Java Reflection API – Complete Notes

Reflection

- Reflection is the process of analyzing all capabilities of a particular class at **runtime**.
- Using Reflection, we can read metadata of a class such as:
  - Name of the class
  - Superclass details
  - Implemented interfaces
  - Variables
  - Constructors
  - Methods

Reflection API

- Java provides a predefined set of classes and interfaces under **java.lang.reflect** to perform Reflection.
- Frequently used classes:
  - `java.lang.Class`
  - `java.lang.reflect.Field`
  - `java.lang.reflect.Constructor`
  - `java.lang.reflect.Method`
  - `java.lang.reflect.Modifier`

---

1. java.lang.Class

Purpose

- Manages metadata of a class such as:
  - Class name (`String`)
  - Superclass metadata (`Class`)
  - Interfaces metadata (`Class[]`)
  - Variables metadata (`Field[]`)
  - Constructors metadata (`Constructor[]`)
  - Methods metadata (`Method[]`)

Ways to Get `Class` Object

Method	Description
`Class.forName()`	Loads bytecode at runtime by class name
`object.getClass()`	From an object instance
`ClassName.class`	Using class literal

Using `Class.forName()`

```
java
CopyEdit
Class cls = Class.forName("Employee");
```

What Happens Internally:

1. JVM searches for the `.class` file in the current dir, libraries, and classpath.
2. If not found → `ClassNotFoundException`.
3. If found → bytecode is loaded to memory.
4. JVM reads metadata and stores it in a `Class` object.

Using `getClass()`

```
java
CopyEdit
Employee emp = new Employee();
Class cls = emp.getClass();
```

Using `.class`

```
java
CopyEdit
Class cls = Employee.class;
```

Important Methods of Class

Method	Description
`getName()`	Gets class name
`getSuperclass()`	Gets superclass metadata
`getInterfaces()`	Gets implemented interfaces
`getModifiers()`	Gets access modifiers (use `Modifier.toString()`)

```
java
CopyEdit
int val = clazz.getModifiers();
String accessModifiers = Modifier.toString(val);
```

Class Example

Employee.java

```
java
CopyEdit
public class Employee implements Serializable, Comparable, Cloneable {
    private int eno;
    private String ename;
    private float esal;
    private String eaddr;

    public Employee(int eno) {}
    public Employee(int eno, String ename) {}
    public Employee(int eno, String ename, float esal) {}
    public Employee(int eno, String ename, float esal, String eaddr) {}

    public void add(int eno, String ename, float esal, String eaddr) {}
    public Employee search(int eno) { return null; }
    public String update(int eno, String ename, float esal, String eaddr) {
return null; }
    public String delete(int eno) { return null; }
}
```

Main.java

```
java
CopyEdit
import java.lang.reflect.Modifier;

public class Main {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("Employee");
        System.out.println("Class Name          : " + clazz.getName());
        System.out.println("Super Class Name    : " +
clazz.getSuperclass().getName());
        System.out.print("Interfaces Names    : ");
        Class[] interfaces = clazz.getInterfaces();
        for (Class i : interfaces) {
            System.out.print(i.getName() + " ");
        }
        System.out.println();
        int val = clazz.getModifiers();
        System.out.println("Modifiers           : " +
Modifier.toString(val));
    }
}
```

Output

```
pgsql
CopyEdit
Class Name          : Employee
Super Class Name    : java.lang.Object
Interfaces Names    : java.io.Serializable java.lang.Cloneable
Modifiers           : public final
```

---

2. java.lang.reflect.Field

- Manages metadata of class fields (variables).

Methods

Method	Description
`getFields()`	All public fields (including inherited)
`getDeclaredFields()`	All fields regardless of modifier
`getName()`	Variable name
`getType()`	Data type
`get(field)`	Value of the variable (static fields only)

Field Example

Employee.java

```
java
CopyEdit
public class Employee {
    private static int eno = 111;
    public static String ename = "Durga";
    protected static float esal = 100000.0f;
    public static String eaddr = "Hyderabad";
}
```

Main.java

```
java
CopyEdit
import java.lang.reflect.Field;
import java.lang.reflect.Modifier;

public class Main {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("Employee");
        Field[] fields = clazz.getDeclaredFields();
        for (Field field : fields) {
            field.setAccessible(true);
            System.out.println("Name      : " + field.getName());
            System.out.println("Type      : " + field.getType().getName());
            System.out.println("Modifiers : " +
Modifier.toString(field.getModifiers()));
            System.out.println("Value     : " + field.get(field));
            System.out.println("------------------------------");
        }
    }
}
```

Output

```
yaml
```

```
CopyEdit
Name      : eno
Type      : int
Modifiers : private static
Value     : 111
------------------------------
Name      : ename
Type      : java.lang.String
Modifiers : public static
Value     : Durga
------------------------------
Name      : esal
Type      : float
Modifiers : protected static
Value     : 100000.0
------------------------------
Name      : eaddr
Type      : java.lang.String
Modifiers : public static
Value     : Hyderabad
------------------------------
```

---

3. java.lang.reflect.Constructor

- Manages metadata of constructors.

Methods

Method	Description
`getConstructors()`	Public constructors
`getDeclaredConstructors()`	All constructors
`getName()`	Constructor name
`getParameterTypes()`	Constructor parameters
`getExceptionTypes()`	Thrown exceptions

Constructor Example

Employee.java

```
java
CopyEdit
public class Employee {
    public Employee(int eno) throws Exception {}
    protected Employee(int eno, String ename) throws ClassCastException,
ClassNotFoundException {}
    private Employee(int eno, String ename, float sal) throws
ArithmeticException, NullPointerException {}
    Employee(int eno, String ename, float sal, String eaddr) throws
ArithmeticException, ClassCastException {}
}
```

Main.java

```
java
CopyEdit
import java.lang.reflect.Constructor;
import java.lang.reflect.Modifier;

public class Main {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("Employee");
        Constructor[] constructors = clazz.getDeclaredConstructors();
        for (Constructor constructor : constructors) {
            System.out.println("Name           : " +
constructor.getName());
            System.out.println("Modifiers      : " +
Modifier.toString(constructor.getModifiers()));
            System.out.print("ParameterTypes : ");
            for (Class param : constructor.getParameterTypes()) {
                System.out.print(param.getName() + " ");
            }
            System.out.println();
            System.out.print("ExceptionTypes : ");
            for (Class ex : constructor.getExceptionTypes()) {
                System.out.print(ex.getName() + " ");
            }
            System.out.println("\n---------------------------------");
        }
    }
}
```

---

4. java.lang.reflect.Method

- Provides metadata of methods (name, return type, parameters, exceptions).

Methods

Method	Description
`getMethods()`	All public methods
`getDeclaredMethods()`	All methods
`getName()`	Method name
`getReturnType()`	Return type
`getParameterTypes()`	Parameters
`getExceptionTypes()`	Thrown exceptions

Method Example

Employee.java

```
java
CopyEdit
public class Employee {
    public void add(int eno, String ename, float esal, String eaddr) throws
Exception {}
```

```
    protected String search(int eno) throws ClassCastException,
ClassNotFoundException { return ""; }
    private void update(int eno, String ename, float sal, String eaddr)
throws ArithmeticException, NullPointerException {}
    String delete(int eno) throws ArithmeticException, ClassCastException {
return ""; }
}
```

Main.java

```
java
CopyEdit
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;

public class Main {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("Employee");
        Method[] methods = clazz.getDeclaredMethods();
        for (Method method : methods) {
            System.out.println("Name           : " + method.getName());
            System.out.println("Modifiers      : " +
Modifier.toString(method.getModifiers()));
            System.out.println("ReturnType     : " +
method.getReturnType().getName());
            System.out.print("ParameterTypes : ");
            for (Class param : method.getParameterTypes()) {
                System.out.print(param.getName() + " ");
            }
            System.out.println();
            System.out.print("ExceptionTypes : ");
            for (Class ex : method.getExceptionTypes()) {
                System.out.print(ex.getName() + " ");
            }
            System.out.println("\n---------------------------------");
        }
    }
}
```