Differences between Interface and Class in Java

Difficulty Level : Easy Last Updated : 13 Jul, 2021

This article highlights the differences between a class and an interface in Java. They seem syntactically similar, both containing methods and variables, but they are different in many aspects.

Class:

A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. **Modifiers** : A class can be public or has default access (Refer this for details).
2. **Class name:** The name should begin with a initial letter (capitalized by convention).
3. **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
4. **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. **Body:** The class body surrounded by braces, { }.

Constructors are used for initializing new objects. Fields are variables that provides the state of the class and its objects, and methods are used to implement the behavior of the class and its objects.

Example:

```
// Java program to demonstrate Class

// Class Declaration
public class Dog {

    // Instance Variables
    String name;
    String breed;
    int age;
```

```
    String color;

    // Constructor Declaration of Class
    public Dog(String name, String breed,
               int age, String color)
    {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }

    // method 1
    public String getName()
    {
        return name;
    }

    // method 2
    public String getBreed()
    {
        return breed;
    }

    // method 3
    public int getAge()
    {
        return age;
    }

    // method 4
    public String getColor()
    {
        return color;
    }

    @Override
    public String toString()
    {
        return ("Hi my name is "
                + this.getName()
                + ".\nMy breed, age and color are "
                + this.getBreed() + ", "
                + this.getAge() + ", "
                + this.getColor());
    }

    public static void main(String[] args)
    {
        Dog tuffy = new Dog("tuffy", "papillon",
                            5, "white");
        System.out.println(tuffy.toString());
    }
}
```

Output:

```
Hi my name is tuffy.
My breed, age and color are papillon, 5, white
```

Interface:

Like a class, an interface can have methods and variables, but the methods declared in interface are by default abstract (only method signature, no body).

- Interfaces specify what a class must do and not how. It is the blueprint of the class.
- An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.
- If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.
- A Java library example is, Comparator Interface. If a class implements this interface, then it can be used to sort a collection.

Syntax :

```
interface <interface_name> {

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

Example:

```
// Java program to demonstrate
```

```
// working of interface.

import java.io.*;

// A simple interface
interface in1 {

    // public, static and final
    final int a = 10;

    // public and abstract
    void display();
}

// A class that implements the interface.
class testClass implements in1 {

    // Implementing the capabilities of
    // interface.
    public void display()
    {
        System.out.println("Geek");
    }

    // Driver Code
    public static void main(String[] args)
    {
        testClass t = new testClass();
        t.display();
        System.out.println(a);
    }
}
```

Output:

```
Geek
10
```

Differences between a Class and an Interface:

Class	Interface

Class	Interface
The keyword used to create a class is “class”	The keyword used to create an interface is “interface”
A class can be instantiated i.e, objects of a class can be created.	An Interface cannot be instantiated i.e, objects cannot be created.
Classes does not support multiple inheritance.	Interface supports multiple inheritance.
It can be inherit another class.	It cannot inherit a class.
It can be inherited by another class using the keyword ‘extends’.	It can be inherited by a class by using the keyword ‘implements’ and it can be inherited by an interface using the keyword ‘extends’.
It can contain constructors.	It cannot contain constructors.
It cannot contain abstract methods.	It contains abstract methods only.
Variables and methods in a class can be declared using any access specifier(public, private, default, protected)	All variables and methods in a interface are declared as public.
Variables in a class can be static, final or neither.	All variables are static and final.