**Java Reflections**

**java.lang.Class**

|  |
| --- |
| There is a pre-defined class called Class in java.lang package |

**forName()**

|  |
| --- |
| 1. It is a static method existed in java.lang.Class 2. To this method we can pass a fully qualified class name as a string 3. This forName() method loads the specified class into method area and also it returns an object of a class called Class which represents the loaded class. |

**First example to create object using reflections**

**package** main;

**public** **class** ReflectDemo1 {

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

{

**try** {

Class c=Class.*forName*("mts.Emp");

System.***out***.println("Emp class is loaded and Class object is created");

System.***out***.println("Name:\t"+c.getName());

mts.Emp e1=(mts.Emp)c.~~newInstance~~();

System.***out***.println(e1);

} **catch** (Exception e)

{

e.printStackTrace();

}

}

}

**Output:**

Emp class is loaded and Class object is created

Name: mts.Emp

default constructor of Emp class

Emp [eno=0, ename=null, sal=0.0]

ReflectDemo1.java(calling a static method )

**package** main;

**import** java.lang.reflect.Field;

**import** java.lang.reflect.Method;

**public** **class** ReflectDemo1

{

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

{

**try** {

Class c=Class.*forName*("mts.Emp");

Method[] methods= c.getMethods();

**for**(Method m:methods)

{

**if**(m.getName().equals("fun1"))

{

m.invoke(**null**, **null**);

}

//System.out.println(m.getName());

}

// Field[] fields=c.getFields();

// System.out.println("Fields.....................");

// for(Field f:fields)

// {

// System.out.println(f.getName());

// }

// System.out.println(c.newInstance());

} **catch** (Exception e)

{

e.printStackTrace();

}

}

}

Output:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*static method of emp class