Reflection API – the dark side of java

Light talk author

Stanislau Sukora,
Software Engineer

What is reflection?

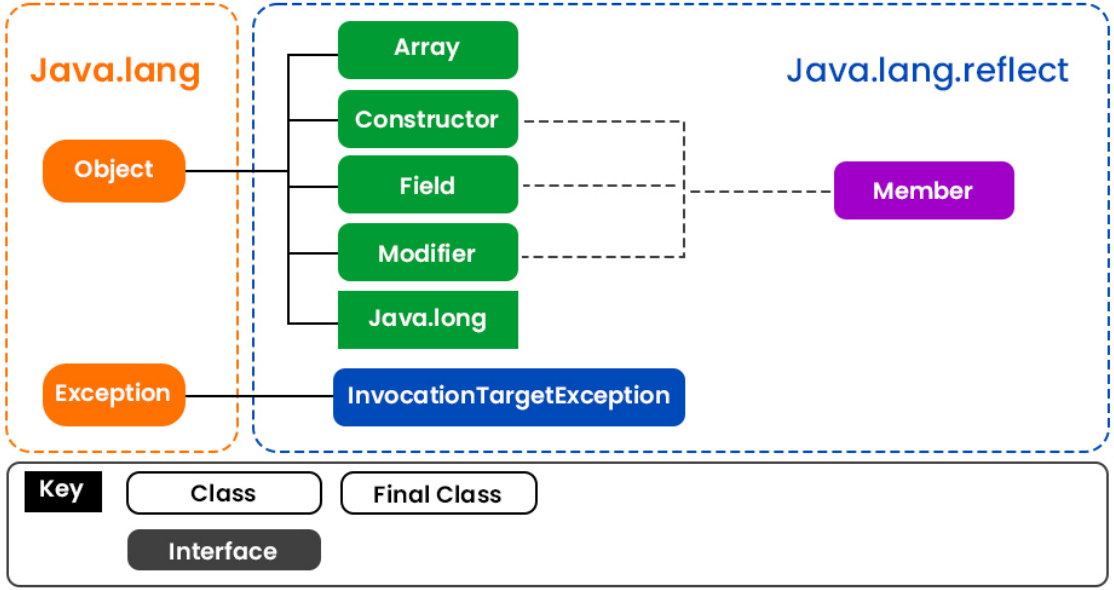
The dark side of java

Why? Because you can literally juggle classes and their components.

What is Reflection?

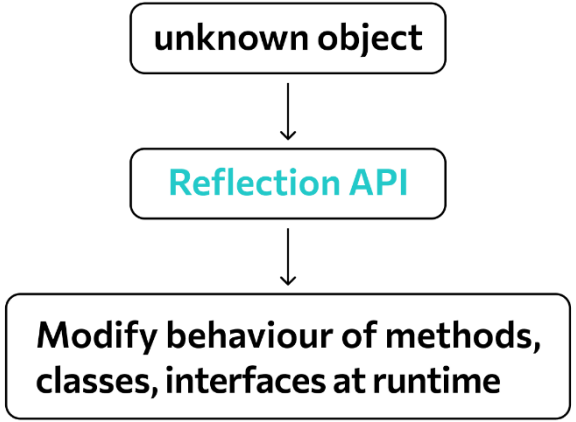
What is Reflection?

This mechanism research data about the program during its execution. Is commonly used by programs which require the ability to examine or modify the runtime behavior of applications running in the Java virtual machine. Reflection allows us to investigate the information about the fields , methods and constructors of classes.



What can we do?

- Determine the object class .
- Get information about modifiers classes , fields, methods, constructors, and super classes .
- To find out what constants and methods belong to the interface.
- Create an instance of a class whose name is not known until runtime.
- Get and set the value of the object.
- Call the object method.
- Create a new array , the size and type of components which are not known until runtime programs.



Practice

```
public class MyClass {
    3 usages
    private int number;
    2 usages
    private String name = "default";

    2 usages
    public int getNumber() { return number; }

    public void setNumber(int number) { this.number = number; }

    public void setName(String name) { this.name = name; }

    private void printData() {
        System.out.println(number + name); }
}
```

So what's the problem?

- ✓ **Add getter**
- ✓ **Change modifier**

Practice – get private field

```java
public class MainField {

    public static void main(String[] args) {
        MyClass myClass = new MyClass();

        int number = myClass.getNumber();
        String name = null; //no getter =(

        System.out.println(number + name);//output 0null

        //Reflection API - the dark side of java
        try {
            Field field = myClass.getClass().getDeclaredField( name: "name");

            field.setAccessible(true);

            field.set(myClass, "new value");

            name = (String) field.get(myClass);

        } catch (NoSuchFieldException | IllegalAccessException e) {
            e.printStackTrace();
        }
        System.out.println(number + name);
    }

}
```

Java has a wonderful class Class

- **GetFields()** - this method will return to us **all the available fields** of the class
- **getDeclaredFields()** - this method also returns an array of class fields, but now both private and protected
- **getDeclaredField(String)** method, where String is the name of the desired field

- **setAccessible(true)** - give access to work with field

Practice – get private method

```java
public class MyClass {
    3 usages
    private int number;
    2 usages
    private String name = "default";

    //    public MyClass(int number, String name) {
    //        this.number = number;
    //        this.name = name;
    //    }
    2 usages
    public int getNumber() { return number; }

    public void setNumber(int number) { this.number = number; }

    public void setName(String name) { this.name = name; }

    private void printData() {
        System.out.println(number + name); }
}
```

```java
//Reflection API - the dark side of java
2 usages
public static void printData(Object myClass){
    try {
        Method method = myClass.getClass().getDeclaredMethod( name: "printData");
        method.setAccessible(true);
        method.invoke(myClass);
    } catch (NoSuchMethodException | InvocationTargetException | IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

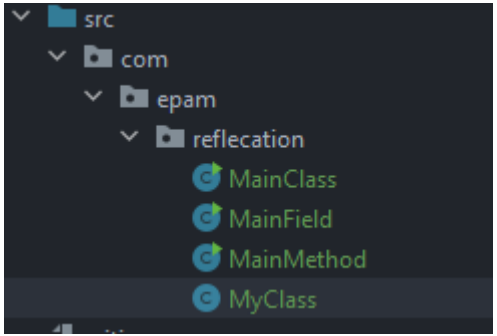
- **getDeclaredMethod("printData")** – get private method
- **method.invoke(myClass)** - call the Method object

Practice – get private method

```java
public static void main(String[] args) {
    MyClass myClass = new MyClass();

    int number = myClass.getNumber();
    String name = null; //no getter =(

    printData(myClass);//output 0default

    //Reflection API - the dark side of java
    try {
        Field field = myClass.getClass().getDeclaredField( name: "name");

        field.setAccessible(true);

        field.set(myClass, "new value");

        name = (String) field.get(myClass);

    } catch (NoSuchFieldException | IllegalAccessException e) {
        e.printStackTrace();
    }
    printData(myClass); //output 0new value
}
```

```java
//Reflection API - the dark side of java
2 usages
public static void printData(Object myClass){
    try {
        Method method = myClass.getClass().getDeclaredMethod( name: "printData");
        method.setAccessible(true);
        method.invoke(myClass);
    } catch (NoSuchMethodException | InvocationTargetException | IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

Practice – create class via reflection




```
public static void main(String[] args) {
    System.out.println(MyClass.class.getName());
}
```

com.epam.reflecation.MyClass

Practice – create class via reflection

```java
public class MainClass {

    public static void main(String[] args) {
        System.out.println(MyClass.class.getName()); //output com.epam.reflection.MyClass

        MyClass myClass = null;
        try {
            Class clazz = Class.forName(MyClass.class.getName());

            myClass = (MyClass) clazz.getDeclaredConstructor().newInstance();

        } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | NoSuchMethodException |
                InvocationTargetException e) {
            e.printStackTrace();
        }
        System.out.println(myClass);//output com.epam.reflection.MyClass@7c30a502
    }

}
```

When?

When do you need to use it?

Never!!!

There are three main disadvantages:

- Productivity is declining
- There are security restrictions
- Risk of disclosure of internal information

Thank you!

Links to resources and bio

Bio:
LinkedIn

Repository:
GitHub repository

Our community:
MJC
MJC-School
mjc.school
LinkedIn
LinkedIn group
YouTube channel

Do you have any questions?