Reflection In Java:

Java Reflection makes it possible to inspect classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It is also possible to instantiate new objects, invoke methods and get /set field values using reflection.

Java Reflection is quite powerful and can be very useful. For instance, when mapping objects to tables in a database at runtime. Or, when mapping the statements in a script language to method calls on real objects at runtime. This concept is often mixed with introspection. The following are their definitions:

• Introspection is the ability of a program to examine the type or properties of an object at runtime.

• Reflection is the ability of a program to examine and modify the structure and behavior of an object at runtime. From their definitions, introspection is a subset of reflection. Some languages support introspection, but do not support reflection, e.g., C++.

Why do we need reflection?

Reflection enables us to:

1. Examine an object's class at runtime.

2. Construct an object for a class at runtime. 3. Examine a class's field and method at runtime.

4. Invoke any method of an object at runtime.

5. Change accessibility flag of Constructor, Method and Field.

Object of unknown type-->reflection-->Modify Structure and behaviour

Example : Get class name from object import java.lang.reflect.\*;

public class ReflectionHelloWorld

{

public static void main(String[] args)

{

Hello f = new Hello(); System.out.println(f.getClass().getName());

}

}

class Hello

{

public void print()

{

System.out.println("abc");

}

}

Output : Reflection.Hello