# Java Reflection API

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.

### Where it is used

The Reflection API is mainly used in:

* IDE (Integrated Development Environment) e.g., Eclipse, MyEclipse, NetBeans etc.
* Debugger
* Test Tools etc.

*Do You Know?*

* How many ways can we get the instance of Class class?
* How to create the javap tool?
* How to create the appletviewer tool?
* How to access the private method from outside the class?

### java.lang.Class class

The java.lang.Class class performs mainly two tasks:

* provides methods to get the metadata of a class at run time.
* provides methods to examine and change the run time behavior of a class.

### Commonly used methods of Class class:

| **Method** | **Description** |
| --- | --- |
| 1) public String getName() | returns the class name |
| 2) public static Class forName(String className)throws ClassNotFoundException | loads the class and returns the reference of Class class. |
| 3) public Object newInstance()throws InstantiationException,IllegalAccessException | creates new instance. |
| 4) public boolean isInterface() | checks if it is interface. |
| 5) public boolean isArray() | checks if it is array. |
| 6) public boolean isPrimitive() | checks if it is primitive. |
| 7) public Class getSuperclass() | returns the superclass class reference. |
| 8) public Field[] getDeclaredFields()throws SecurityException | returns the total number of fields of this class. |
| 9) public Method[] getDeclaredMethods()throws SecurityException | returns the total number of methods of this class. |
| 10) public Constructor[] getDeclaredConstructors()throws SecurityException | returns the total number of constructors of this class. |
| 11) public Method getDeclaredMethod(String name,Class[] parameterTypes)throws NoSuchMethodException,SecurityException | returns the method class instance. |

### How to get the object of Class class?

There are 3 ways to get the instance of Class class. They are as follows:

* forName() method of Class class
* getClass() method of Object class
* the .class syntax

### 1) forName() method of Class class

* is used to load the class dynamically.
* returns the instance of Class class.
* It should be used if you know the fully qualified name of class.This cannot be used for primitive types.

Let's see the simple example of forName() method.

FileName: Test.java

* **class** Simple{}
* **public** **class** Test{
* **public** **static** **void** main(String args[]) **throws** Exception {
* Class c=Class.forName("Simple");
* System.out.println(c.getName());
* }
* }

Output:

*Simple*

### 2) getClass() method of Object class

It returns the instance of Class class. It should be used if you know the type. Moreover, it can be used with primitives.

FileName: Test.java

* **class** Simple{}
* **class** Test{
* **void** printName(Object obj){
* Class c=obj.getClass();
* System.out.println(c.getName());
* }
* **public** **static** **void** main(String args[]){
* Simple s=**new** Simple();
* Test t=**new** Test();
* t.printName(s);
* }
* }

Output:

*Simple*

### 3) The .class syntax

If a type is available, but there is no instance, then it is possible to obtain a Class by appending ".class" to the name of the type. It can be used for primitive data types also.

FileName: Test.java

* **class** Test{
* **public** **static** **void** main(String args[]){
* Class c = **boolean**.**class**;
* System.out.println(c.getName());
* Class c2 = Test.**class**;
* System.out.println(c2.getName());
* }
* }

Output:

*boolean*

*Test*

### Determining the class object

The following methods of Class class are used to determine the class object:

1) public boolean isInterface(): determines if the specified Class object represents an interface type.

2) public boolean isArray(): determines if this Class object represents an array class.

3) public boolean isPrimitive(): determines if the specified Class object represents a primitive type.

Let's see the simple example of reflection API to determine the object type.

FileName: Test.java

* **class** Simple{}
* **interface** My{}
* **class** Test{
* **public** **static** **void** main(String args[]){
* **try**{
* Class c=Class.forName("Simple");
* System.out.println(c.isInterface());
* Class c2=Class.forName("My");
* System.out.println(c2.isInterface());
* }**catch**(Exception e){System.out.println(e);}
* }
* }

Output:

*false*

*true*

### Pros and Cons of Reflection

Java reflection should always be used with caution. While the reflection provides a lot of advantages, it has some disadvantages too. Let's discuss the advantages first.

Pros: Inspection of interfaces, classes, methods, and fields during runtime is possible using reflection, even without using their names during the compile time. It is also possible to call methods, instantiate a clear or to set the value of fields using reflection. It helps in the creation of Visual Development Environments and class browsers which provides aid to the developers to write the correct code.

Cons: Using reflection, one can break the principles of encapsulation. It is possible to access the private methods and fields of a class using reflection. Thus, reflection may leak important data to the outside world, which is dangerous. For example, if one accesses the private members of a class and sets null value to it, then the other user of the same class can get the NullReferenceException, and this behaviour is not expected.

Another demerit is the overhead in performance. Since the types in reflection are resolved dynamically, JVM (Java Virtual Machine) optimization cannot take place. Therefore, the operations performed by reflections are usually slow.

### Conclusion

Because of the above-mentioned cons, it is generally advisable to avoid using reflection. It is an advanced feature that should only be used by programmers or developers who have a good knowledge of the basics of the language. Always remember! Whenever reflection is used, the security of the application is compromised.