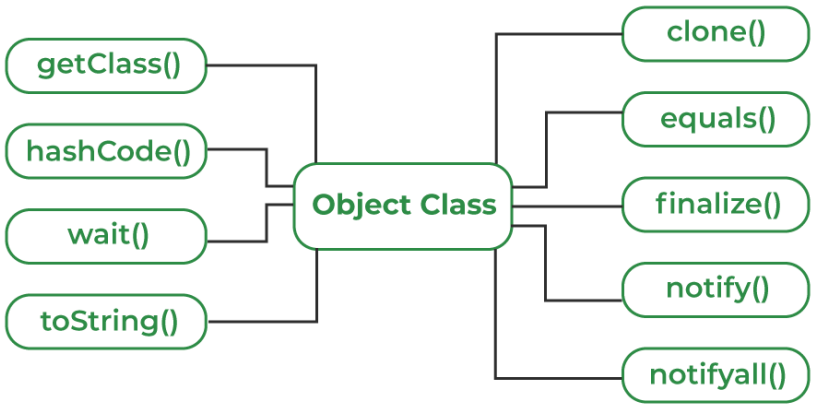
Topics to cover:

1. Object class in Java

1. Object class

Object class is present in java.lang package. Every class in Java is directly or indirectly derived from the Object class. If a class does not extend any other class then it is a direct child class of Object and if extends another class then it is indirectly derived. Therefore the Object class methods are available to all Java classes. Hence the Object class acts as a root of the inheritance hierarchy in any Java Program.



Using Object Class Methods

The Object class provides multiple methods which are as follows:

- tostring() method
- hashCode() method
- equals(Object obj) method
- finalize() method
- getClass() method
- clone() method
- wait(), notify() notifyAll() methods

In this tutorial, we will discuss only some of the methods. For more details, follow the link Object Class in Java on geeksforgeeks or any other site.

1.1 toString() method

The toString() provides a String representation of an object and is used to convert an object to a String. The default toString() method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. In other words, it is defined as:

```
// Default behavior of toString() is to print class name, then

// @, then unsigned hexadecimal representation of the hash code

// of the object

public String toString()

{

   return getClass().getName() + "@" + Integer.toHexString(hashCode());

}
```

It is always recommended to override the toString() method to get our own String representation of Object. For more on override of toString() method refer – Overriding toString() in Java

Note: Whenever we try to print any Object reference, then internally toString() method is called.

```
Student s = new Student();
// Below two statements are equivalent
System.out.println(s);
System.out.println(s.toString());
```

1.2 hashCode() method

For every object, JVM generates a unique number which is a hashcode. It returns distinct integers for distinct objects. A common misconception about this method is that the hashCode() method returns the address of the object, which is not correct. It converts the internal address of the object to an integer by using an algorithm. The hashCode() method is native because in Java it is impossible to find the address of an object, so it uses native languages like C/C++ to find the address of the object.

Use of hashCode() method

It returns a hash value that is used to search objects in a collection. JVM(Java Virtual Machine) uses the hashcode method while saving objects into hashing-related data structures like HashSet, HashMap, Hashtable, etc. The main advantage of saving objects based on hash code is that searching becomes easy.

```
// Java program to demonstrate working of hashCode()
class SampleClass{
}
```

```
public class GetClassMethod {
	public static void main(String[] args) {
		SampleClass obj1 = new SampleClass();
		SampleClass obj2 = new SampleClass();
		SampleClass obj3 = new SampleClass();
		SampleClass obj4 = new SampleClass();
		SampleClass obj5 = new SampleClass();

		System.out.println(obj1.hashCode());
		System.out.println(obj2.hashCode());
		System.out.println(obj3.hashCode());
		System.out.println(obj4.hashCode());
		System.out.println(obj5.hashCode());
	}
}
```

Output :

```
488970385
557041912
1134712904
985922955
1435804085
```

1.3 equals(Object obj) method

It compares the given object to “this” object (the object on which the method is called). It gives a generic way to compare objects for equality.

```
class Parrot{
	String name;
}

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

		Parrot obj1 = new Parrot();
		Parrot obj2 = new Parrot();

		obj1.name="Amazon";

		obj2.name="Amazon";
```

```
                System.out.println(obj1.name+"\t"+obj2.name);

                Parrot obj3=obj1;

                System.out.println(obj1.equals(obj2));

                System.out.println(obj1.equals(obj3));
        }
}
```

Output :

```
Amazon        Amazon
false
true
```

1.4 getClass() method

It returns the class object of “this” object and is used to get the actual runtime class of the object. As it is final so we don’t override it.

```
// Java program to demonstrate working of getClass()

package ObjectPackage;

class SampleClass{

}

public class GetClassMethod {
        public static void main(String[] args) {
                SampleClass obj = new SampleClass();

                System.out.println(obj.getClass());
        }
}
```

Output :

```
class ObjectPackage.SampleClass
```