Abstraction & Interface Made Easy

Core Java

Shohaur Rahaman

2/21/2015

Home

Today I will try my level best to make you understand the most important core topic in Java “interface” and “abstraction” in java.

Abstraction : When I was starting java some topics were pretty confusing to me, some topics pained me much to got understand abstraction is one of them.

What is abstraction ?

Abstraction is a process to hiding the background details/ implementation and show only the essential things to users. May be you are confused what the heck I am talking about? Don’t worry, let me give you an example,

Have you a cell phone ? I am sure the answer is yes, at least one time you sent a text message to your friends, parents whatever it is, when you sent a text message to another client you didn’t knew the process how a text message passed one place to another , how it decrypted and how it encrypt. That means the cell phone abstract the process of sending message to you. That is called abstraction.

Java has the same kind of mechanism to hiding details.

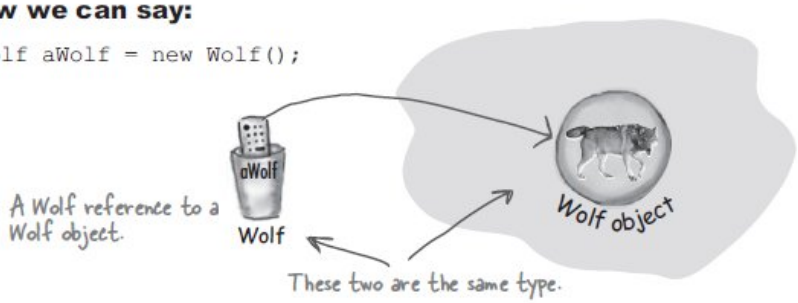
If you still confused about abstraction.....

Let me explain this with an example. Suppose there is a class Animal and there are few other classes like wolf, Dog and Hippo. These classes extends Animal class so basically they are having few common habits(methods in technically) which they are inheriting from Animal class. Now, if you have understood the above example then you would have been able to figure out that creating object of Animal class has no significance as you can’t judge that the new object of Animal class will represent which animal. Hence for such kind of scenarios we generally creates an abstract class and later concrete classes extends these classes and overrides their methods accordingly and can have their own methods as well.
Concrete class means that its ok you can create object that types of class.

Here is some visual of the explanation above:

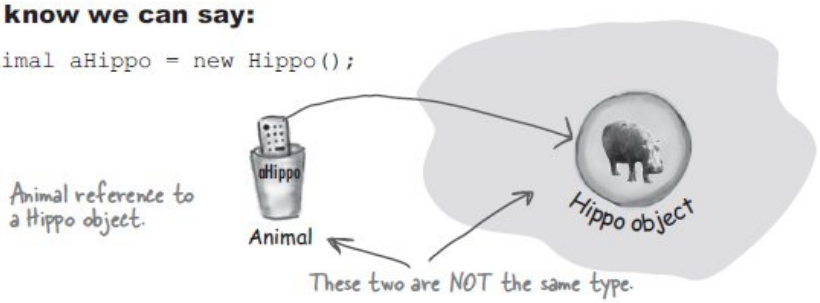
We know we can say:

```
Wolf aWolf = new Wolf();
```



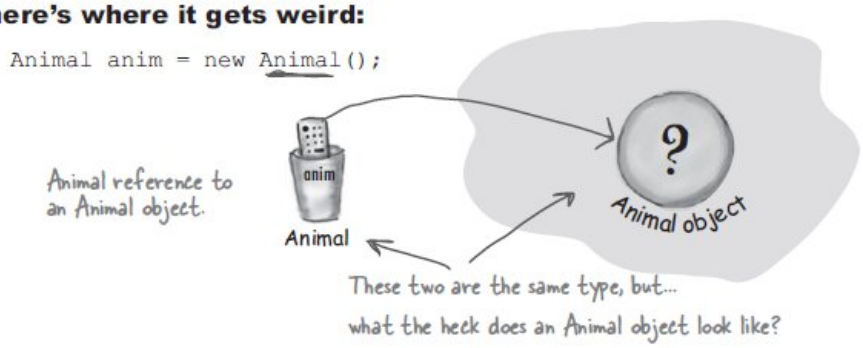
And we know we can say:

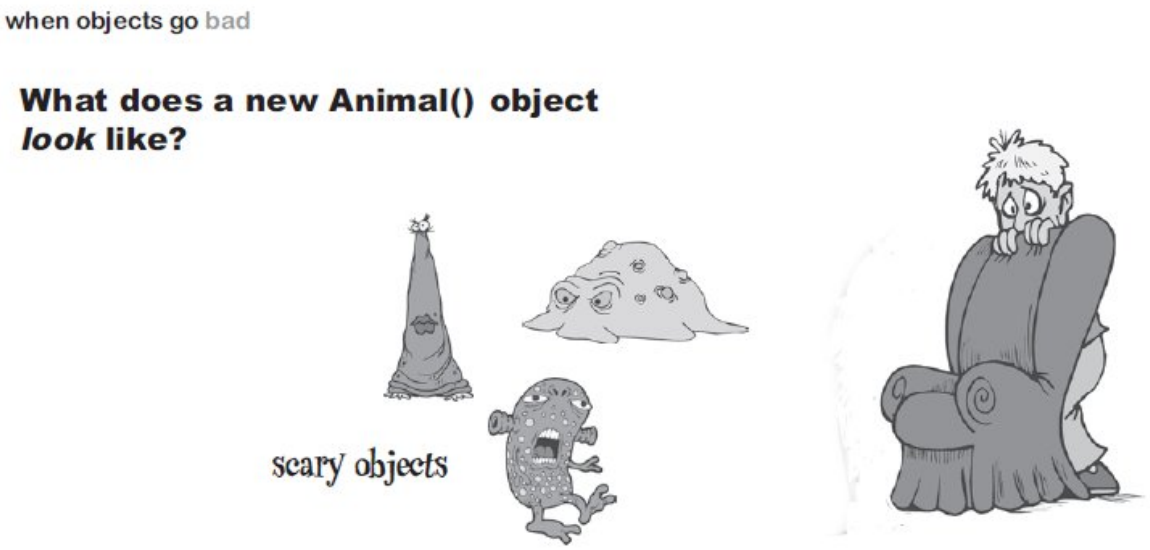
```
Animal aHippo = new Hippo();
```



But here's where it gets weird:

```
Animal anim = new Animal();
```





You cannot instantiate abstract class. some points to remember,

1. An abstract class has no use until unless it is extended by some other class.
2. If you declare an abstract method (example below) in a class then you must declare the class abstract as well. you can't have abstract method in a non-abstract class. It's vice versa is not always true: If a class is not having any abstract method then also it can be marked as abstract.
3. Abstract class can have non-abstract method (concrete) as well.

How to create abstract class :

```
abstract classname{
// do something bla bla ....
}
```

4. If you define any abstract method inside an abstract class, when you extends the abstract class, you must be provide a implementation of the abstract method to override the abstract method.

   // Here is a simple program :

```
package abstract_example;
/**
```

```
 *
 * @author Shohan-CSE
 */
abstract class animal{
   String name;

       void walk(){
       System.out.println("Its walking");
       }

       abstract void food();
}

class Hippo extends animal{

   void food(){
       System.out.println("I dont eat Rice !!");
   }
       void skin(){
           System.out.println("My skin is too hard !!");
       }
}

public class Abstract_example {

   public static void main(String[] args) {
      //     animal myani=new animal(); // abstract class cannot be instantiated
           Hippo motu=new Hippo();
           motu.food();
           motu.skin();
   }
}
```

5. An abstract class have constructor, data member method (normal or abstract) as well.

I am forgotten to tell what is abstract methd. Abstract method is method that has no body.

i.e : abstract myclass(); // defined an abstract class

If cannot define an abstract method inside any concrete class. You must implement all abstract method. Implementing abstract class just like overriding method (same method name, same return type same parameters indeed a carbon copy of the method).

This page is Intentionally Blank.

Abstract class cannot achieve 100% abstraction. May be you questioned why?

Because inside an an abstract class we can also define concrete method, data member and there are many high thought fact we did not focusing those.

Now we will focus on Interface using this mechanism we can achieve 100% abstraction.

Interface is a 100% abstract class. It contains only constants and method signatures. In other words it is a contract. It is a promise to provide certain behaviors and all classes which implement the interface guarantee to also implement those behaviors. To define the expected behaviors the interface lists a number of method signatures. Any class which uses the interface can rely on those methods being implemented in the runtime class which implements the interface.
This allows anyone using the interface to know what functionality will be provided without having to worry about how that functionality will actually be achieved.

The implementation details are hidden from the client, this is a crucial benefit of abstraction. Interface is an incomplete class, you cannot instantiate an incomplete class so, an interface can't be instantiated. It can be implemented by a class or extended by another interface.

How To Define:

An interface can be defined as the following:

```
public interface DriveCar {
void turnLeft();
void turnRight();
void moveBack();
void accelerate();
int carnumber=1425471;
}
```

The methods declared in an interface don't have method bodies. By default all the methods in an interface are public abstract. Similarly all the variables we define in an interface are

essentially constants because they are implicitly public static final. So, the following definition of interface is equivalent to the above definition.

```
public interface DriveCar {
public abstract void turnLeft();
public abstract void turnRight();
public abstract void moveBack();
public abstract void accelerate();
public static final int carnumber=1425471;
}
```

Note :Constructor is not supported in Interface;

// simple program to define and implement interface

```
/**
 *
 * @author Shohan-CSE
 */
interface mycar{
       void drive();
       void stop();
       int carnumber=1445895;
}
class toyota implements mycar{
   public void drive(){
       System.out.println("Drive car");
    }
       public void stop(){
           System.out.println("Stop car");
       }
     //   int carnumber=50000; // if i define another variable of same name it will hide
the previous varaible
       // see what happen just remove the comment of the variable just above
}
```

```
public class interface_example {
   public static void main(String args[]){

      toyota myc=new toyota();
      int a=myc.carnumber;
      System.out.println("Car number : "+a);
         myc.drive();
         myc.stop();
   }
}
```

Why Use Interfaces:

Interfaces act as APIs (Application Programming Interfaces). Let us take an example of an image processing company which writes various classes to provide the image processing functionalities. So, a nice approach will be creating interfaces, declaring the methods in them and making the classes implement them. In this way the software package can be delivered to the clients and the clients can invoke the methods by looking at the method signatures declared inside the interfaces. They won't see the actual implementation of the methods. As a result the implementation part will be a secret. Later on the company may decide to re-implement the methods in another way. But the clients are concerned about the interfaces only.

Interfaces provide an alternative to multiple inheritance. Java programming language does not support multiple inheritance. But interfaces provide a good solution. Any class can implement a particular interface and importantly the interfaces are not a part of class hierarchy. So, the general rule is extend one but implement many. A class can extend just one class but it can implement many interfaces. So, here we have multiple types for a class. It can be of the type of its super class and all the interfaces it implements.

Extending an interface:

Consider the following scenario. You have an interface A and several implementing classes. It defines 2 methods.

```
interface A{
int doThis();
int doThat();
```

```
}
```

Now suppose you want to add another method to the interface A:

```
interface A{
int doThis();
int doThat();
int doThisAndThat();
}
```

If you add the third method to the interface it will break the code because the implementing classes will no more be adhering to the contract. But we can avoid the problem if we create another interface and make it extend the previous interface.

```
interface APlusPlus extends A{
int doThisAndThat();
}
```

Now your users have the option to either use the old interface or upgrade to the new interface.

Note:

Any class that implements an interface must implement the methods declared in that interface plus all the methods that are present in the super interface.

If the implementing class is abstract it may choose to implement all, some or none of the methods declared in the interface. But a concrete subclass of the abstract class must implement all the non-implemented methods.

Multiple Inheritance using Interface

In java we cannot achieve multiple inheritance just the similar procedure how the other inheritance (single, multilevel..) inheritance achieved.

To solve this problem we use the power of interface,

What a Java class does have is the ability to implement multiple interfaces – which is considered a reasonable substitute for multiple inheritance, but without the added complexity. This is what a Java class that implements multiple interfaces would look like:

```
// Wolf and Canine are interfaces

public class Dog implements Wolf, Canine {
/*...*/
}
```

You can also have a class that extends one class, while implementing an interface – or even multiple interfaces. So, a class like this is perfectly legal in Java:

```
// Wolf is an interface
// Dog is a base class


// Poodle both extends from a class and implements
// an interface
public class Poodle extends Dog implements Wolf {
/*...*/
}
```

So, remember that although Java does not have multiple inheritance, a Java class *does* have the ability to implement multiple interfaces instead.

// Here is a simple Program :

```
/**
```

```
 *
 * @author Shohan-CSE
 */
/// Example of multiple Intewrface interface

interface car{
    void wheel();
    void seat();
}

interface engine{
    void v8();
}

    class yourcar implements car,engine{

     public void seat(){
            System.out.println("Seat");
        }
        public void wheel(){
           System.out.println("Wheel");

        }
            public void v8(){
                System.out.println("V8");
            }
    }

public class multiple_inheritance {
        public static void main(String args[]){

            yourcar toyota=new yourcar();
            toyota.seat();
            toyota.wheel();
            toyota.v8();

        }
}
```

Difference between Abstract Class and Interface in Java.

	abstract Classes	Interfaces
1	abstract class can extend only one class or one abstract class at a time	interface can extend any number of interfaces at a time
2	abstract class can extend from a class or from an abstract class	interface can extend only from an interface
3	abstract class can have both abstract and concrete methods	interface can have only abstract methods
4	A class can extend only one abstract class	A class can implement any number of interfaces
5	In abstract class keyword 'abstract' is mandatory to declare a method as an abstract	In an interface keyword 'abstract' is optional to declare a method as an abstract
6	abstract class can have protected , public and public abstract methods	Interface can have only public abstract methods i.e. by default
7	abstract class can have static, final or static final variable with any access specifier	interface can have only static final (constant) variable i.e. by default

Hope you enjoyed it, If you have any query mail me.

Contact : shohan4556@gmail.com