Abstract Class

- A class that is declared using “abstract” keyword is known as **abstract class**.
- It may or may not include **abstract method**s which means in **abstract class** you can have concrete methods (methods with body)

Abstract Class

Remember two rules:

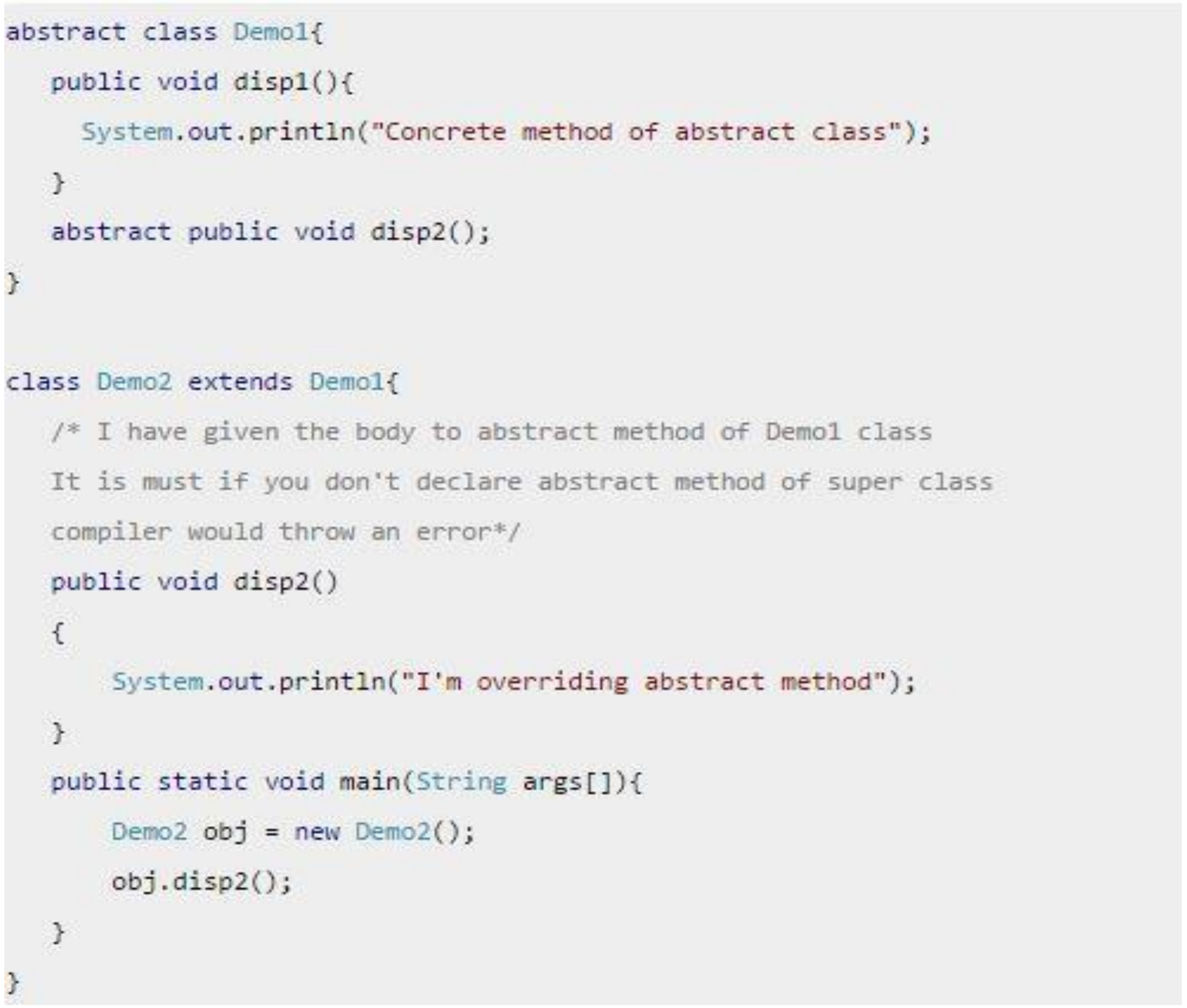
- If the class is having few <u>**abstract methods**</u> and few concrete methods: declare it as abstract class.
- If the class is having only <u>**abstract methods**</u>: declare it as interface.
- **Object creation of abstract class is not allowed**

Abstract Class

Key Points:

- An abstract class has no use until unless it is extended by some other class.
- If you declare an **abstract method** (discussed below) in a class then you must declare the class abstract as well.
- Abstract class can have non-abstract method (concrete) as well.

Abstract Class



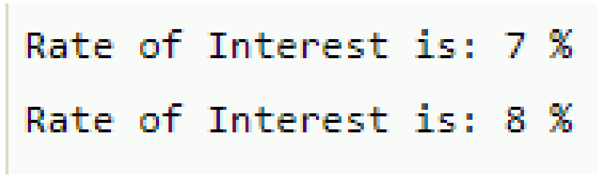
```java
abstract class Demo1{
   public void disp1(){
     System.out.println("Concrete method of abstract class");
   }
   abstract public void disp2();
}

class Demo2 extends Demo1{
   /* I have given the body to abstract method of Demo1 class
   It is must if you don't declare abstract method of super class
   compiler would throw an error*/
   public void disp2()
   {
       System.out.println("I'm overriding abstract method");
   }
   public static void main(String args[]){
       Demo2 obj = new Demo2();
       obj.disp2();
   }
}
```

Another example of Abstract class in java

```
abstract class Bank{
abstract int getRateOfInterest();
}
class SBI extends Bank{
int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
int getRateOfInterest(){return 8;}
}

class TestBank{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
b=new PNB();
System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
}}
```



```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

Interface

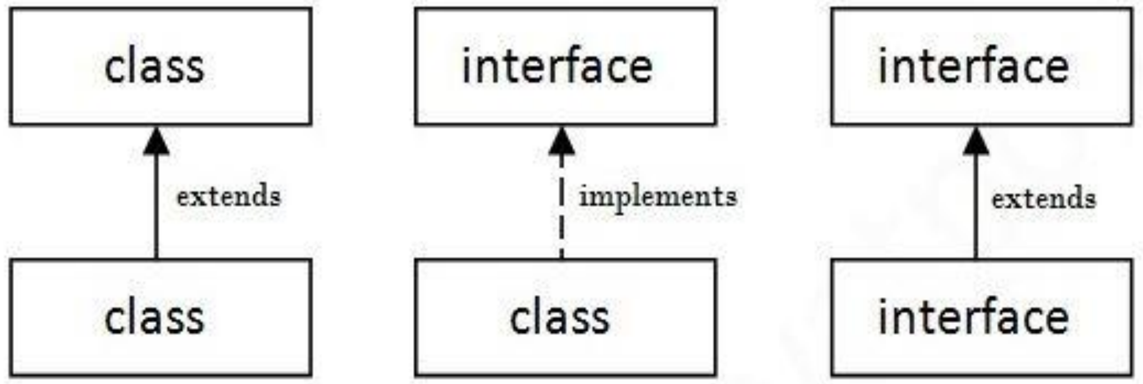
- ✓Similar to a class.
- ✓Consists of only abstract methods and final variables.
- ✓Any number of classes can implement an interface.
- ✓One class can implement any number of interfaces.
- ✓To implement an interface a class must define each of the method declared in the interface. Each class can also add new features.

Interface

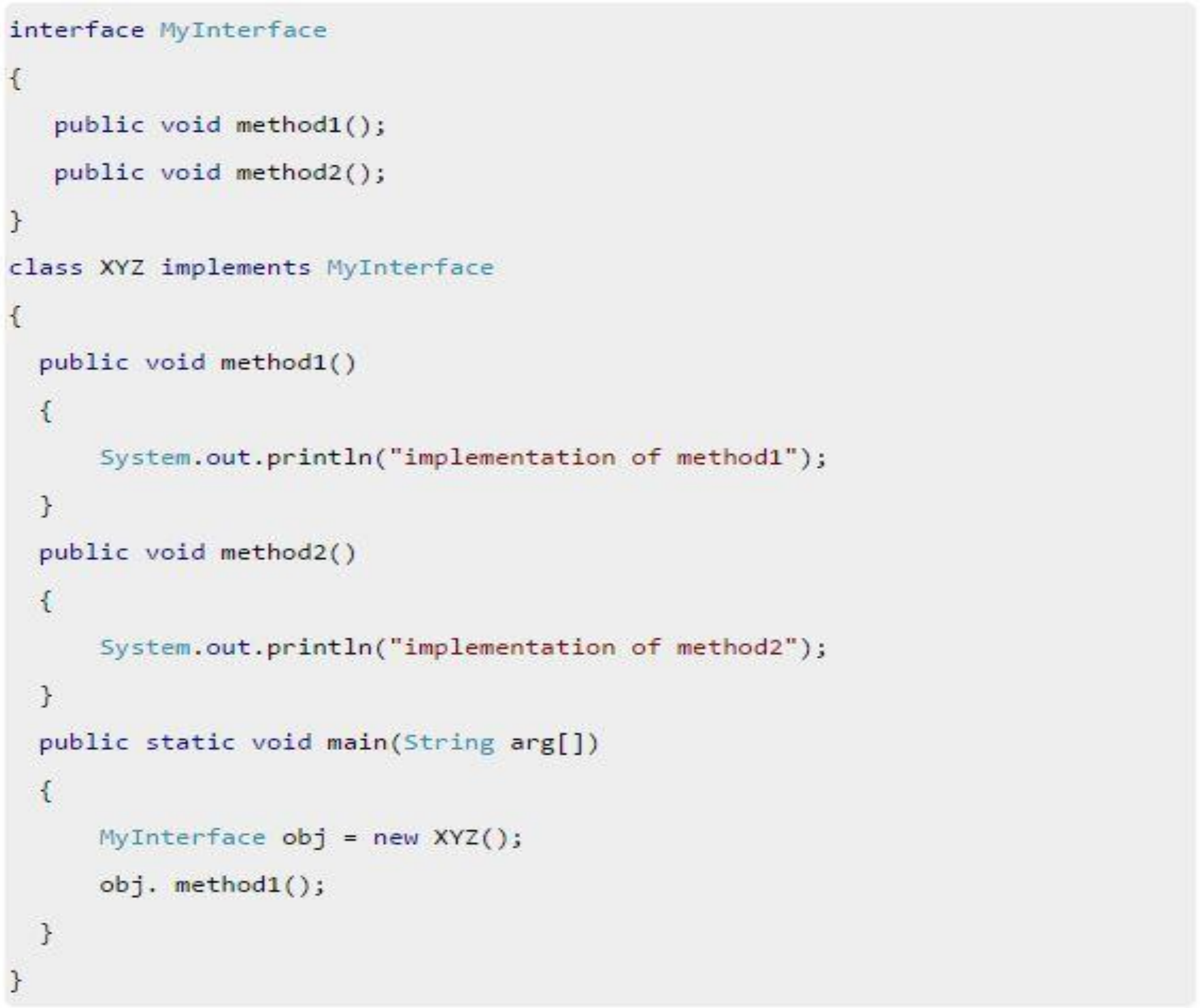
- Interface looks like class but it is not a class.
- An interface can have methods and variables just like the class but the methods declared in interface are by default abstract (only method signature, no body).
- Also, the variables declared in an interface are public, static & final by default.
- Multiple **inheritance is not supported** through **class in java** but it is **possible by interface**

Interface

- 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.
- As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.



Interface Implementation

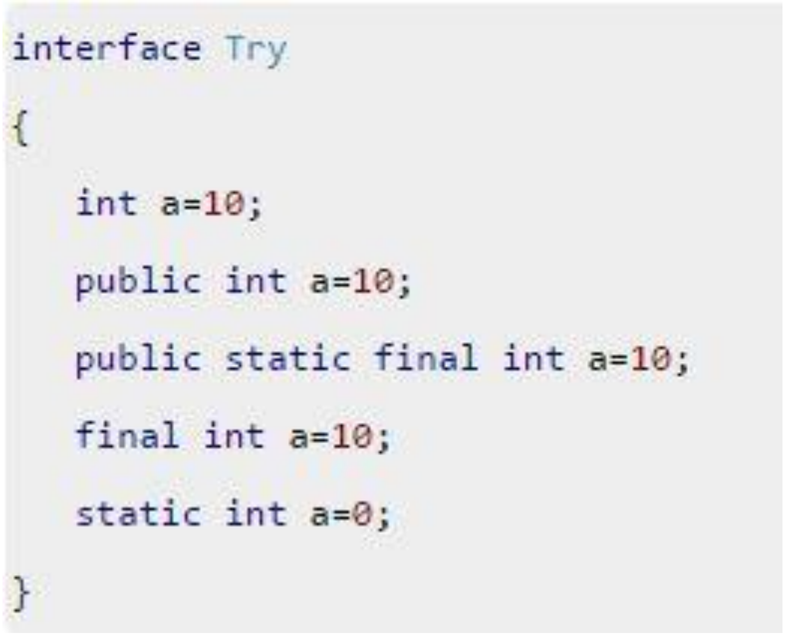


```
interface MyInterface
{
   public void method1();
   public void method2();
}
class XYZ implements MyInterface
{
  public void method1()
  {
      System.out.println("implementation of method1");
  }
  public void method2()
  {
      System.out.println("implementation of method2");
  }
  public static void main(String arg[])
  {
      MyInterface obj = new XYZ();
      obj. method1();
  }
}
```

Note: Class `implements` interface but an interface `extends` another interface.

Interface

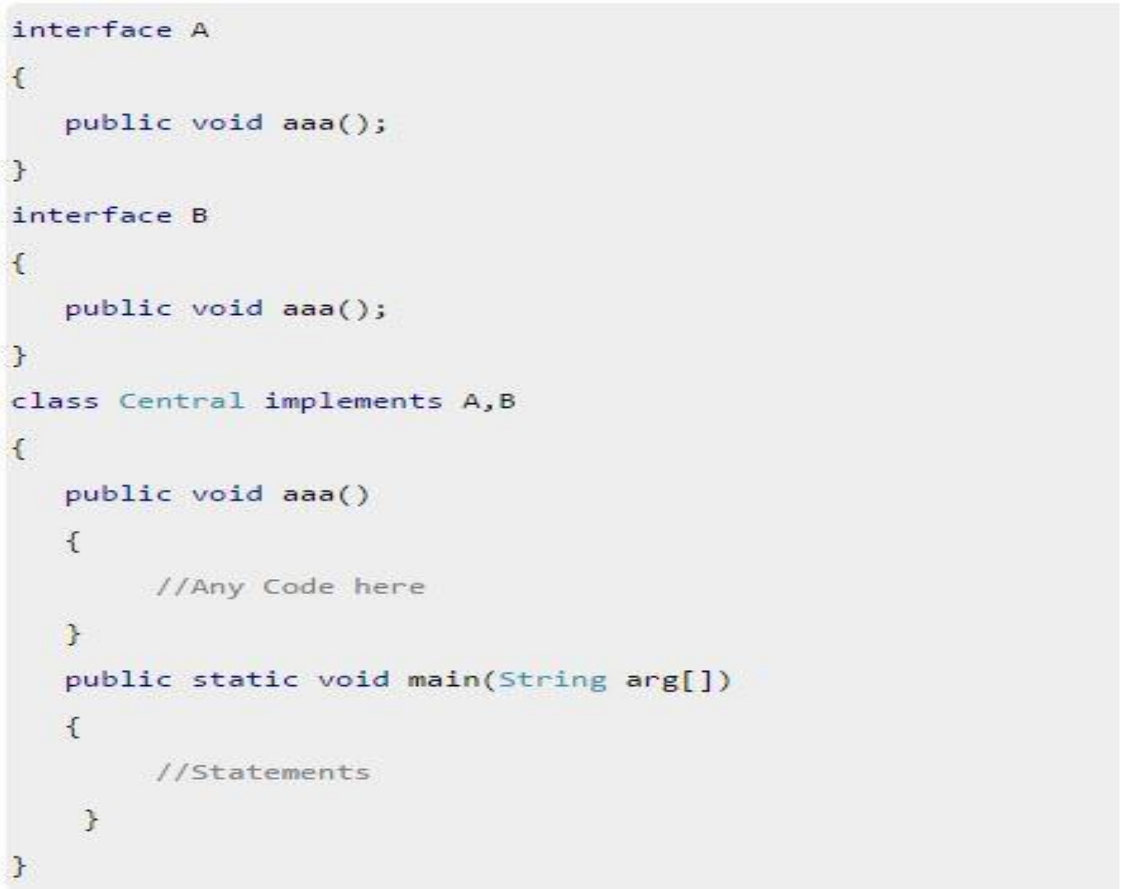
- Interface cannot be declared as private, protected or transient.
- All the interface methods are by default **abstract and public**.
- Variables declared in interface are **public, static and final** by default.



```
interface Try
{
   int a=10;
   public int a=10;
   public static final int a=10;
   final int a=10;
   static int a=0;
}
```

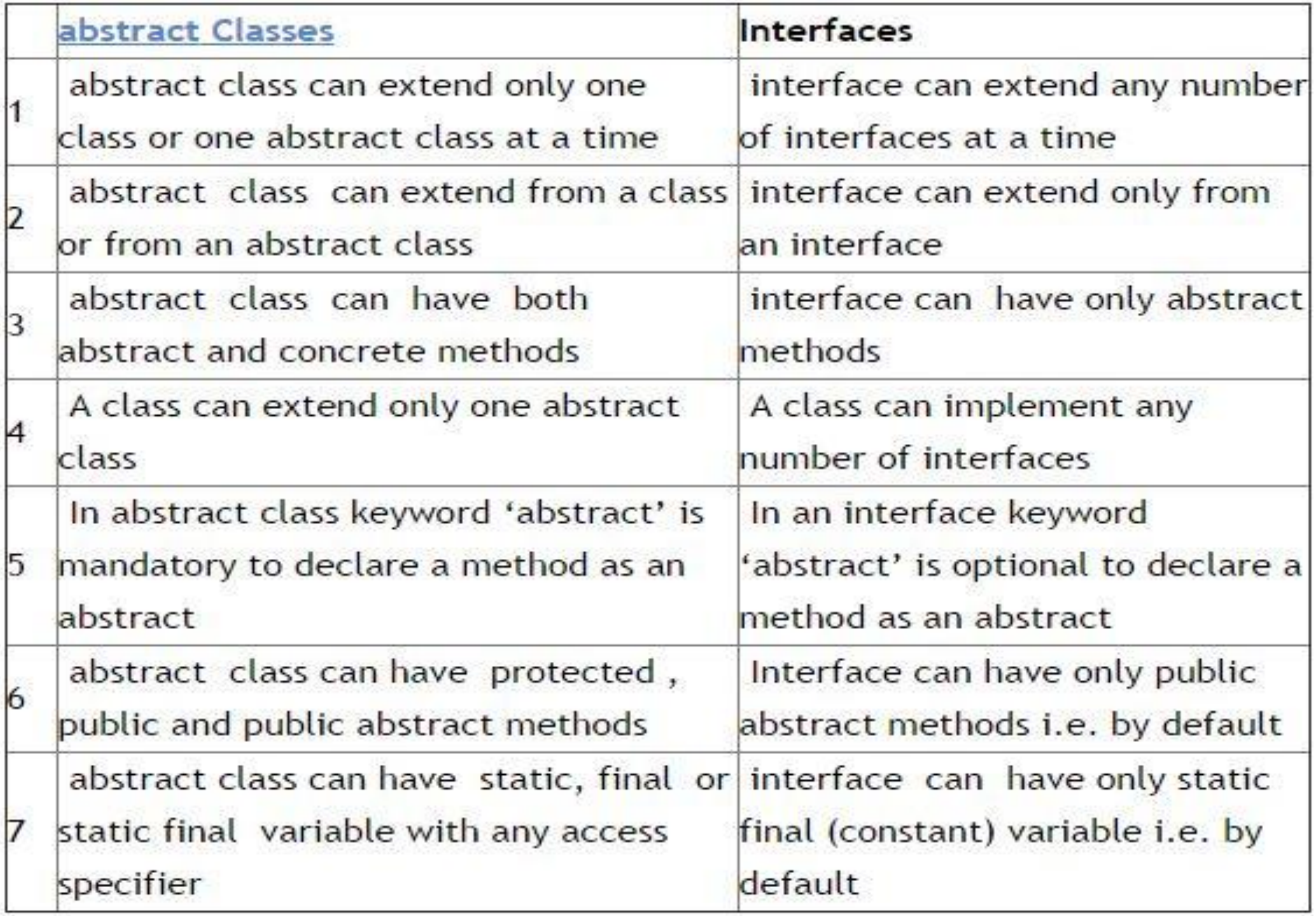
Interface

- A **class** can implement any **<u>number of interfaces</u>**.
- If there are **<u>two or more same methods</u>** in two interfaces and a class implements both interfaces, implementation of the method once is enough.



```
interface A
{
   public void aaa();
}
interface B
{
   public void aaa();
}
class Central implements A,B
{
   public void aaa()
   {
        //Any Code here
   }
   public static void main(String arg[])
   {
        //Statements
    }
}
```

Difference Between Abstract Class and Interface in Java



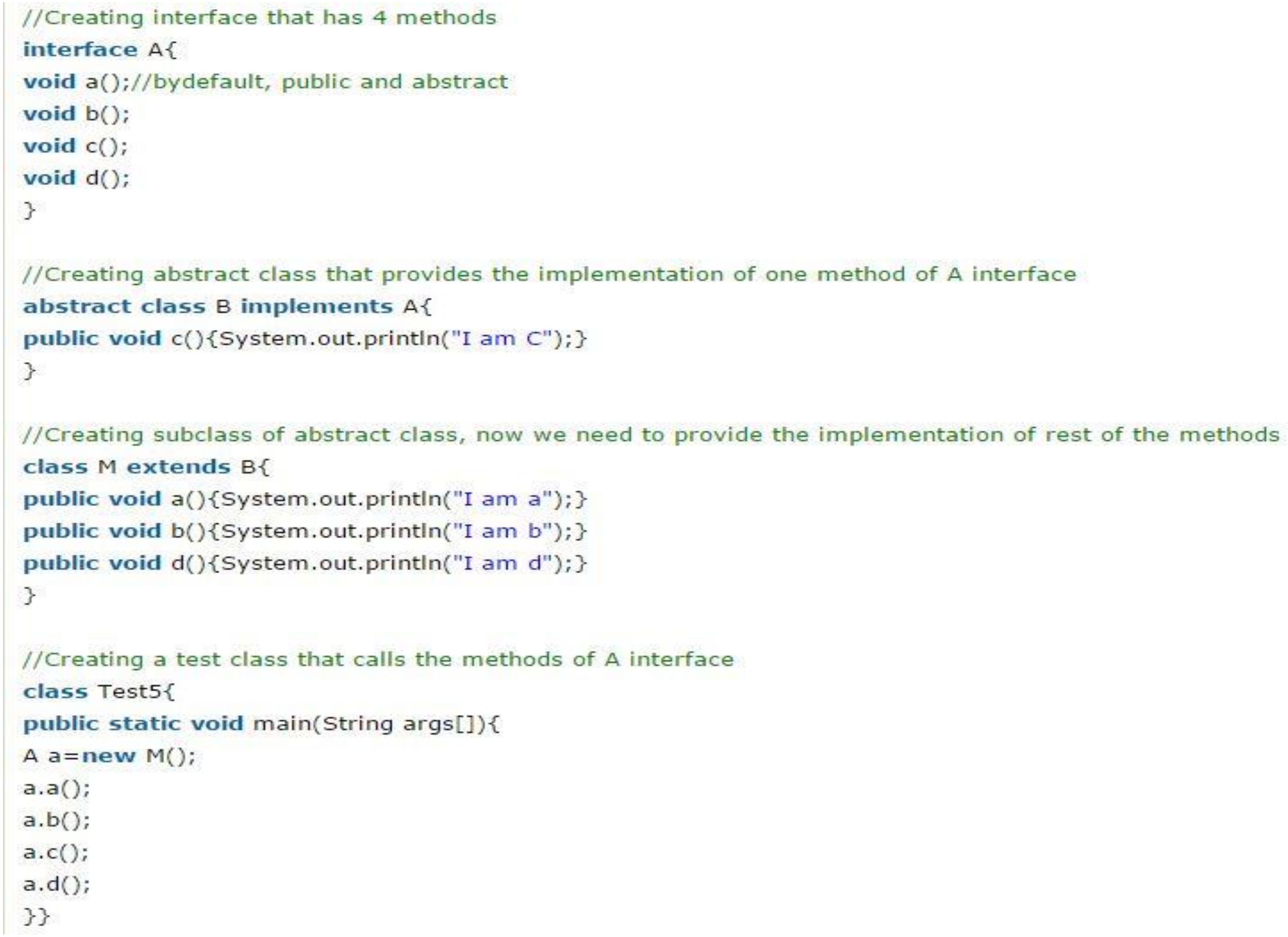
	abstract Classes	Interfaces
1	abstract class can extend only one class or one abstract class at a time	interface can extend any number of interfaces at a time
2	abstract class can extend from a class or from an abstract class	interface can extend only from an interface
3	abstract class can have both 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

Difference Between Abstract Class and Interface in Java

```java
class Example1{
   public void display1(){
      System.out.println("display1 method");
   }
}
abstract class Example2{
   public void display2(){
      System.out.println("display2 method");
   }
}
abstract class Example3 extends Example1{
   abstract void display3();
}
class Example4 extends Example2{
   public void display3(){
      System.out.println("display3 method");
   }
}
class Demo{
   public static void main(String args[]){
       Example4 obj=new Example4();
       obj.display3();
   }
```

```java
//first interface
interface Example1{
    public void display1();
}
//second interface
interface Example2 {
    public void display2();
}
//This interface is extending both the above interfaces
interface Example3 extends Example1,Example2{
}
class Example4 implements Example3{
    public void display1(){
        System.out.println("display2 method");
    }
    public void display2(){
        System.out.println("display3 method");
    }
}
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display1();
    }
}
```

Example of abstract class and interface in Java



```
//Creating interface that has 4 methods
interface A{
void a();//bydefault, public and abstract
void b();
void c();
void d();
}

//Creating abstract class that provides the implementation of one method of A interface
abstract class B implements A{
public void c(){System.out.println("I am C");}
}

//Creating subclass of abstract class, now we need to provide the implementation of rest of the methods
class M extends B{
public void a(){System.out.println("I am a");}
public void b(){System.out.println("I am b");}
public void d(){System.out.println("I am d");}
}

//Creating a test class that calls the methods of A interface
class Test5{
public static void main(String args[]){
A a=new M();
a.a();
a.b();
a.c();
a.d();
}}
```