G18 - Project I

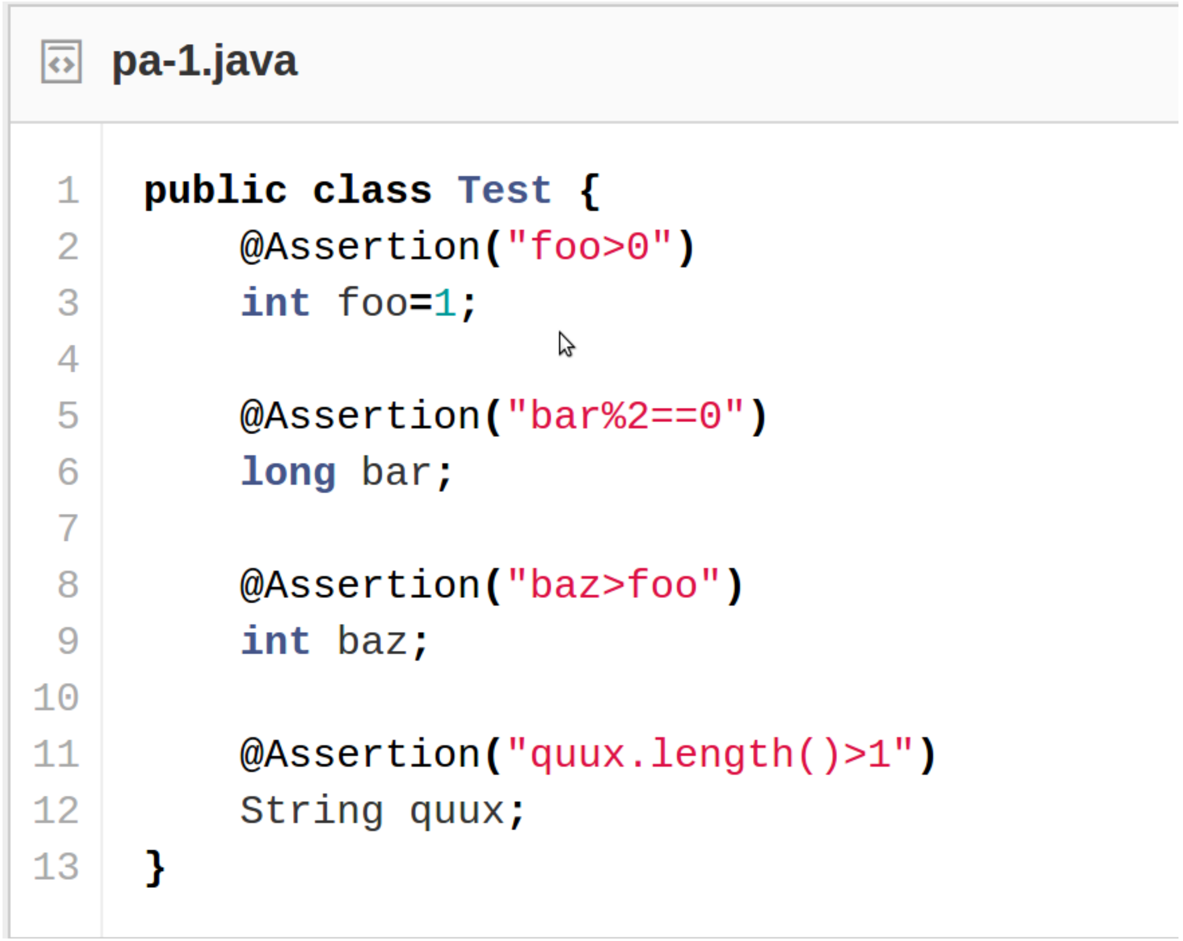
Check-Assertions on Java

TÉCNICO LISBOA

Field Assertions

We have to deal with:

- An assertion can be any kind of expression.
- it must have full context of class which belongs.
- Every assertion should be called before the code block when a write is invoked
- Despite of non initialization check from Java Perspective we have to control when it happens



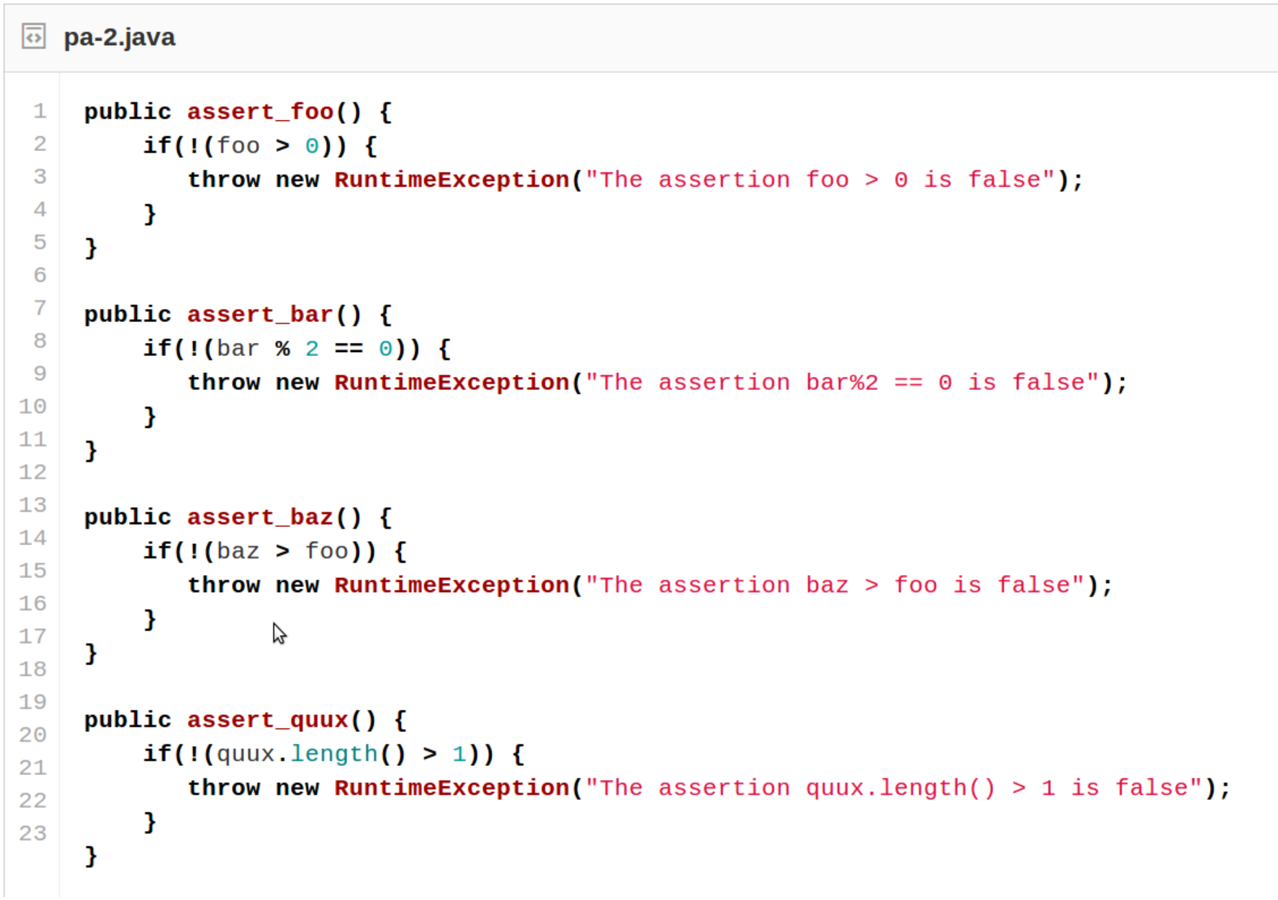
pa-1.java

```java
public class Test {
    @Assertion("foo>0")
    int foo=1;

    @Assertion("bar%2==0")
    long bar;

    @Assertion("baz>foo")
    int baz;

    @Assertion("quux.length()>1")
    String quux;
}
```

Field Assertions

In order to have full access over class attributes we opted to introduce self-modification on attributes with annotation Assertion.

When a class is loaded by javassist ClassLoader, our Field Inspector injects code snippets on the loaded class and before completes a write it calls for the respective method already created

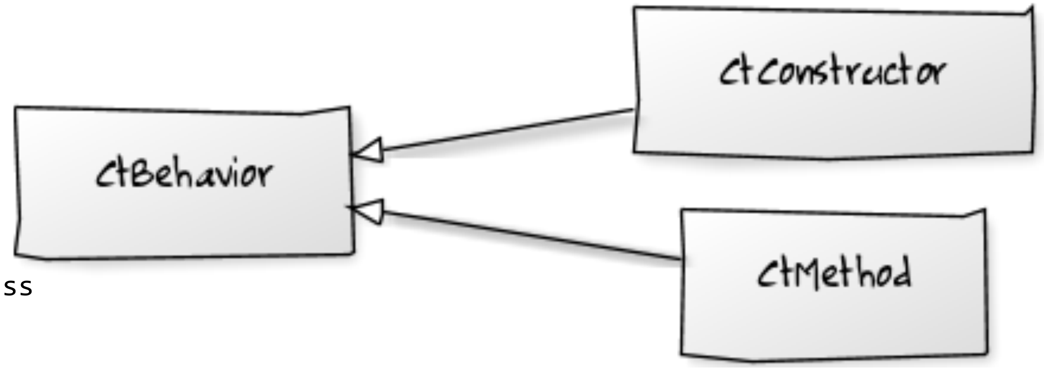

pa-2.java

```java
public assert_foo() {
    if(!(foo > 0)) {
       throw new RuntimeException("The assertion foo > 0 is false");
    }
}

public assert_bar() {
    if(!(bar % 2 == 0)) {
       throw new RuntimeException("The assertion bar%2 == 0 is false");
    }
}

public assert_baz() {
    if(!(baz > foo)) {
       throw new RuntimeException("The assertion baz > foo is false");
    }
}

public assert_quux() {
    if(!(quux.length() > 1)) {
       throw new RuntimeException("The assertion quux.length() > 1 is false");
    }
}
```

Field Assertions - Initialization

Usage of CtBehavior and Why?

-> CtBehavior represents a method, a constructor, or a static constructor (class initializer)



Check field initialization
- FieldMapper:

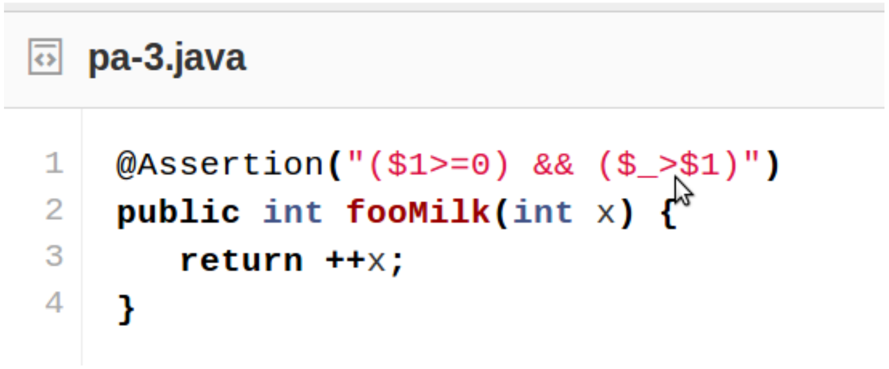
Class used to make Structural Reification. Every write on a asserted field adds a column on the table for that instance assuring that the variable on use is already initialized

Method & Constructor Assertions

PROBLEM

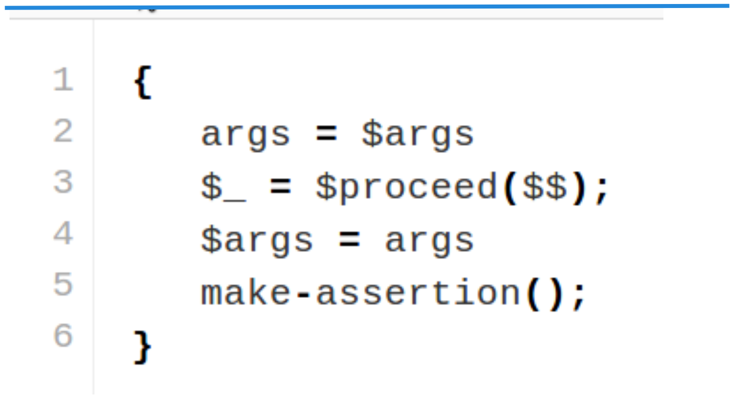
Apparently all should be ok if we just add a code snippet after the call....but....it will not work.

The stack of the program context can be changed, so x will increment it's value and the final assertion will be wrong.



pa-3.java

```java
@Assertion("($1>=0) && ($_>$1)")
public int fooMilk(int x) {
    return ++x;
}
```



```
{
    args = $args
    $_ = $proceed($$);
    $args = args
    make-assertion();
}
```

IMPLEMENTATION

We store the entry parameters on a temporarly variable and after the method rans we store that values again on $args to make assertions without problems.

THE SOLUTION ISN'T THREAD SAFE :(

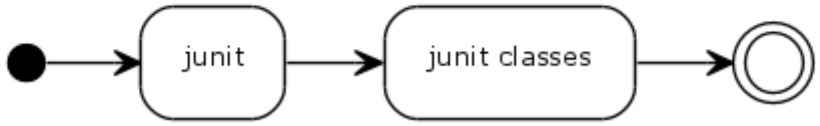
Unit Tests

WHY JUNIT?

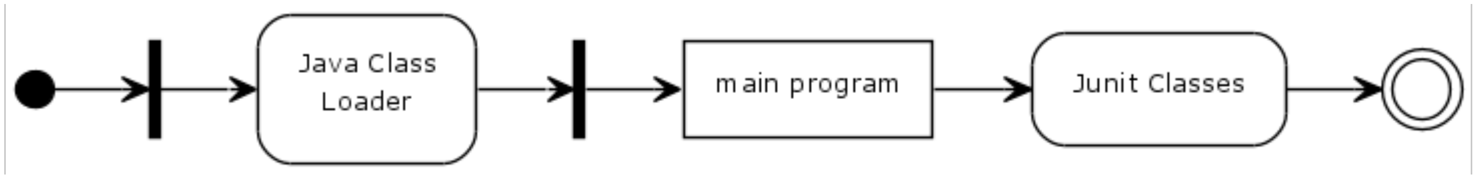
Testing was the most annoying part of the development phase. To increase the process we used Junit Framework to test different features at the same time.

Unfortunatly Junit with Javassist out of the box doesn't work expected.

WRONG WAY



RIGHT WAY TO MAKE JUNIT WORK


Java Class Loader
main program
Junit Classes