getClass() method:

- We can use `getClass()` method to get *runtime* class definition of an object. `public final Class getClass()`
- By using this `Class class` object we can access Class level properties like:
    1. Fully qualified name of a class
    2. Methods information
    3. Constructors information etc...

```
    import java.lang.reflect*; // Method is present here
    class Test {
        public static void main(String[] args) {
            int count = 0;
            Object o = new String("Ahmed");
            Class c = o.getClass();

            System.out.println("Fully qualified name of class: " +
c.getName());

            Method[] m = c.getDeclaredMethods();
            System.out.println("Methods information: ");
            for (Method m1 : m) {
                System.out.println(m1.getName());
                count++;
            }
            System.out.println("Number of methods is: " + count);
        }
    }
```

Note:

- After loading every *.class* file, *jvm* will create an object data type `java.lang.Class` in the heap area. Programer can use this class object to get class level information.
- We can use `getClass()` very frequently in reflections.

finalize() method:

- Just before destroying an object, *garbage collector* calls `finalize()` method to perfom **clean up** activities. Once `finalize()` method completes automatically *garbage collector* destroys that object.

wait(), notify(), and notifyAll() methods:

- We can use these methods for inter-thread comunications.
- The thread which is expecting updation, it is responsible to call `wait()` method, then immediatly enter into *waiting state*.

- The thread which is responsible to perform updation, after performing updation the thread can call `notify()` method, the waiting thread will receive that notification and continue its excecution with those updates.