

# Assertions

Syntax: `assert <Bedingung> [: <Beschreibung>]`

```
package assertions;

public class StartAssertions {

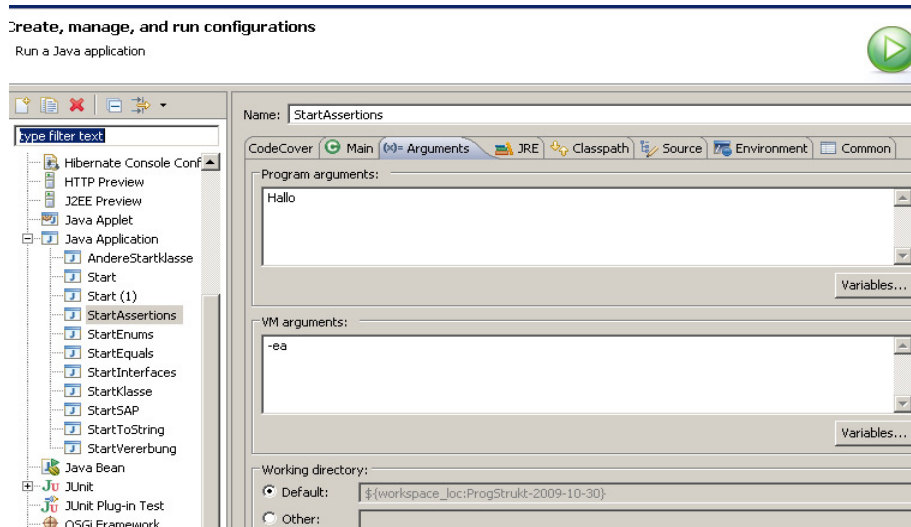
    public static void main(String[] args) {
        System.out.println("Das erste Argument ist " + args[0]);
        StartAssertions ich = new StartAssertions();
        System.out.println("Ich teile 4 durch 0: " + ich.teile(4, 0));
    }

    public int teile(int zaehler, int nenner){
        assert nenner != 0 : "He uffpasse, der Nenner darf nicht 0 sein!";

        return zaehler/nenner;
    }
}
```

Falls die Bedingung NICHT erfüllt ist, wird ein `AssertionError` geworfen und die Beschreibung ausgegeben.

Die Assertions müssen über das JVM-Argument -ea aktiviert werden.



Auf der Kommandozeile würde unser Programm wie folgt gestartet werden:

```
java -ea assertions.StartAssertions Hallo
```

↖  
JVM- Parameter

↖  
Programmparameter (hier nicht notwendig)

# Reflection & Introspection

Reflection/Introspection wird eingesetzt, um

- zur Laufzeit (!) Objekte zu untersuchen
- zur Laufzeit (!) dynamisch Objekte zu erzeugen (instanzieren) und Methoden aufzurufen (die während der Entwicklungszeit noch nicht festgelegt werden)

Bewertung

- + Dynamik (Anwendung siehe unten)
- keine Compilersicherheit
- Performance (z.B. Optimierung durch Hotspot Compiler)

## Anwendung

- Konfiguration von Programmen (Datenbank, Treiber, Auswahl Interfaces, Auswahl "Look & Feel")
- Verteilte Anwendungen mit RPC (Remote Procedure Call = Aufruf entfernter Methoden)

```

public static void main(String[] args) {
    String className = args[0];
    String methodName = args[1];
    String uebergabewertKonstruktor = args[2];
    String uebergabewertMethode = args[3];

    try {
        //Klasse suchen
        Class unbekannteKlasse = Class.forName(className);

        //Passenden Konstruktor finden
        Class[] parameterTypes = new Class[]{String.class};
        Constructor constructor = unbekannteKlasse.getConstructor(parameterTypes);

        //Klasse instanzieren, dabei dem Konstruktor die Werte übergeben
        Object[] initargs = new Object[]{uebergabewertKonstruktor};
        MaxKlasse1 maxklasse = (MaxKlasse1)constructor.newInstance(initargs); //Inkonsistenz, lösen nachher auf!

        //Alle Methoden ausgeben
        Method[] alleMethoden = unbekannteKlasse.getDeclaredMethods();
        for(Method methode : alleMethoden){
            System.out.println("Wir haben eine Methoden gefunden und die heißt: " + methode.getName());
        }

        //Methode aufrufen
        Class[] parameterTypes2 = new Class[]{String.class};
        Method unbekannteMethode = unbekannteKlasse.getMethod(methodName, parameterTypes2);
        Object[] uebergabewerte = new Object[]{uebergabewertMethode};
        unbekannteMethode.invoke(maxklasse, uebergabewerte);

    } catch (ClassNotFoundException e) {
        // T020.java-generated catch block
    }
}

```