Reflection in Java

Let's say You are working on automation and your client or team lead want you to write the name of the test cases which is getting executed . Now tell me how will you do that ???

We can achieve this easily using reflection

Basically, **Java Reflection** is a *process of examining or modifying the run time behavior of a class at run time*.

The **java.lang.Class** class provides many methods that can be used to get metadata, examine and change the run time behavior of a class.

The java.lang and java.lang.reflect packages provide classes for java reflection.

Java reflection can be used to get many information at runtime, but we will be looking at getting/fetching below info at runtime:

1. Get Object class name.
2. Get declared constructors of a class.
3. Get declared methods of a class.

1. Get class name of an Object:-

```
public class ReflectionTest {


public int var1;
private int var2;

public ReflectionTest()
{

}
private ReflectionTest(int i)
{

}
public void m1()
{

}
private void m2()
{

}
}
public class Demo {

	public static void main(String[] args) {

			ReflectionTest obj = new ReflectionTest();
			Class clazz= obj.getClass();

			System.out.println("Name of the class of the object-->"+clazz.getName());
			for(Constructor c:clazz.getConstructors())
			{
				System.out.println(c.getName());
			}
			System.out.println(clazz.getDeclaredConstructors().length);
			for(Constructor c:clazz.getDeclaredConstructors())
			{
```

```
                    System.out.println(c.getName());
                }
        }

}
```

2. Get list of methods:-

```
public class Main {

  public static void main(String[] args) {
    Class aClass = String.class;

    // Get the methods
    Method[] methods = aClass.getDeclaredMethods();

    // Loop through the methods and print out their names
    for (Method method : methods) {
      System.out.println(method.getName());
    }
  }
}
```

3. Get method return types:-

```
public class A {
        public static void main(String[] args) {
                A a= new A();
                Class clazz=a.getClass();
                System.out.println(clazz.getDeclaredMethods());
                for(Method m:clazz.getDeclaredMethods())
                {
                        System.out.println(m.getName());
                        System.out.println(m.getReturnType().getName());
                        //System.out.println(m.get);
                }
        }

        public void m1()
        {

        }
}
```

Similarly we can find/access the declared constructors as well.