Core Java 8 and Development Tools

Lesson 07 : Abstract Classes and Interfaces

Lesson Objectives

- After completing this lesson, participants will be able to:
  - Understand concept of Abstract classes and Interfaces
  - Default and static methods in interface
  - Differentiate between abstract classes and interfaces
  - Implement Runtime polymorphism

Abstract Class

- Provides common behavior across a set of subclasses
- Not designed to have instances that work
- One or more methods are declared but may not be defined, these methods are abstract methods.
- Abstract method do not have implementation
- Advantages:
  - Code reusability
  - Help at places where implementation is not available

Abstract Class (cont..)

- Declare any class with even one method as abstract as *abstract*
- Cannot be instantiated
- Cannot use *Abstract* modifier for:
  - Constructors
  - Static methods
- Abstract class' subclasses should implement all methods or declare themselves as *abstract*
- Can have concrete methods also

Demo

- Execute the Executor.java program

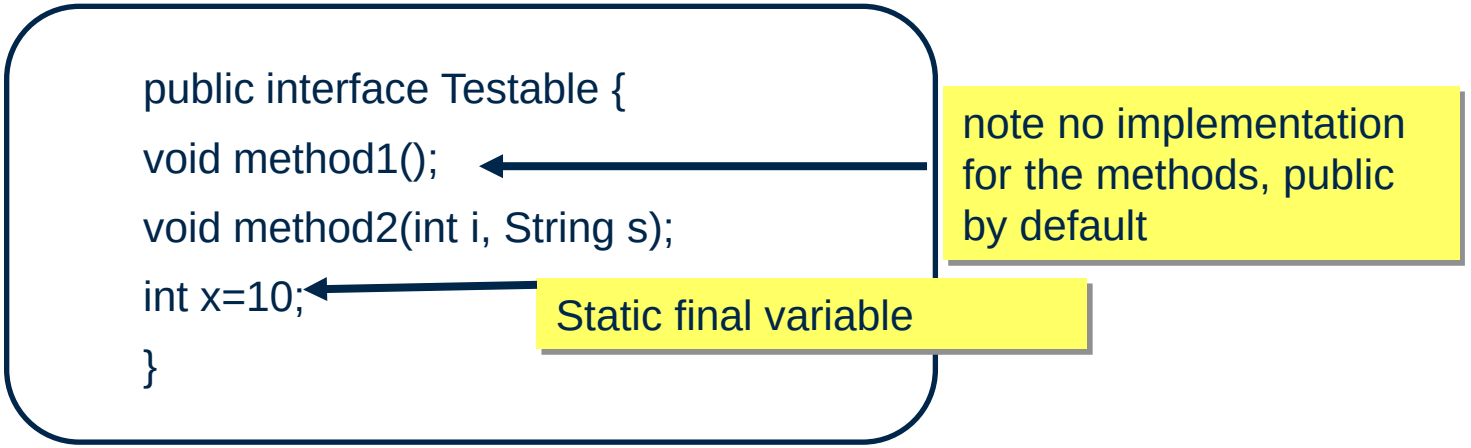
Interface

- Special kind of class which consist of only the constants and the method signatures.
- Approach also known as “programming by contract”.
- It’s essentially a collection of constants and abstract methods.
- It is used via the keyword “implements”. Thus, a class can be declared as follows:

```
class MyClass implements MyInterface{
   ...
}
```

What is Interface?

- A Java interface definition looks like a class definition that has only abstract methods, although the abstract keyword need not appear in the definition



Declaring and Using Interfaces

```
public interface SimpleCalc {
        int add(int a, int b);
        int i = 10;
 }
//Interfaces are to be implemented.
class Calc implements SimpleCalc {
        int add(int a, int b){
                return a + b;
        }
}
```

abstract method

By default is public, static and final

Interface - Rules

- Methods other than default and static in an interface are always public and abstract.
- Static methods in interface are always public .
- Data members in a interface are always public, static and final.
- Interfaces can extend other interfaces.
- A class can inherit from a single base class, but can implement multiple interfaces.

Abstract Classes and Interfaces

Abstract classes	Interfaces
Abstract classes are used only when there is a “is-a” type of relationship between the classes.	Interfaces can be implemented by classes that are not related to one another.
You cannot extend more than one abstract class.	You can extend more than one interface.
Abstract class can contain abstract as well as implemented methods.	Interfaces contain only abstract, default and static methods.
With abstract classes, you grab away each class’s individuality.	With Interfaces, you merely extend each class’s functionality.

Demo

- Execute the Interface Implementation.java program

Default Methods

- Starting from Java SE 8, interfaces can define default methods
- A default method in an interface is a method with implementation
- Use “default “ keyword in method signature to make it default.

```
interface xyz {
    default return-type method-name(argument-list) {
        ------------------
        ------------------
    }
}
```

- A class which implements the interface doesn’t need to implement default methods

Static Methods

- Along with the default methods an Interface can also have static methods
- The syntax of static method is similar to default method, where static keyword will replace default

```
interface xyz {
    static return-type method-name(argument-list) {
        ------------------
        ------------------
    }
}
```

Runtime Polymorphism

- Runtime polymorphism enables a method can do different things based on the object used for invoking method at runtime
- Runtime polymorphism is implemented by doing method overriding

```
class Parent {
    public String sayHello() {
        return “Hello from Parent”;
    }
}
class Child extends Parent {
    public String sayHello() {
        return “Hello from Child”;
    }
}
```

```
Parent object = new Child();
object.sayHello();
```

Accessing Implementations through Interface Reference

```
class sample implements TestInterface {
// Implement Callback's interface
public void interfacemthod() {
        System.out.println("From interface method"); }
public void noninterfacemthod() {
        System.out.println("From interface method"); }
 }
```

```
class Test {
   public static void main(String args[]) {
        TestInterface t = new sample();
         t.interfacemethod()      //valid
         t.noninterfacemethod()       //invalid }
```

Demo

- Lesson-7 Runtime polymorphism

Lab

- Lab 1: Abstract classes and Interfaces

Summary

- In this lesson, you have learnt about:
  - Abstract class
  - Interfaces
  - default methods
  - static methods on Interface
  - Runtime Polymorphism

Review Question

- Question 1: All variables in an interface are :
  - **Option 1:** Constant instance variables
  - **Option 2:** Static and final
  - **Option 3:** Constant instance variables
- Question 2: Will this code throw a compilation error?

```
interface sample
{
  int x;
}
```

  - **Option 1:** True
  - **Option 2:** False