Applause from Cory Althoff, Tariq Ellis, and 1 other

Adrian D. Finlay
@thewiprogrammer. Writer @hackernoon. Code, LOTS of it. Mangos, LOVE THEM! Barbering. Health. Travel. Business. & more! Network w/ me @ adriandavid.me/network
Oct 23, 2017 · 4 min read

Assertions in the Java Programming Language

Giphy: Pink & The Brain

Assertions are a development tool and programming language feature used to check if a conditional expression evaluates to true when the program is run. They are useful in the testing and development process and are typically omitted for production code. A key reason for this is that they require command line flags to be enabled, thus limiting proper functioning portability if said command line flags are enabled. As such, they are not recommended for production code; they were not designed for this. They can replace situations where programmers used to check for with if statements. They are much more compact then if statements and if assertions are not enabled at run-time, they are ignored. Assertions are accomplished in Java with the **assert** keyword.

An **AssertionError** is thrown if the condition evaluates to false. We have **Joshua Bloch** to thank for this language feature.

To configure assertion options one must use either the -ea or -da command line flags to enable or disable assertions with the command line tool: “java”. For example, “java -ea Assert” where Assert is a java class file. You may also specify a specific class or package as follows. For a class: “-ea:Class”, “-ea:Package/Class”. For a package (and it’s sub-packages): “-ea:Package…”. Notice the ellipses (three succeeding periods), they are part of the format.

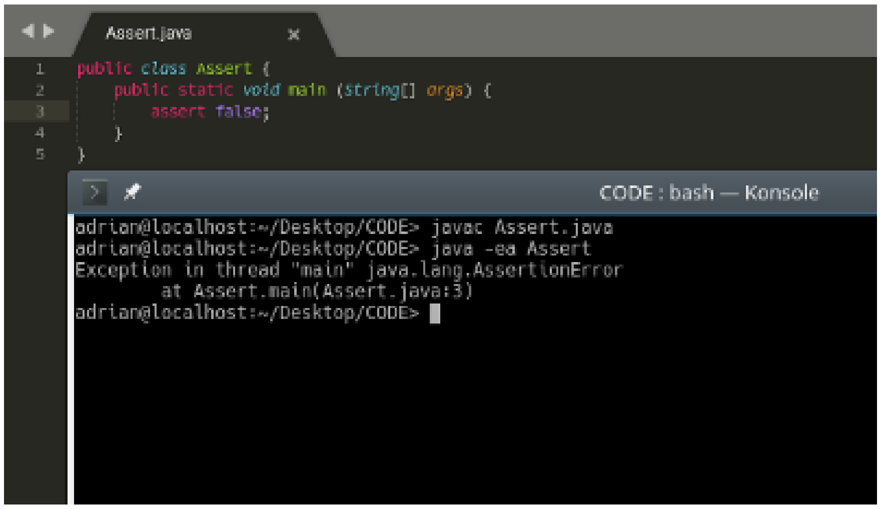
Within java code assertions take the following two forms:

- assert condition;
- assert condition: expression;

In the first form, condition is a conditional expression in java that results to True or False. In the second form, an expression is executed if the condition is false. This may be any expression that is not of type void. **The String representation of the result of this Expression will be used in forming the AssertionError object.** In both cases, an AssertionError is thrown if the conditional expression results to false.

Assertion Examples

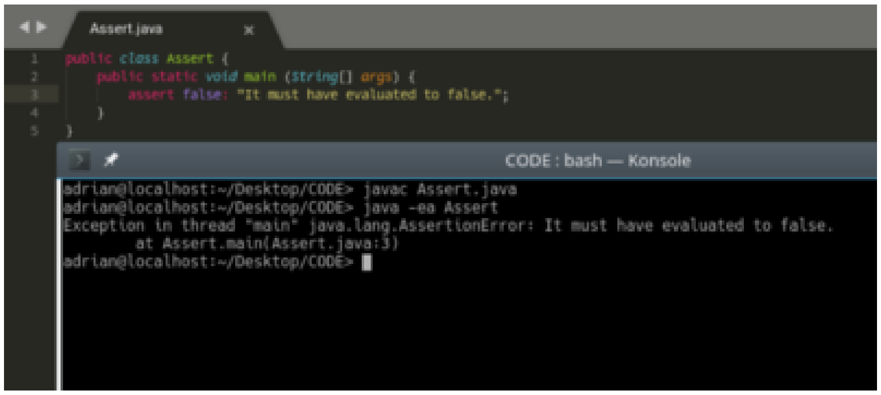
Basic Assertion usage



```java
public class Assert {
    public static void main (String[] args) {
        assert false;
    }
}
```

```
adrian@localhost:~/Desktop/CODE> javac Assert.java
adrian@localhost:~/Desktop/CODE> java -ea Assert
Exception in thread "main" java.lang.AssertionError
        at Assert.main(Assert.java:3)
adrian@localhost:~/Desktop/CODE>
```

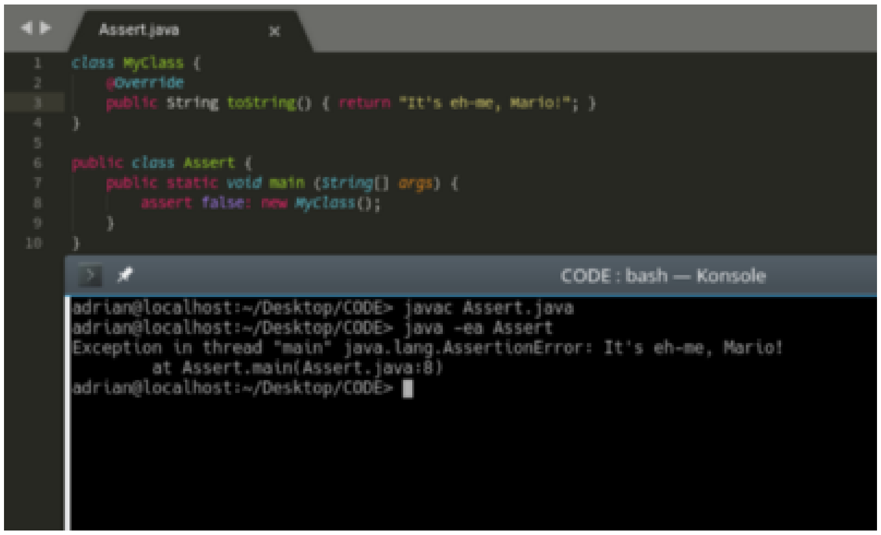
Basic Assertion usage with example of Expression added



```java
public class Assert {
    public static void main (String[] args) {
        assert false: "It must have evaluated to false.";
    }
}
```

```
adrian@localhost:~/Desktop/CODE> javac Assert.java
adrian@localhost:~/Desktop/CODE> java -ea Assert
Exception in thread "main" java.lang.AssertionError: It must have evaluated to false.
        at Assert.main(Assert.java:3)
adrian@localhost:~/Desktop/CODE>
```

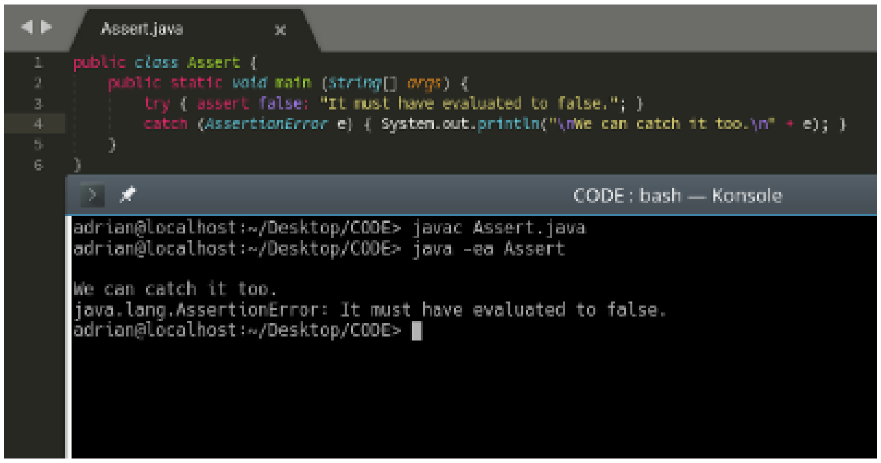
Basic Assertion usage with Custom Class instance used as expression



```java
class MyClass {
    @Override
    public String toString() { return "It's eh-me, Mario!"; }
}

public class Assert {
    public static void main (String[] args) {
        assert false: new MyClass();
    }
}
```

```
adrian@localhost:~/Desktop/CODE> javac Assert.java
adrian@localhost:~/Desktop/CODE> java -ea Assert
Exception in thread "main" java.lang.AssertionError: It's eh-me, Mario!
        at Assert.main(Assert.java:8)
adrian@localhost:~/Desktop/CODE>
```

Catching the AssertionError exception object generated by assert



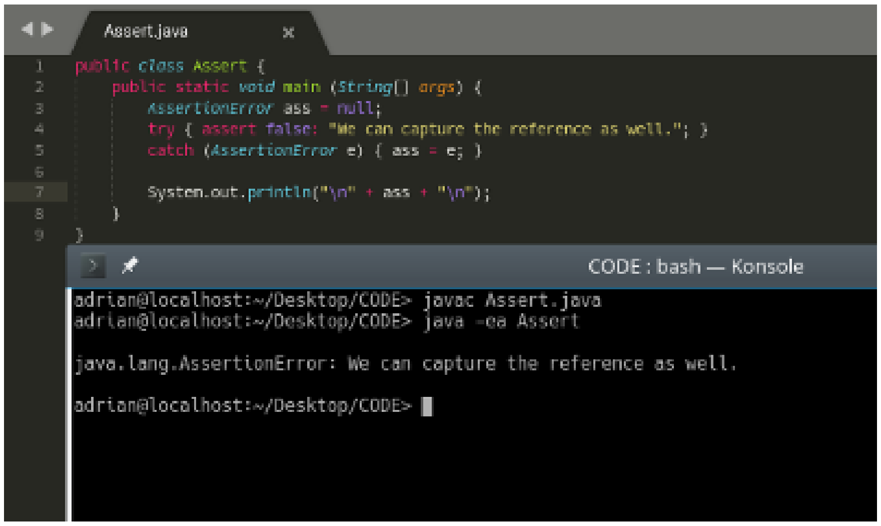
Assert.java

```java
public class Assert {
    public static void main (String[] args) {
        try { assert false: "It must have evaluated to false."; }
        catch (AssertionError e) { System.out.println("\nWe can catch it too.\n" + e); }
    }
}
```

CODE : bash — Konsole

```
adrian@localhost:~/Desktop/CODE> javac Assert.java
adrian@localhost:~/Desktop/CODE> java -ea Assert

We can catch it too.
java.lang.AssertionError: It must have evaluated to false.
adrian@localhost:~/Desktop/CODE>
```

Capturing the AssertionError object for use outside of try-catch



Assert.java

```java
public class Assert {
    public static void main (String[] args) {
        AssertionError ass = null;
        try { assert false: "We can capture the reference as well."; }
        catch (AssertionError e) { ass = e; }

        System.out.println("\n" + ass + "\n");
    }
}
```

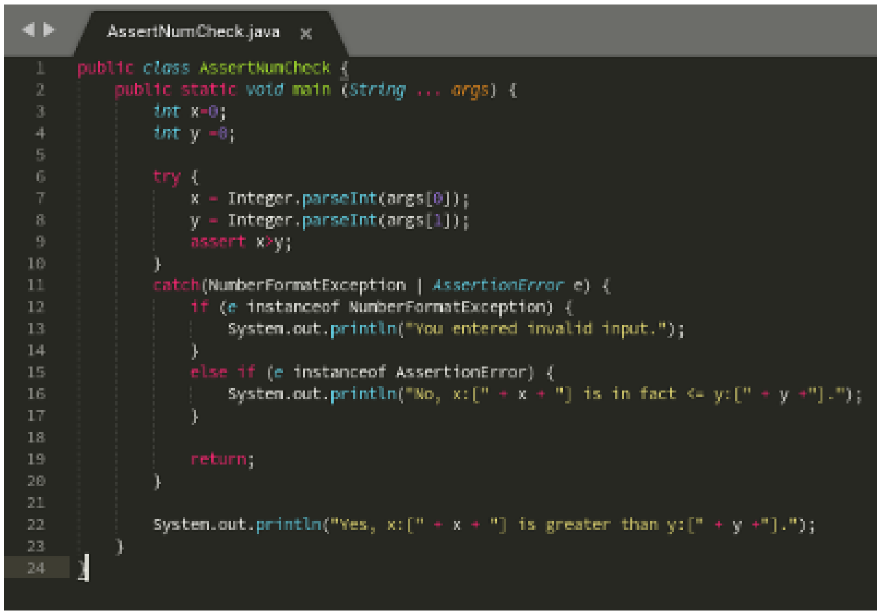
CODE : bash — Konsole

```
adrian@localhost:~/Desktop/CODE> javac Assert.java
adrian@localhost:~/Desktop/CODE> java -ea Assert

java.lang.AssertionError: We can capture the reference as well.

adrian@localhost:~/Desktop/CODE>
```

Note!: For the purpose of completeness, we have shown how to catch an AssertionError. However, **you should never attempt to catch Errors**, only Exceptions, if needed!

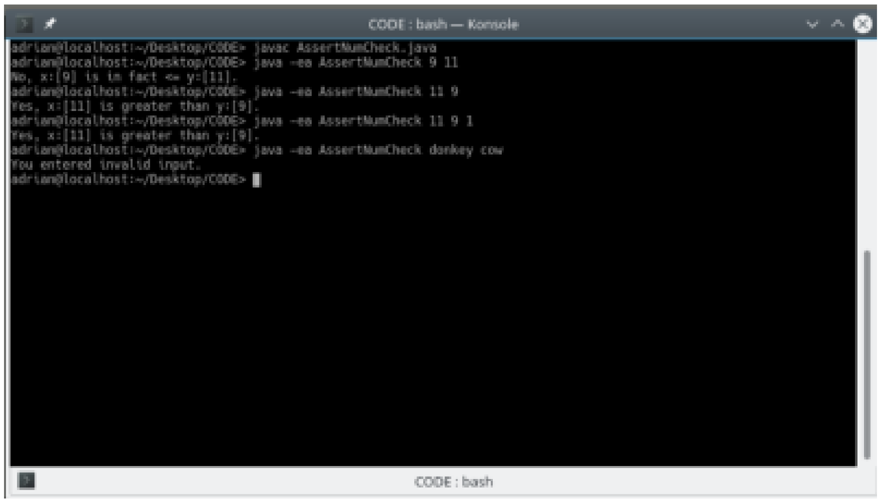
Program that parses command line arguments and checks against a condition. **Remember:** this is advised against for production code.

Source



```java
public class AssertNumCheck {
    public static void main (String ... args) {
        int x=0;
        int y =0;

        try {
            x = Integer.parseInt(args[0]);
            y = Integer.parseInt(args[1]);
            assert x>y;
        }
        catch(NumberFormatException | AssertionError e) {
            if (e instanceof NumberFormatException) {
                System.out.println("You entered invalid input.");
            }
            else if (e instanceof AssertionError) {
                System.out.println("No, x:[" + x + "] is in fact <= y:[" + y +"].");
            }

            return;
        }

        System.out.println("Yes, x:[" + x + "] is greater than y:[" + y +"].");
    }
}
```

Ouput



```
adrian@localhost:~/Desktop/CODE> javac AssertNumCheck.java
adrian@localhost:~/Desktop/CODE> java -ea AssertNumCheck 9 11
No, x:[9] is in fact <= y:[11].
adrian@localhost:~/Desktop/CODE> java -ea AssertNumCheck 11 9
Yes, x:[11] is greater than y:[9].
adrian@localhost:~/Desktop/CODE> java -ea AssertNumCheck 11 9 1
Yes, x:[11] is greater than y:[9].
adrian@localhost:~/Desktop/CODE> java -ea AssertNumCheck donkey cow
You entered invalid input.
adrian@localhost:~/Desktop/CODE> 
```

And much more. Play around with it and let me know how you like it :)

Want the source? Grab it here.

afinlay5/JavaAssertions

JavaAssertions - Java source code example demonstrating assertions, posted on personal bl...

github.com

Interested in Java? Join my Java group on Facebook:

Join My Java Facebook Group

Interested in Java? Check out my Facebook Group: Java Software Development Group!

medium.com

Like my Content? Subscribe to my mailing list:

This embedded content is from a site that does not comply with the Do Not Track (DNT) setting now enabled on your browser.

Please note, if you click through and view it anyway, you may be tracked by the website hosting the embed.

Learn More about Medium's DNT policy

Don’t forget to give it a.... ;)

IEmoji.com

No **Works Cited** this time ;).