Abstraction

Abstraction

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.
- Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

- Abstract class
- Interface

Abstract Classes

- A class that is declared with abstract keyword, is known as abstract class in java.
- It can have abstract and non-abstract methods (method with body).
- An abstract class can not be **instantiated** (you are not allowed to create **object** of Abstract class).

Abstract class declaration

```
abstract class AbstractDemo{
   // Concrete method: body and braces
   public void myMethod(){
      //Statements here
   }
   // Abstract method: without body and braces
   abstract public void anotherMethod();
}
```

Abstract methods

Points to remember about abstract method:

1. Abstract method has no body.
2. Always end the declaration with a **semicolon**(;).
3. It must be overridden. An abstract class must be extended and in a same way abstract method must be overridden.
4. Abstract method must be in a abstract class.

Note: The class which is extending abstract class must override (or implement) all the abstract methods.

Object creation of abstract class is not allowed

```
abstract class Animal{
  public void sleep(){
    System.out.println("Zzzz");
  }
  abstract public void animalSound();
}
class Cat extends Animal{
  public void animalSound() {
  System.out.print("The cat says: meew meew");
  }
}
```

```
class Test{
  public static void main(String args[]){
//Can't create object of abstract class-error!
    Animal obj = new Animal();
    Cat obj = new Cat();
    obj.animalSound();
    obj.sleep();
  }
}
```

abstract class Example:

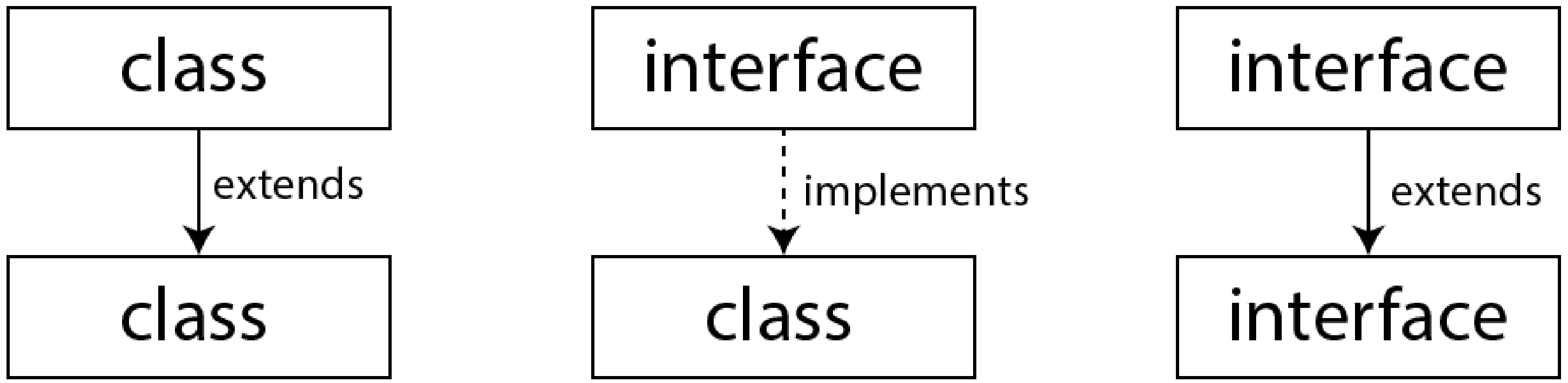
```
abstract class Demo1{
          public void disp1(){
                   System.out.println("Concrete method of abstract class");
          }
         abstract public void disp2();
}
class Demo2 extends Demo1{
           public void disp2() {
                   System.out.println("I'm overriding abstract method");
           }
           public static void main(String args[]){
                    Demo2 obj = new Demo2();
                    obj.disp2();
           }
}
```

What is interface

- An interface is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.
- An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts.
- A class describes the attributes and behaviors of an object. An interface contains behaviors that a class implements.

Declaring Interfaces:

```
interface NameOfInterface
{
            //Any number of final, static fields
           //Any number of abstract method declarations
}
```


class
extends
class
interface
implements
class
interface
extends
interface

An interface is different from a class in several ways, including:

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.

Interface Example:

```
interface Drawable {
        public void draw();
}
class Rectangle implements Drawable{
        public void draw()  {
                System.out.println(“drawing rectangle");
        }
        public static void main(String arg[])  {
                Rectangle d = new Rectangle();
                d.draw();
        }
}
```

What is the use of interfaces

- As stated above they are used for abstraction.
- Since methods in interfaces do not have body, they have to be implemented by the class before you can access them.
- The class that implements interface must implement all the methods of that interface.
- Also, java programming language does not support multiple inheritance, but using interfaces we can achieve this as a class can implement more than one interfaces.

Interface and Inheritance Example:

```
interface A{
    public void methodA();
}
interface B extends A{
    public void methodB();
}
interface C extends A{
    public void methodC();
}
```

```
class D implements B, C {
   public void methodA(){
       System.out.println("MethodA");
   }
   public void methodB(){
       System.out.println("MethodB");
   }
   public void methodC(){
       System.out.println("MethodC");
   }
   public static void main(String args[]){
       D obj1= new D();
       obj1.methodA();
       obj1.methodB();
       obj1.methodC();
    }
}
```

Output:

```
MethodA
MethodB
MethodC
```

<u>Remember two rules:</u>

1. If the class is having few abstract methods and few non-abstract methods: declare it as abstract class.
2. If the class is having only abstract methods: declare it as interface.

Multiple Inheritance by Interface

- If a class implements multiple interfaces, or an interface extends multiple interfaces, then it is known as multiple inheritance.

```
interface Printable{
                void  print();
}
interface Showable{
                void show();
}
class Test implements Printable, Showable{
                public void print(){
                    System.out.println(“Hello”);
                }
                public void show(){
                   System.out.println(“Welcome”);
                }
                public static void main(String[] args){
                                Test obj = new Test();
                                obj.print();
                                obj.show();
                }
}
```

Difference between abstract class and Interface

abstract Class	Interfaces
abstract class can extend only one class or one abstract class at a time	interface can extend any number of interfaces at a time
abstract class can have both abstract and non-abstract methods	interface can have only abstract methods
A class can extend only one abstract class	A class can implement any number of interfaces
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
abstract class can have static, non-static, final, non-final variables.	interface can have only static final (constant) variable i.e. by default