**Reflection**

Java reflection is the process of examining or modifying the run time behavior of a class.

The java.lang.Class class performs two tasks:

1. Proivdes methods to get the metadata of a class at runtime.
2. Provides methods to examine and change the run time behavior of a class.

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

The reflection API is mainly used in

1. IDE
2. Debugger
3. Test tools etc.

Some of the main classes and interface which you will use while using reflection in java are as follows:

1. Class (java.lang.Class)
2. Member (java.lang.reflect.Member)
3. Field (java.lang.reflect.Field)
4. Method (java.lang.reflect.Method)
5. Constructor (java.lang.reflect.constructor)
6. Array (java.lang.reflect.Array)

**Reflection in Java – Constructor**

Using java.lang.reflect.constructor class in java you can get information about the modifiers,parameters, annotations and throws exception. You can also create a new instance of class using a specified constructor.

* getConstructor(Class<?>…parameterType
* getConstructors
* getDeclaredConstructor(Class<?>.. parameter types)
* getDeclaredConstructors

**package** JavaConceptsRelflectionConcepts;

**public** **class** TestClass {

**private** **int** value;

**private** String name;

**public** TestClass(**int** value,String name) {

**this**.name=name;

**this**.value=value;

}

**private** TestClass()

{

}

**public** **void** showValue() {

System.***out***.println("Name:"+name+"\t"+"Value:"+value);

}

}

**package** JavaConceptsRelflectionConcepts;

**import** java.lang.reflect.Constructor;

**import** java.util.Arrays;

**public** **class** ReflectionConstructorTest {

**public** **static** **void** main(String[] args) {

**try** {

Class<?> cls = Class.*forName*("JavaConceptsRelflectionConcepts.TestClass");

System.***out***.println("-------1. To get the constructor with 2 args-----------");

Constructor<?> constructor = cls.getConstructor(**int**.**class**,String.**class**);

System.***out***.println("constructor: "+constructor.toString());

//2. Getting constructors of the class

System.***out***.println("-------2. Getting constructors of the class--------");

Constructor<?>[] constructors = cls.getConstructors();

System.***out***.println("constructors: "+Arrays.*toString*(constructors));

//3. TO get the private constructor using getDeclaredConstructor() method

System.***out***.println("-------3. To get the private construcotr using getDeclaredConstructor-----");

constructor = cls.getDeclaredConstructor();

System.***out***.println("constructor: "+constructor);

//4. To get all public, protected, private, default constructors

System.***out***.println("---------4.To get all public, protected, private, default constructors--------- ");

Constructor<?>[] constructors2 = cls.getDeclaredConstructors();

System.***out***.println("Decalred constructors: "+Arrays.*toString*(constructors2));

} **catch** (ClassNotFoundException | NoSuchMethodException| SecurityException e) {

}

}

}

**Creating class instance using reflection**

In reflection API there are two methods for creting instances pf classes

* **Class.newInstance()** can only invoke the zero-argument constructor. While Constructor.newInstance() may invoke any constructor regardless of the number of parameters.
* **Class.newInstance()** requires that constructor be visible; Constructor.newInstance() can invoke private constructor also by setting accessibility to true
* **Class.newInstance()** throws any exception thrown by the constructor whether it is checked or unchecked exception . **Constructor.newInstance()** always wraps the thrown exception with an **InvocationTargetException**