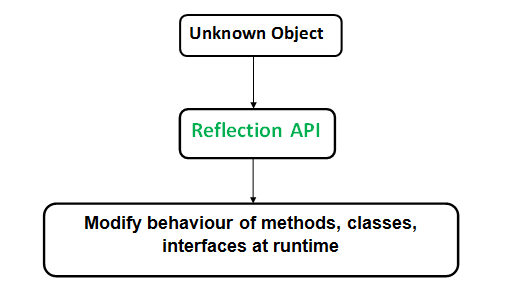
**Reflection in Java**

Reflection is an API which is used to examine or modify the behavior of methods, classes, interfaces at runtime.

* The required classes for reflection are provided under java.lang.reflect package.
* Reflection gives us information about the class to which an object belongs and also the methods of that class which can be executed by using the object.
* Through reflection we can invoke methods at runtime irrespective of the access specifier used with them.



### Class in java.lang.reflect Package

1. **Class** The getClass() method is used to get the name of the class to which an object belongs.
2. **Constructors** The getConstructors() method is used to get the public constructors of the class to which an object belongs.
3. **Methods** The getMethods() method is used to get the public methods of the class to which an objects belongs.

### Methods used in java.lang.Class

* **Public String getName ()**: Returns the name of the class.
* **public Class getSuperclass()**: Returns the super class reference
* **Public Class[] getInterfaces()** : Returns an array of interfaces implemented by the specified class
* **Public in getModifiers ():** Returns an integer value representing the modifiers of the specified class which needs to be passed as parameter to "**public static String toString (int i )"** method which returns the access specifier for the given class.

**Example-**

import java.lang.reflect.Method;

import java.lang.reflect.Field;

import java.lang.reflect.Constructor;

class Test

{

// creating a private field

private String s;

// creating a public constructor

public Test()

{

s = "Netparam";

}

// Creating a public method with no arguments

public void method1() {

System.out.println("The string is " + s);

}

// Creating a public method with int as argument

public void method2(int n) {

System.out.println("The number is " + n);

}

// creating a private method

private void method3() {

System.out.println("Private method invoked");

}

}

class Demo

{

public static void main(String args[]) throws Exception

{

// Creating object whose property is to be checked

Test obj = new Test();

// Creating class object from the object using

// getclass method

Class cls = obj.getClass();

System.out.println("The name of class is " +

cls.getName());

Constructor constructor = cls.getConstructor();

System.out.println("The name of constructor is " +

constructor.getName());

System.out.println("The public methods of class are : ");

Method[] methods = cls.getMethods();

// Printing method names

for (Method method:methods)

System.out.println(method.getName());

Method methodcall1 = cls.getDeclaredMethod("method2",int.class);

methodcall1.invoke(obj, 19);

Field field = cls.getDeclaredField("s");

field.setAccessible(true);

field.set(obj, "NetParam");

Method methodcall2 = cls.getDeclaredMethod("method1");

// invokes the method at runtime

methodcall2.invoke(obj);

Method methodcall3 = cls.getDeclaredMethod("method3");

methodcall3.setAccessible(true);

// invokes the method at runtime

methodcall3.invoke(obj);

}

}

**Output**

The name of class is Test

The name of constructor is Test

The public methods of class are :

method2

method1

wait

wait

wait

equals

toString

hashCode

getClass

notify

notifyAll

The number is 19

The string is NetParam

Private method invoked