Inheritence & Interface

CCVT-2018

Ravi Tomar

Assistant Professor(Senior Scale)
Centre for Information Technology
University of Petroleum & Energy Studies

February 16, 2018

Need of Inheritance

Let's say we wrote a class today with four methods for client 1. Tomorrow, we got a new client and needs the same four methods plus one additional method. If we don't have inheritance, what we need to do is simply create a new class, copy all the four methods from the old class and add the one new method, right? At this point, you might wonder why not just add the one new method in the old class? You can do, but why do you want to give additional functionality to client 1 when he doesn't need it? Moreover, if we get another client needing for 6 methods, we again have to update the class, and therefore we are impacting client 1 as well as client 2. Aren't we? This is a very bad practice.

Inheritance

A class that is derived from another class is called a subclass (also a derived class, extended class, or child class). The class from which the subclass is derived is called a superclass (also a base class or a parent class).

1. **Excepting Object**, which has no superclass, every class has one and only one direct superclass (single inheritance). In the absence of any other explicit superclass, every class is implicitly a subclass of Object.
2. Classes can be derived from classes that are derived from classes that are derived from classes, and so on, and ultimately derived from the topmost class, Object. Such a class is said to be descended from all the classes in the inheritance chain stretching back to Object.
3. A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass. s

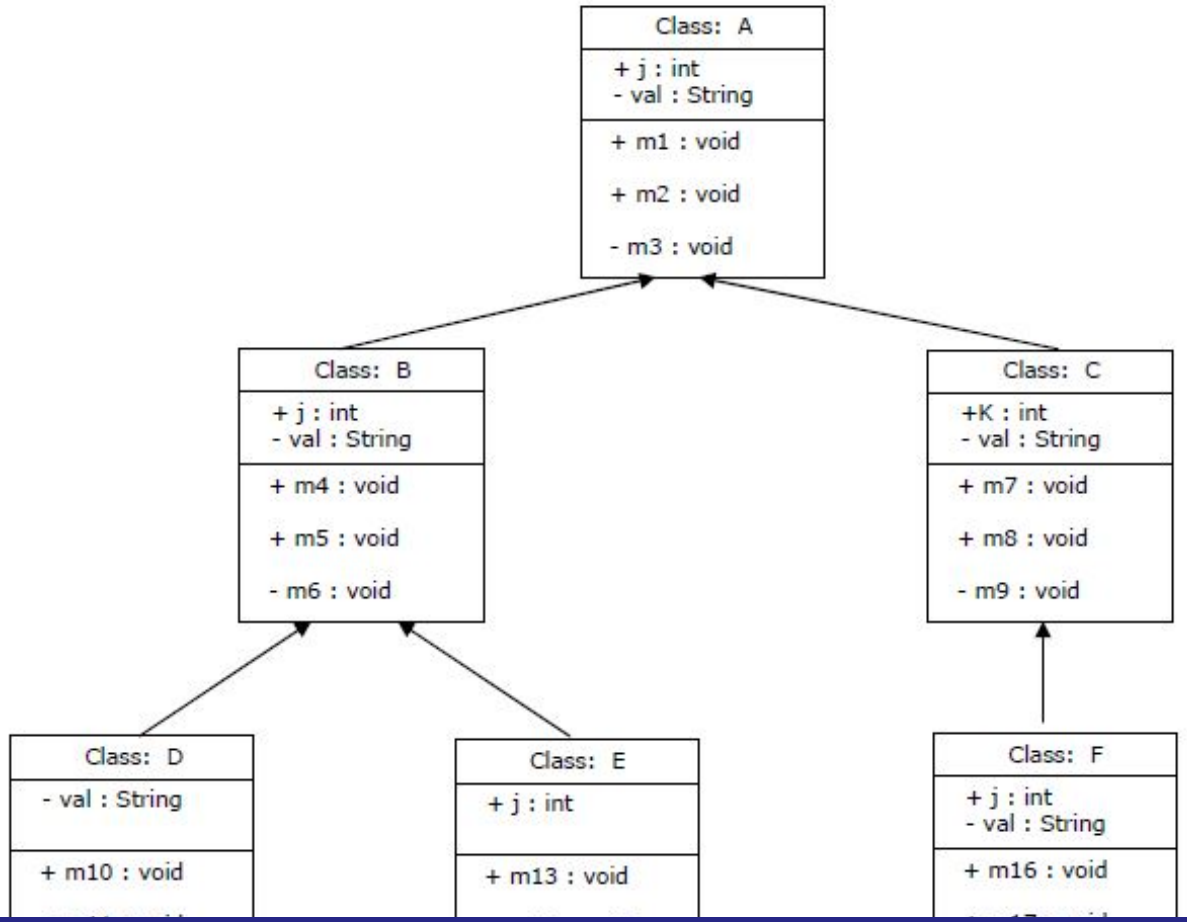
What can we do in subclass?

A subclass inherits all of the public and protected members of its parent, no matter what package the subclass is in. If the subclass is in the same package as its parent, it also inherits the package-private members of the parent. You can use the inherited members as is, replace them, hide them, or supplement them with new members:

1. The inherited fields can be used directly, just like any other fields.
2. You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not recommended).
3. You can declare new fields in the subclass that are not in the superclass.
4. The inherited methods can be used directly as they are.
5. You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it.
6. You can write a new static method in the subclass that has the same signature as the one in the superclass, thus hiding it.
7. You can declare new methods in the subclass that are not in the superclass.
8. You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword super.

Example

+ means Public, - means private



Question 1: I need to build a class with the methods m1, m2, m4, m5 from existing classes and a new method p1. Which of the following classes my new class must extend.

Question 2: I need to build a class with the methods m1, m2, m7, m8, m16, m17 from existing classes and a new method p2. Which of the following classes my new class must extend.

What we will learn now??

1. Method Overriding
2. Constructors Revisited
3. Abstract Methods
4. Abstract classes
5. Final methods
6. Final class
7. Final variables
8. covariant return type

Introduction

What & Why ?

- a mechanism to achieve fully abstraction
- represents IS-A relationship
- cannot be instantiated like **abstract class**
- an interface is a reference type, similar to a class, that can contain only:
  - constants(static & final),
  - method signatures,
  - default methods(from jdk1.8 onwards),
  - static methods(from jdk1.8 onwards),
  - and nested types.

Pointes to Ponder about interface!!

- Think of a contract.
- Application Programming interface.
- Example??
- Inheritence in Interface(extends Keyword)
- Using Interface as a Type.
- Why there is multiple inheritence supported by interface & not by class?
-

Marker Interface

An interface that have no member is known as marker or tagged interface. For example: Cloneable, Comparable,etc.They are used to provide some essential information to the JVM so that JVM may perform some useful operation.For eg.

interface samplemarker{
}

Cloneable Interface

Cloning: A clone is an exact copy of the original. In java, it essentially means the ability to create an object with similar state as the original object. The clone() method of Object class provides this functionality.

- By default, java cloning is 'field by field copy'
- If the class has only primitive data type members then a completely new copy of the object will be created and the reference to the new object copy will be returned.
- If the class contains members of any class type then only the object references to those members are copied and hence the member references in both the original object as well as the cloned object refer to the same object.

Cloning Contd...

Java infrastructure for cloning:

We need to do following two things:

- You must implement Cloneable interface.
- You must override clone() method from Object class. [Shouldn't clone() method should have been in Cloneable interface.]

What JavaDoc says about clone method.

protected native Object clone() throws CloneNotSupportedException;

1. x.clone() != x will be true. MOST IMPORTANT REQUIREMENT
2. x.clone().getClass() == x.getClass() will be true, but these are not absolute requirements.
3. x.clone().equals(x) will be true, this is not an absolute requirement.

Cloning Contd...

There are two types of cloning:
1. Shallow Cloning.
2. Deep Cloning.

Comparable Interface

Leaving for now...will discuss in Collections...Remember to ask Comparator also!!

Source Code of All Programs

https://github.com/ravitomar7/Java_Programming_CCVT.git