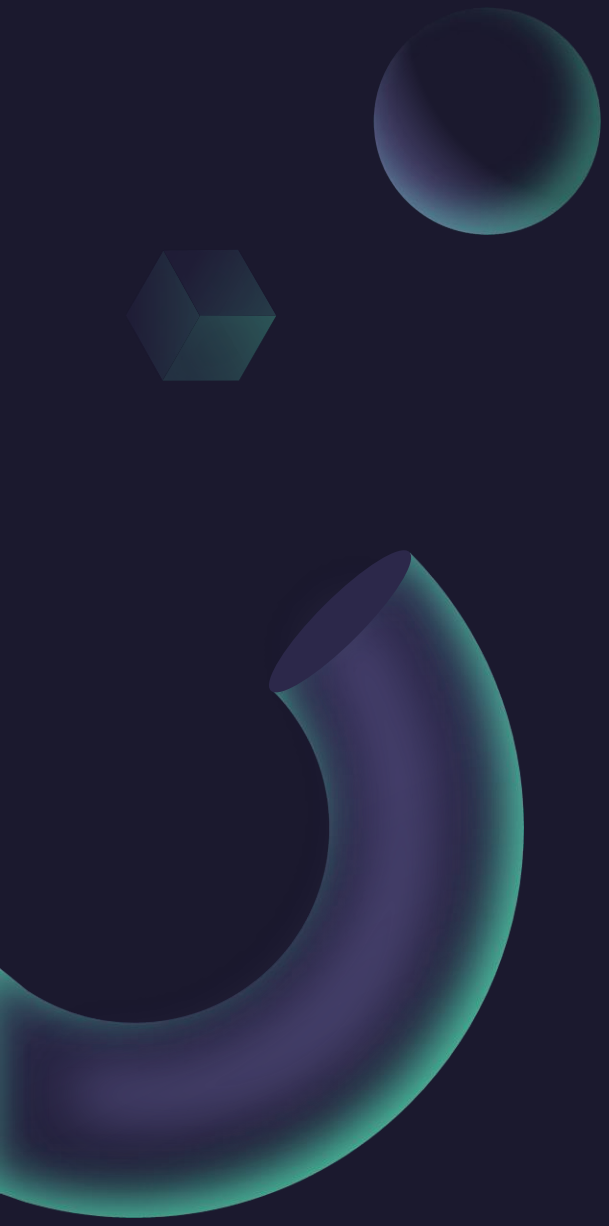
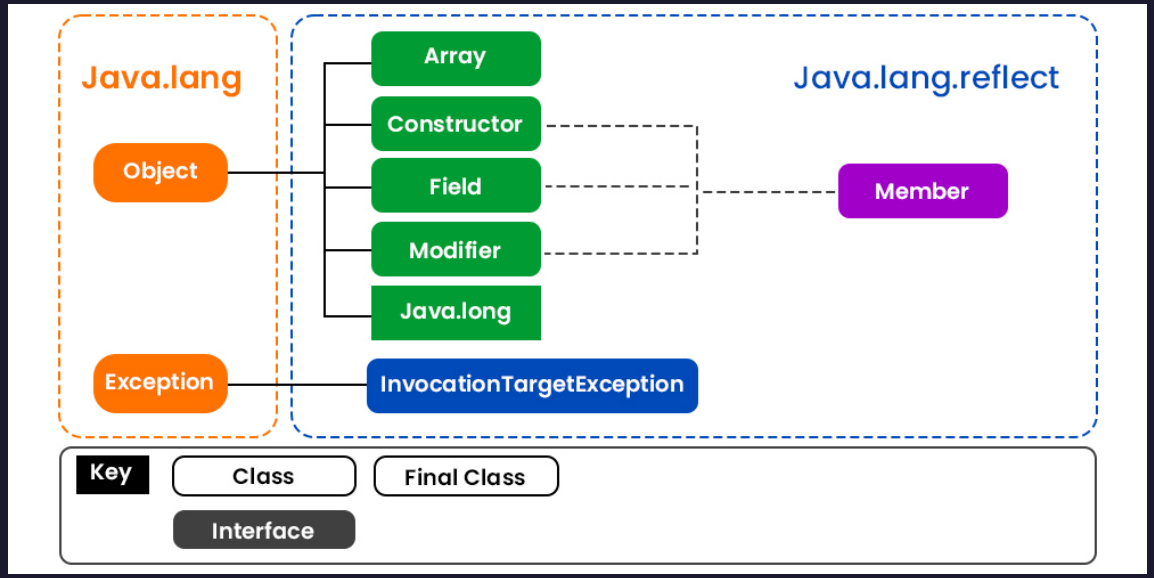


Java Reflection

Definition

"Reflection is an API that is used to examine or modify the behavior of methods, classes, and interfaces at runtime. The required classes for reflection are provided under **java.lang.reflect** package which is essential in order to understand reflection"



https://www.geeksforgeeks.org/reflection-in-java/

Wichtige Bausteine

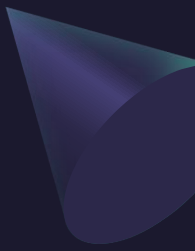
Grundlagen und Class

Method

Constructor

Field

Modifier



Grundlagen und Class

Wichtiger Import:

- import java.lang.reflect.*
- import java.lang.Class (Standard – zum Verständnis erwähnt)

Drei Schritte, um eine Reflection zu realisieren:

1. Zu manipulierendes Objekt erhalten
2. Methode auf das Objekt anwenden und Informationen erhalten
3. Reflection-API verwenden und Informationen manipulieren

```java
import java.lang.reflect.Method;
import java.lang.Class;

public class Method {
    public static void main(String args[])
    {
        try {
                Class c = Class.forName("java.lang.String");
                Method m[] = c.getDeclaredMethods();
                System.out.println(m[0].toString());
            }
        catch (Throwable e) {
            System.err.println(e);
        }
    }
}
```

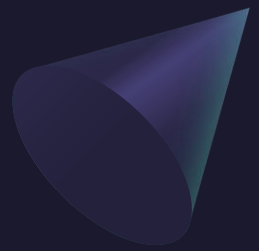
Method

- Ermöglicht es, alle deklarierten Methoden einer Klasse abzufragen
- Lässt sich mit Modifier kombinieren und gibt dann Auskunft über Sichtbarkeit

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

Method[] methods_feld = grossesFeld.getDeclaredMethods();

for (Method method : methods_grossesFeld) {
            System.out.println(method);
        }
```



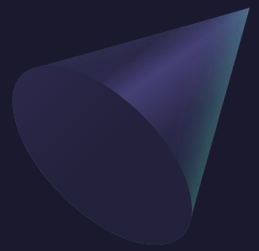
Constructor

- Ermöglicht es, alle deklarierten Konstruktoren einer Klasse abzufragen
- Durch geeignete Methoden der Klasse lassen sich zusätzlich Parameter und Exceptions abfragen

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

Class cnst = Class.forName("constructorname");
Constructor cnstlst[] = cnst.getDeclaredConstructors();

Constructor ct = ctorlist[1];
System.out.println("name = " + ct.getName());
System.out.println("decl class = " + ct.getDeclaringClass());
```

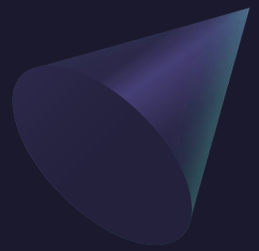


Field

- Ermöglicht es, definierte Datenfelder einer Klasse zu erhalten
- Klassische Datenfelder sind Variablen mit double, int, String….

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

Field feld = grossesFeld.getDeclaredField("Ackerflaeche");
```

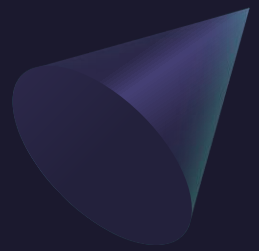


Modifier

- Ermöglicht einen Einblick in die Zugriffsrechte/Sichtbarkeit von Klassen und Objekten
- Public, privat, protected
- Durch geeignete Methoden lassen sich Zugriffsrechte mit Hilfe von Reflection ändern

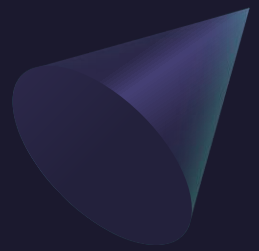
```
import java.lang.reflect.Modifier;

System.out.println(object.getModifiers());
```



Anwendungsfälle

- Verwendung von GUIs
- Debugging- und Testwerkzeuge
- Sicherheit (Introspektion von Code und prüfen auf Schwachstellen)
- Allgemeine Codeanalyse



Quellen

- https://www.oracle.com/technical-resources/articles/java/javareflection.html
- https://stackoverflow.com/questions/37628/what-is-reflection-and-why-is-it-useful
- https://stackoverflow.com/questions/49737/use-cases-for-reflection
- https://www.newthinktank.com/2012/09/java-reflection-video-tutorial/
- https://www.delftstack.com/de/howto/java/reflection-in-java/
- https://www.geeksforgeeks.org/reflection-in-java/