IT-22041

~~Ans~~ +

Ques1: When to use interface and when to use abstract class. Develop a story and write codes to explain.

⇒ Story: Smart Garage System:

In the city of Dhaka, a new smart Garage system is being developed. The goal is to manage different kinds of vehicles than enter and exit the garage. This vehicles can be:

1. Bikes
2. Cars
3. Electronic cars
4. Electric scooters

All vehicles should have -

1. Ability to start
2. Ability to stop

Some vehicles are electric and have

1. Battery level
2. Ability to charge battery

Example: (Interface)
Use the interface when we want a common behaviour contract for all types of vehicles - cars, bikes, electric scooters etc.

```
public class interface vehicle{
        void start();
        void stop();
    }
```

Abstract class: Electric vehicle
Use an abstract class when some types (electric) share common code or fields.

```
public abstract class Electricvehicle implements vehicle{
    protected int batterylevel = 100;
    public void changebattery(){
        batterylevel = 100;
 System. out. println ("Battery fully changed");
 }
 public abstract void start();
  public abstract void stop();
}
```

class : ElectricCar

```
public class ElectricCar extends Electricvehicle {
    public void start (){
System.out.println("Electric car started. Battery: " +
                                   Batterylevel + "%");
     Battery level --=10; }
   public void stop (){
   system.out.println ("Electric car stopped");
   }
}
```

class: Bike (non-electric, implements interface directly)

```
public class Bike implements vehicle{
     public void start (){
    system.out.println ("Bike started");
    }
     public void stop(){
      system.out.println ("Bike stopped");
    }
  }
```

Main method:

```
public class Main{
public static void main (string[] args) {
vehicle bike= new bike();
  bike. start();
  bike. stop();
Electric car car= new Electriccar();
    car. start();
    car. stop();
    car. charge_battery();
 }
}
```

ques 2: Is invoking methods in interface slower than in abstract class?

⇒ No, not significantly.

In early JVM versions, method calls through interfaces used to be slightly slower because they required more indirection than class-based dispatch.

But in modern JVMs, method calls (interface or abstract) are treated equally efficiently by the runtime.

⇒ Interface example:

```
    public interface Device {
        void start();}

  public class printer Implements Device {
    public void start(){
    system.out.println ("printer Starting via interface");
   }
  }
```

Abstract class Example:

```
 public abstract class Machine{
   public void start(){
   system.out.println ("Machine starting via abstr
   class.");
  }
}
 public class Scanner extends Machine{
 }
```

Main class:

```
 public class Main {
  public static void main (String[] args){
 Device printer = new Printer();
  printer . start();
 Machine scanner = new scanner();
  scanner. start();}
                  }
```

Output:

```
 printer starting via Interface.
 Machine starting via abstract.
```

Ques3 : Abstract class vs Interface in Java.

Feature	Abstract class	Interface
purpose	partial abstraction (some methods with code)	Full abstraction (behaviour contract)
Method Types	can have abstract and non- abstract methods	can have abstract default and static method.
constructor	yes	No constructor allowed
Multiple inheritance	Not allowed.	allowed
Access modifiers	can use private, protected etc.	Methods are public by default.
Example	abstract class Animal{ void breathe() {}}	interface Flyable{ void fly();}.