Reflection in Java

MRINAL KANTI SAHA

Contents

Let’s see what we can do with Reflection…

Before we head into the theory part…

We have a class “Testclass”

Fields :-

private int private_field

public String public_field

public static int static_field

private final int final_field

Constructor :-

public Testclass()

Methods :-

private void func1()

public void func2()

How are we to know the **contents** of a class from only “Testclass.class” ?

Provided we do not have a decompiler.

Let’s see whether it can be done...

Now, can we **peep** into the values of an Object's **fields** at run-time ???

- ✓ public members are anyways visible.
- ✓ But, **private** & **protected** ?

- ✓ We can have a **generic** toString()

Change the values of **private** fields ?
Invoke the **private** methods ?

- ✓ C++ had **friend** function to do so.
- ✓ But, in JAVA , it is totally a **run-time** thing.

Let’s go for the definition now

- Reflection - an API, used to examine or **modify** the behavior of methods, classes, interfaces at **runtime**.
- class **Class** is the major class used with reflection.
- Rest all the necessary classes are under “ java.lang.reflect “.
- Introduced in JDK 1.1

Classes involved in Reflection

The **Class** class

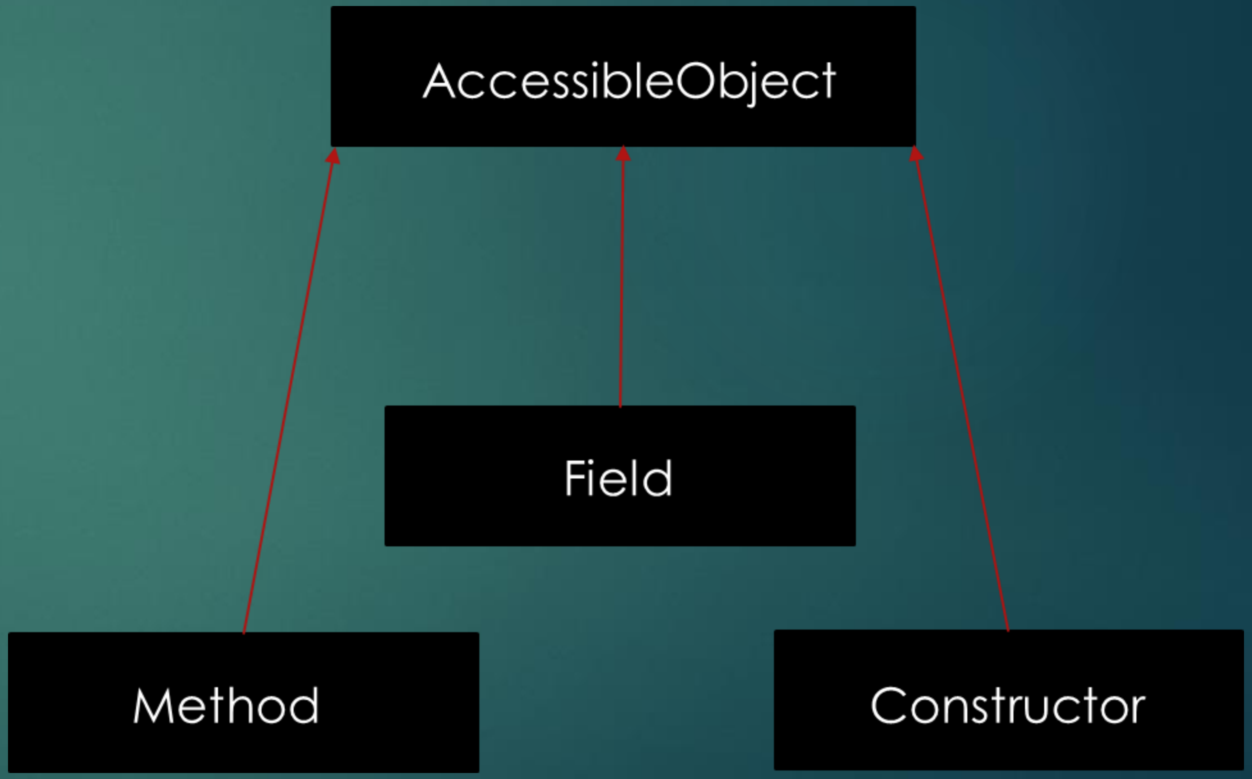
- Under java.lang package
- Universal type for the **metadata** that describes objects within the Java system.
- Methods :-
  - forName()
  - getClass()
  - newInstance()
  - getSuperClass()
  - getInterfaces()

Classes in java.lang.reflect

- Field
  - getDeclaredFields()
  - getModifiers()
  - get(obj)
  - set(obj, value)
- Method
  - getDeclaredMethods()
  - invoke(obj, parameters[])
- Constructor
  - getConstructors()
- AccessibleObject
  - setAccessible(boolean)

Classes in java.lang.reflect

- Field
  - getDeclaredFields()
  - getModifiers()
  - get(obj)
  - set(obj, value)
- Method
  - getDeclaredMethods()
  - invoke(obj, parameters[])
- Constructor
  - getConstructors()
- AccessibleObject
  - setAccessible(boolean)



FAQs

FAQs

- Is there no way to **curb** Reflection ?

FAQs

- Is there no way to **curb** Reflection ?

SecurityManager manages access to the private members of an object.

With the SecurityManager turned on, using Reflection on private or protected, will lead to **SecurityException**.

FAQs

- Does reflection break all the **myths** about data security via abstraction and access modifiers ?

FAQs

- Does reflection break all the **myths** about data security via abstraction and access modifiers ?

Yes and **No**

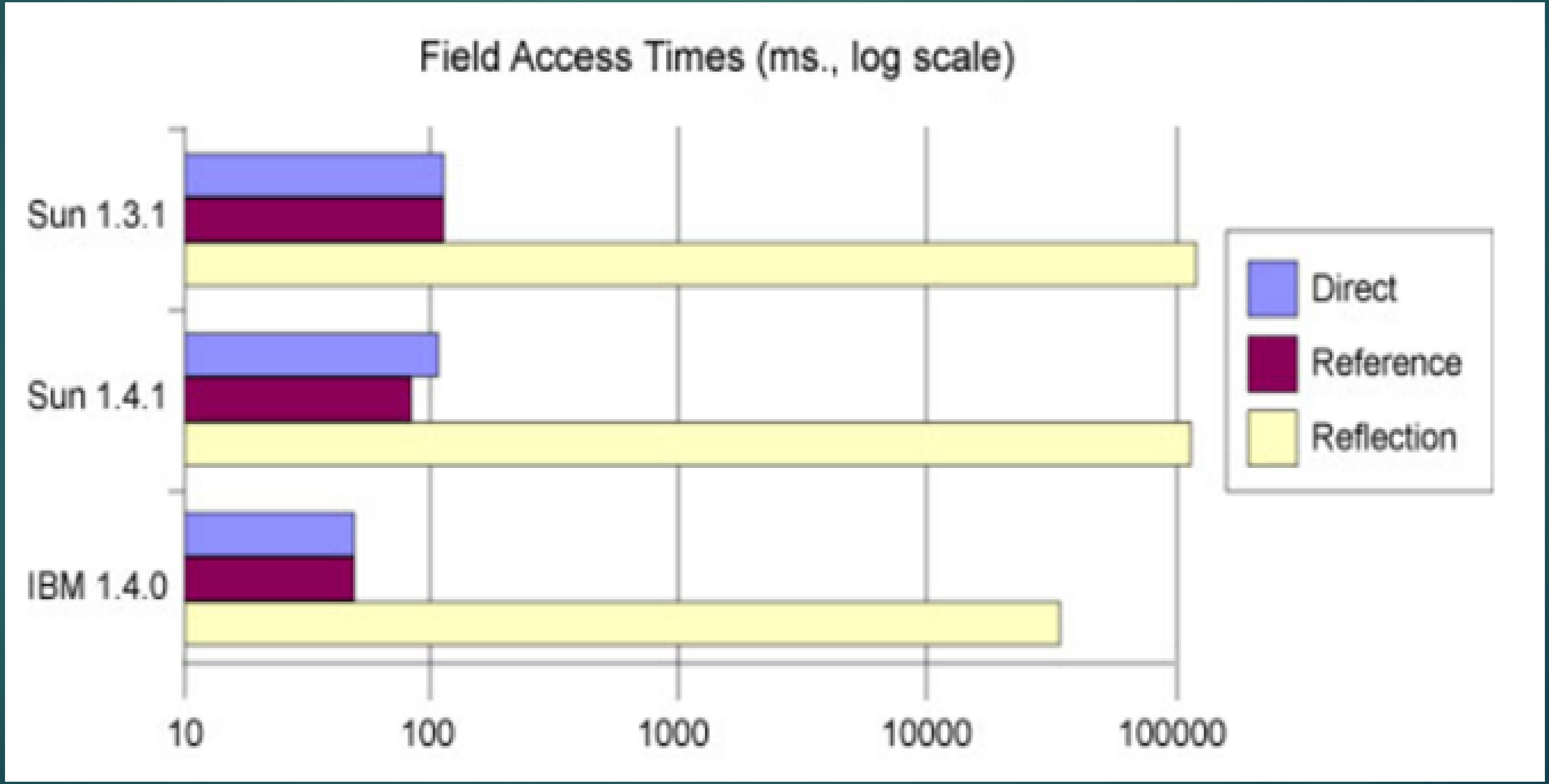
Why is it not used commonly ?

- Performance Overhead
- Security Restrictions
- Exposure of Internals

Where is it used ?

- Debugging tool (recall GDB)
- JDBC (Java DataBase Connectivity)
- JavaBeans
- Visual Development Environment
- Java based Web Servers (e.g. Tomcat)

Performance Overhead (Field access)



References

- GeeksforGeeks
- Core Java, Volume I
- JournalDev
- JenkovPoint
- IBMdeveloperWorks

Thank You…