab:

Overloading	Overriding
Banyak function dengan nama yang sama dalam super class	Subclass memodifikasi/menspesifikasikan function/method yang sudah disediakan pada parent class
Function pada subclass memiliki kesamaan nama dan perbedaan jumlah, parameter, dan return type	Function pada subclass harus memiliki kesamaan nama, jumlah, parameter, dan return type dengan parent class
Merupakan compile-time polymorphism	Merupakan run-time polymorphism
Memerlukan/Tidak memerlukan inheritance	Memerlukan Inheritance
Dapat digunakan pada private & final methods	Tidak dapat digunakan pada private dan final methods
Digunakan untuk meningkatkan readability program	Digunakan untuk menngimplementasikan hal khusus dari method yang telah disediakan parent class
Poor Performance	Better Performance

   Example Code:

   - Overriding

*Animal.java dog.java

```
1 package no1;
2
3 public class Animal {
4
5     public void sound() {
6         System.out.println("Here's my sound ");
7         System.out.println("I'm an animal");
8     }
9 }
10
```

```java
package no2;

public interface ability {
    public abstract void move();
    public abstract void attack();
    public abstract void buyitems();
}
```

- Overloading

2. Interfaces are usually used to group common behaviors of class / object. Write an example by mentioning the class name and common behaviors between classes! Use Video Games context for the example.

   Jawab:

   Contoh dari interface yang mengelompokkan common behaviors dari class/object dapat dilihat dengan video games. Contoh yang digunakan berupa video games bergenre mobile arena (ML/AOV/DOTA,etc) dimana terdapat class berupa player, enemy, dan npc. Player, enemy dan npc dalam permainan mobile arena dapat melakukan beberapa kemampuan, seperti menyerang, jalan, dan membeli item. Kemampuan ini dapat dikelompokkan dengan interface berupa ability.

```java
package no1;

public class multiply {
    public int multiplying(int a, int b) {
        return a*b;
    }
    public double multiplying(double a,int b,float c) {
        return a*b*c;
    }
}
```

   Interface ability yang dimiliki tiap class akan diimplements, dan dirubah menyesuaikan dengan peran tiap class (overriding)

*player.java npc.java enemy.java *ability.java

```java
package no2;

public class enemy implements ability {

    @Override
    public void move() {
        // TODO Auto-generated method stub
        System.out.println("enemy is moving forward");
    }

    @Override
    public void attack() {
        // TODO Auto-generated method stub
        System.out.println("Enemy is attacking turret");
        System.out.println("Enemy is attacking npc");
    }

    @Override
    public void buyitems() {
        // TODO Auto-generated method stub
        System.out.println("Enemy is buying sword");
    }

}
```

```java
package no2;

public class npc implements ability {

    @Override
    public void move() {
        // TODO Auto-generated method stub
        System.out.println("npc is moving forward");

    }

    @Override
    public void attack() {
        // TODO Auto-generated method stub
        System.out.println("Friendly npc is attacking enemy");
        System.out.println("Enemey npc is attacking player");
    }

    @Override
    public void buyitems() {
        // TODO Auto-generated method stub
        System.out.println("Npc is buying shield");
    }

}
```

player.java npc.java enemy.java *ability.java

```java
package no2;

public class player implements ability{

    @Override
    public void move() {
        // TODO Auto-generated method stub
        System.out.println("Player is moving backward");
    }

    @Override
    public void attack() {
        // TODO Auto-generated method stub
        System.out.println("Player is attacking turret");
        System.out.println("Player is attacking npc");
    }

    @Override
    public void buyitems() {
        // TODO Auto-generated method stub
        System.out.println("Player is buying gun");
    }

}
```

3. What is the difference between Abstract Class and Interface? Provide example code for them.

Jawab:

Abstract Class	Interface
Variable yang digunakan bebas	Variable yang digunakan harus berupa public static final
Constructor dapat dipanggil subclass	Tidak ada constructor
Method yang digunakan bebas (bisa abstract & concrete method)	Method harus berupa abstract method
Diextended dengan keyword extends	Diimplementasikan dengan keyword implements
Dapat mengimplement interface	Tidak dapat mengextend abstract class
Dapat mengimplements multiple interfaces dan mengextend satu java class	Dapat mengimplements multiple interfaces

Example code:

- Abstract Class

```
1  package no3;
2
3  public abstract class animal implements movement{
4      private String name;
5      private String sound;
6      private Integer age;
7      private Integer type;
8
9      public abstract void displayidentity();
10     public void introduction() {
11         System.out.println("Hello my name is "+ name);
12     }
13     S
14 }
```

- Interface

```
1  package no3;
2
3  public interface movement{
4      public abstract void eat();
5      public abstract void drink();
6      public abstract void play();
7  }
```

4. Create a generic method to sort data in Ascending order.

```
1.      public <T extends Comparable<T>> void
   sort(T[] data) {
2.          for(int i=0;i<data.length-1;i++) {
3.              for(int j=i+1;j<data.length;j++) {
4.                  if(data[i].compareTo(data[j])>0)
   {
5.                      T temp = data[i];
6.                      data[i]=data[j];
7.                      data[j]=temp;
8.                  }
9.              }
10.             }
11.         }
```

5. Thread in java has several states. Explain those states!
   - New: State ketika membuat hal baru dalam thread class, maka tercipta thread baru yang belum berjalan dan belum di run/execute
   - Running: State dimana thread sedang dijalankan/siap untuk dijalankan menunggu waktunya

- Suspended: State dimana aktivitas dalam thread distop secara temporary dan dapat dijalankan kembali sesuai dengan aktivitas terakhir
- Blocked: State dimana thread diblokir saat menunggu resource untuk mengurangi jumlah queue, saat resource didapatkan, maka akan berpindah ke running state
- Terminated: State dimana thread dihentikan secara keseluruhan dikarenakan keseluruhan thread sudah dijalankan ataupun terdapat eror. Setelah dihentikan, thread tidak dapat diresume.