Interface

Interface is similar to class which is collection of public static final variables (constants) and abstract methods.

The interface is a mechanism to achieve fully abstraction in java. There can be only abstract methods in the interface. It is used to achieve fully abstraction and multiple inheritance in Java.

Why we use Interface?

- It is used to achieve fully abstraction.
- By using Interface, you can achieve multiple inheritance in java.
- It can be used to achieve loose coupling.

Properties of Interface

- It is implicitly abstract. So we no need to use the abstract keyword when declaring an interface.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public.
- All the data members of interface are implicitly public static final.

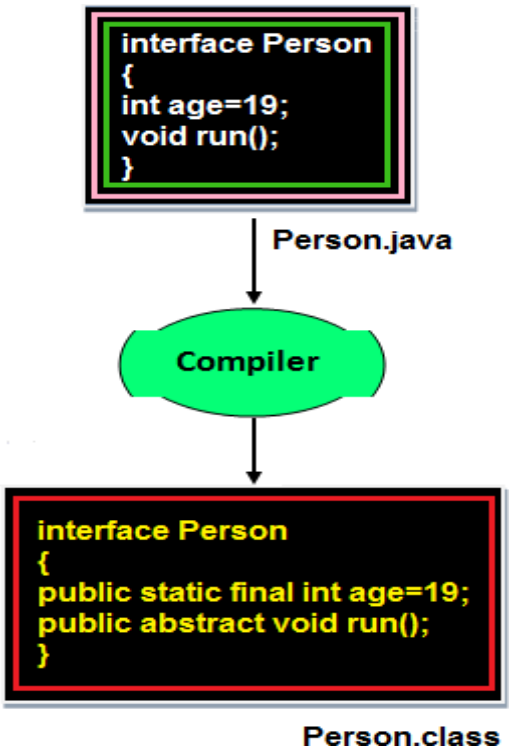
How interface is similar to class?

Whenever we compile any Interface program it generate .class file. That means the byte code of an interface appears in a .class file.

How interface is different from class?

- You cannot instantiate an interface.
- It does not contain any constructors.
- All methods in an interface are abstract.
- Interface cannot contain instance fields. Interface only contains public static final variables.
- Interface is cannot extended by a class; it is implemented by a class.
- Interface can extend multiple interfaces. It means interface support multiple inheritance

Behavior of compiler with Interface program



In the above image when we compile any interface program, by default compiler added public static final before any variable and public abstract before any method. Because Interface is design for fulfill universal requirements and to achieve fully abstraction.

Declaring Interfaces:
The interface keyword is used to declare an interface.
Example

```
interface Person
{
 datatype variablename=value;
 //Any number of final, static fields
 returntype methodname(list of parameters or no parameters)
 //Any number of abstract method declarations
}
```

Explanations
In the above syntax Interface is a keyword interface name can be user defined name the default signature of variable is public static final and for method is public abstract. JVM will add implicitly public static final before data members and public abstract before methods.
Example
public static final datatype variable name=value;---- > for data member
public abstract returntype methodname(parameters) -- > for method
Implementing Interfaces:
A class uses the implements keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.
Example

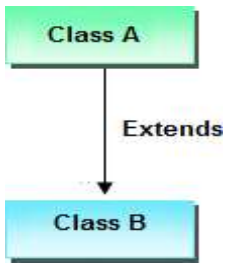
```
interface Person
{
void run();
}
class Employee implements Person
{
public void run()
{
System.out.println("Run fast");
}
}
```

When we use abstract and when Interface
If we do not know about any things about implementation just we have requirement specification then we should be go for Interface
If we are talking about implementation but not completely (partially implemented) then we should be go for abstract
Rules for implementation interface

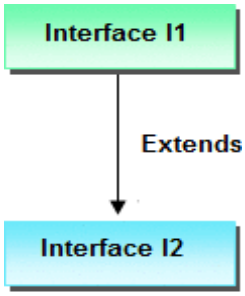
- A class can implement more than one interface at a time.
- A class can extend only one class, but implement many interfaces.
- An interface can extend another interface, similarly to the way that a class can extend another class.

Relationship between class and Interface

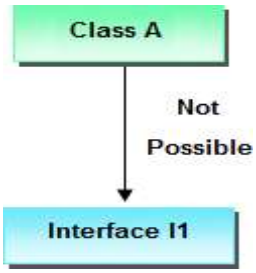
- Any class can extends another class



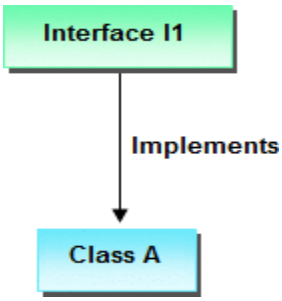
- Any Interface can extends another Interface.



- Any class can Implements another Interface



- Any Interface cannot extend or Implements any class.



Difference between Abstract class and Interface

	Abstract class	Interface
1	It is collection of abstract method and concrete methods.	It is collection of abstract method.
2	There properties can be reused commonly in a specific application.	There properties commonly usable in any application of java environment.
3	It does not support multiple inheritance.	It supports multiple inheritance.
4	Abstract class is preceded by abstract keyword.	It is preceded by Interface keyword.
5	Which may contain either variable or constants.	Which should contains only constants.
6	The default access specifier of abstract class methods are default.	There default access specifier of interface method are public.
7	These class properties can be reused in other class using extend keyword.	These properties can be reused in any other class using implements keyword.
8	Inside abstract class we can take constructor.	Inside interface we cannot take any constructor.

9	For the abstract class there is no restriction like initialization of variable at the time of variable declaration.	For the interface it should be compulsory to initialization of variable at the time of variable declaration.
10	There are no any restriction for abstract class variable.	For the interface variable cannot declare variable as private, protected, transient, volatile.
11	There are no any restriction for abstract class method modifier that means we can use any modifiers.	For the interface method can not declare method as strictfp, protected, static, native, private, final, synchronized.

Example of Interface

```
interface Person
{
void run();  // abstract method
}
class A implements Person
{
public void run()
{
System.out.println("Run fast");
}
public static void main(String args[])
 {
 A obj = new A();
 obj.run();
 }
}
```

Output

```
Run fast
```

Multiple Inheritance using interfaces

Example

```
interface Developer
{
void disp();
}
interface Manager
{
void show();
}

class Employee implements Developer, Manager
{
public void disp()
{
System.out.println("Hello Good Morning");
}
public void show()
{
System.out.println("How are you ?");
}

public static void main(String args[])
{
Employee obj=new Employee();
```

```
obj.disp();
obj.show();
}
}
```

Output

```
Hello Good Morning
How are you ?
```

Marker or tagged interface

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

Example

```
//Way of writing Serializable interface
public interface Serializable
{
}
```

Why interface have no constructor?

Because, constructor are used for eliminate the default values by user defined values, but in case of interface all the data members are public static final that means all are constant so no need to eliminate these values.

Other reason because constructor is like a method and it is concrete method and interface does not have concrete method it have only abstract methods that's why interface have no constructor.