Evaluation of a tool for Java specification testing

Conference or Workshop Item

How to cite:


For guidance on citations see FAQs.

© 2018 The Authors

https://creativecommons.org/licenses/by-nc-nd/4.0/

Version: Version of Record

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.
Java specification checking

Evaluation of a tool for use on M250 “Object-oriented Java Programming”

Anton Dil and Sue Truby
In the public class M250Account:

i. Declare a private instance variable of type String called accountNum.

ii. Edit the constructor for M250Account so that it initialises the accountNum instance variable to the account number "X00000".

iii. Write a getter method for the accountNum instance variable.

iv. A valid account number must have a length of 6 characters. Write a public method isValidLength(String s) that returns true if s is the correct length, and otherwise returns false.

v. A valid account number must begin with a capital letter. Write a public method isValidStart(String s) that returns true if s begins with a capital letter, and otherwise returns false.

vi. In a valid account number, all the characters after the first (capital) letter must be digits. Write a public method hasValidDigits(String s) that tests whether s meets this condition and returns true or false accordingly.

vii. Write a public method isValidAccountNum(String s) that performs the three checks above, and returns true if the account number represented by s is valid, and otherwise returns false.
1. The instance variable `accountNo` not in the specification
2. The instance variable `accountNo` is missing the access modifier `private`
3. The instance variable `s` is not in the specification
4. The method `isValidAccount` is not in the specification – it should be called `isValidAccountNum`
5. The method `isValidAccountNum` should have an argument `s` of type `String`
6. The method `hasValidDigits()` is missing
7. Methods `isValidLength()` and `isValidAccount()` perform incorrect tests
8. Curly brace formatting is inconsistent
9. There are no comments – poor style
What’s available to us to perform automated tests?

What automated tests could help?

1. Does it compile?
2. Does it pass a style check?
3. Does it pass unit tests?
4. **Does it meet the English specification?**

What tools could we use?

1. The compiler, if tutors use it – but they may not.
2. Static style checkers are available – a style sheet is needed.
3. A unit testing framework (JUnit) is available – a test harness is needed.
4. **CheckM250 prototype**
“CheckM250” specification checking tool – Research questions

1) Find out how tutors feel about this, and about use of similar tools – particularly style checking and unit testing.
   • With respect to marking
   • With respect to student use
   • Does it make marking faster?
   • Does it change the focus of marking?
   • Does it find errors tutors would have missed?

2) Explore use in Moodle for students, under CodeRunner
   • Requires code adaptations for the VLE

3) Later I decided to add compilation error help.
Design of the tool

How the tool works
Question authoring using “CheckM250” tool

1. The question author writes a *more formal* specification according to a simple specification language, based on the English specification and a provided solution class from the module team

```java
class:public/M250Account/java.lang.Object/
field:private/java.lang.String/M250Account.accountNum
method:public/java.lang.String/M250Account.getAccountNum/void
method:public/boolean/M250Account.isValidLength/java.lang.String
method:public/boolean/M250Account.isValidStart/java.lang.String
method:public/boolean/M250Account.hasValidDigits/java.lang.String
method:public/boolean/M250Account.isValidAccountNum/java.lang.String
method:public/void/M250Account.setAccountNum/java.lang.String
constructor:public/M250Account/void
```

2. The Specification Checker compares this specification with the student’s solution
   – what’s missing; what’s excess in the solution?
   – Reports on the results
   – student’s code needs to compile

3. Unintended benefit: Cross-checking what the question asked for – did it ask for everything we wanted?
Running the tool: Found some errors!

Class M250Account
Inheritance relationships ✓
Implements relationships ✓

Step 4: Checking requirements for methods...
In class M250Account, the method isValidAccountNum(java.lang.String) should have return type boolean ✗
In class M250Account, the method isValidLength(java.lang.String) should have return type boolean ✗
In class M250Account, the public method getAccountNumber is not in the specification ✗. If you want a helper method, make it private.
In class M250Account, the required method with signature getAccountNum() is missing ✗
Check that you have methods with the required names and signatures.

Step 5: Checking requirements for fields...
All the required fields were present ✓

Step 6: Checking requirements for constructors...
In class M250Account, the required constructor with signature M250Account() is missing ✗

All the checks were completed. Please see the advice above.

Information about code in your Java project:

Tue Oct 24 14:10:03 BST 2017
Methods defined in the solution classes
In class M250Account, public boolean isValidStart(String) ✓
In class M250Account, public void setAccountNum(String) ✓
In class M250Account, public int isValidLength(String) ✓
In class M250Account, public String getAccountNumber() ✓
In class M250Account, public boolean hasValidDigits(String) ✓
In class M250Account, public String isValidAccountNum(String) ✓

Fields defined in the solution classes

Select Project  TMA01_Project_Q1_Sol-locl is selected.
Run Checks
Student facing tool
Green ticks and red crosses are motivating!

Typical CodeRunner output for a partially correct solution

```java
// Now testing Car class
// tests constructor with newly created Engine argument
// should be able to start the engine, but then not start it again (as already running)
Engine e = new Engine();
Car c = new Car(e);
c.start();
c.start();

// Now testing Car class
// tests constructor with newly created Engine argument
// should be able to start the engine, and then accelerate up to the maximum revs
// and then blow up the engine
Engine e = new Engine();
Car c = new Car(e);
c.start(); // revs 1000
System.out.println(c.getRevs()); // 1000
```
### Example “Precheck” in Coderunner

Discrepancy in expected output for Specification Check

<table>
<thead>
<tr>
<th>Expected</th>
<th>Got</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiled OK</td>
<td>Compiled OK</td>
</tr>
<tr>
<td>Specification OK</td>
<td>In class Animal, the required method with signature equals(java.lang.O</td>
</tr>
<tr>
<td></td>
<td>bject) is missing</td>
</tr>
<tr>
<td></td>
<td>In class Animal, the required method with signature hashCode() is miss</td>
</tr>
<tr>
<td></td>
<td>ing</td>
</tr>
<tr>
<td></td>
<td>In class Shelter, the required method with signature getAnimals() is</td>
</tr>
<tr>
<td></td>
<td>mising</td>
</tr>
<tr>
<td></td>
<td>In class Shelter, the required method with signature homed(int, java.</td>
</tr>
<tr>
<td></td>
<td>Object)</td>
</tr>
<tr>
<td></td>
<td>In class Shelter, the required method with signature inTheLastMonth(</td>
</tr>
<tr>
<td></td>
<td>Animal)</td>
</tr>
<tr>
<td></td>
<td>In class Shelter, the required method with signature showRecentAnimals</td>
</tr>
<tr>
<td></td>
<td>r()</td>
</tr>
<tr>
<td></td>
<td>Check that you have methods with the required names and signatures.</td>
</tr>
<tr>
<td></td>
<td>In class Shelter, the required field private Map animals is missing</td>
</tr>
<tr>
<td></td>
<td>Specification errors found: 7</td>
</tr>
</tbody>
</table>
Results

Some results from the survey
So far…

• 8 tutors completing a survey reported that they used the tool
  • Maybe 10 used it

• 20 tutors responded to the survey (12 of whom did not use the tool)

• 6 interviews with tool users were conducted

• **Compilation help** was deployed on one formative Coderunner quiz on Moodle (500 attempts; 102 submitted)

• **Compilation help + Specification checking** was deployed on another formative Coderunner quiz (200 attempts so far; 35 submitted)
Why was it not used by some?

• **Insufficient time.** I had three TMA deadlines in the period 7-14 December and I had to focus on getting the marking done not the optional extras.

• **Little time to go and look for it, time to find how to use it** and **I thought it was something the module team was using, not for a tutor.**

• **I was busy in other areas.** I downloaded it and tried it out **but didn't feel I needed it for TMA1**

• **Didn't really want to spend any significant time in familiarising myself with a tool when the TMA was sufficiently simply to mark.**

• **I can usually find errors by eye.**

• **Its purpose has not been explained to me.** **At this level I can anyway spot coding errors without it.**

• **Did not know about it.**

• **Despite reading the tutor forums and viewing the marking guidelines I did not discover this tool until after marking TMA 01**

• **Wasn't aware of its existence**
Feedback

- 1/8 tutors would not be likely to recommend the tool to other tutors
  - Reported reason: it slowed down marking
- Most tutors very positive about the tool, particularly those interviewed.

10.1 Would recommend

- Not likely: 1 (12.5%)
- Somewhat likely: 2 (25%)
- Very likely: 5 (62.5%)
Ideally I’d have done a test where tutors record the errors they observed and compare that with the tool’s output.
What interviewees said

• There were errors I hadn’t spotted, which was embarrassing
• That’s what computers are useful for as far as I’m concerned.
• I would have liked to say to you that I’d spotted all those errors anyway, but I would be lying.
• When I initially looked at the code I couldn’t see why [the tool flagged an error], but then when I looked a little closer you found there was an issue
• If you’d asked me previously do you ever miss things I’d have gone ‘of course not’
<table>
<thead>
<tr>
<th>Theme</th>
<th>Negative</th>
<th>Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Time available to engage with the tool</td>
<td>No time to use, or slows me down</td>
<td>Worth investing the time</td>
</tr>
<tr>
<td>2 Quality of marking</td>
<td>No need for complete accuracy; could detract</td>
<td>Accuracy matters; improves feedback</td>
</tr>
<tr>
<td>3 Attitude towards tools</td>
<td>Over-reliance on tools is an issue</td>
<td>Tools help us do our job better</td>
</tr>
<tr>
<td>4 Focus of teaching</td>
<td>There are other things to be providing feedback on</td>
<td>Correct specification (also) matters</td>
</tr>
<tr>
<td>5 Need for a tool</td>
<td>The task is too simple to warrant use of a tool</td>
<td>Even with simple tasks, we make mistakes that tools can find</td>
</tr>
</tbody>
</table>
Tutors Comparing utility to students and tutors (percentages)

Percentages of 20 tutors reporting a tool would be Extremely, Very or Moderately useful to themselves and to students

<table>
<thead>
<tr>
<th>Tool use to tutors when marking</th>
<th>Extremely</th>
<th>V+E</th>
<th>M+V+E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit tests</td>
<td>26</td>
<td>58</td>
<td>79</td>
</tr>
<tr>
<td>Style checking</td>
<td>22</td>
<td>39</td>
<td>72</td>
</tr>
<tr>
<td>Compilation error help</td>
<td>20</td>
<td>40</td>
<td>70</td>
</tr>
<tr>
<td><strong>Specification checking help</strong></td>
<td><strong>17</strong></td>
<td><strong>33</strong></td>
<td><strong>72</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool use to students</th>
<th>Extremely</th>
<th>V+E</th>
<th>M+V+E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit tests</td>
<td>27</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td><strong>Specification checking help</strong></td>
<td><strong>21</strong></td>
<td><strong>47</strong></td>
<td><strong>69</strong></td>
</tr>
<tr>
<td>Style checking</td>
<td>20</td>
<td>45</td>
<td>65</td>
</tr>
<tr>
<td>Compilation error help</td>
<td>16</td>
<td>37</td>
<td>53</td>
</tr>
</tbody>
</table>
Cross-tabulation of scores awarded to utility of Specification Checking tool for use by tutors versus use by students showed:

- Association measure Somers’ d = 0.625, Statistically significant (p < 0.001).

*Tutors tended to rate Specification Checking highly for both themselves and students, or for neither.*

- Correlation measure Spearman’s rho between ratings of different tools was positive and statistically significant, except for Compilation Help.
- Most significant correlation was between specification checking and unit testing (0.801) at the 0.01 significance level.
- There are no significant correlations between ratings of other resources (e.g. module text) and tools.

*Tutors tended to rate Tools highly for both themselves and students, or for neither.*
Conclusions and future work

Where to from here?
What did the tool spot?

- Capitalisation errors, e.g. `Boolean` versus `boolean` (wrapper types)
- Inappropriate use of `static`
- Misspelled method names and variable names
- Incorrect access modifiers (e.g. `public` instead of `private`)

So what?

- Quite possibly the student’s code ‘works’ in some sense but
  - It doesn’t meet the specification
  - It may fail unit tests that rely on correct specification
  - The specification check can pick up issues that unit tests miss
    - in CodeRunner picked up failure to override `equals()`
    - Unit testing could have picked this up, but our unit tests missed it.

- In question authoring, the process of producing a specification helped identify several issues with English specifications of questions
What next?

- I intend to rewrite the code from scratch using a different approach
  - Will help fix a couple of technical issues
- Redeploy in Coderunner, and make more extensive use of it there
- Add style checking for tutors, and for students in Coderunner
- Create a standalone version of the tool in anticipation of M250 rewrite
- Use by tutors is optional.

- Write more of this up…
THANK YOU