### Project 2: Program verification with ESC/Java2

Ques 1. In the end, do you think that you found ***all*** problems and that the code is correct?

🡪 I gone through various slides, pdf & I tried really hard to solve all the problems but still I’m not sure that all the problems are solved by my side or not.

I changed some operators in the code to make it more simple and error free.

Overall the project was very interesting.

Ques 2. Can you think of ways in which the tool or the specification language could be improved?

🡪 One way in which JML could be improved is the addition of temporal logic specification operator based on specification pattern as in BSL (Bandera Specification Language). There are synchronization pattern implementations, Such as those presented in this work are sufficient, and that might require JML to be extended with temporal logic annotations to be properly specified.

There are so many different tools for code improvement for different languages such as Java ,C++ etc.

|  |  |
| --- | --- |
| Tool | Notes |
| Check Style | Besides some static code analysis, it can be used to show violations of a configured coding standard. |
| Find Bugs | Based on Jakarta BCEL from the University of Maryland. |
| IntelliJ IDEA | Cross-platform IDE with own set of several hundred code inspections available for analyzing code on-the-fly in the editor and bulk analysis of the whole project. |
| JArchitect | Simplifies managing a complex code base by analyzing and visualizing code dependencies, defining design rules, doing impact analysis, and by comparing different versions of the code |
| Squale | A platform to manage software quality |
| ThreadSafe | A static analysis tool focused on finding concurrency bugs. |
| PMD | A static ruleset based source code analyzer that identifies potential problems. |

The above tools are specially design for analysing, Visualizing and error detecting in code for JAVA.

Que 3. Instead of the tool we used, can you think of other ways (formal or informal, tool-supported or not) to find the problems that the tool found? If so, would these alternatives

* find fewer problems, the same, or more?
* find problems sooner or later than the current approach?
* require more work or less work?
* provide you with more confidence or less confidence that the code is correct?

🡪 We can use theorem provers that assist in finding a proof manually. **Formal methods** are a particular kind of [mathematically](https://en.wikipedia.org/wiki/Mathematically) based techniques for the specification, development and [verification](https://en.wikipedia.org/wiki/Formal_verification) of [software](https://en.wikipedia.org/wiki/Software) and [hardware](https://en.wikipedia.org/wiki/Computer_hardware) systems. But these techniques are very **expensive**

• If we use Mathematically based technique then maybe we can find more problems

• Mathematically based technique is very time consuming

• It requires very large amount of work

• We cannot say that which technique provide more confident or less confident that the code is correct because Mathematically based technique is manual based approach.

Manually people can make mistakes even software can make mistakes.

Therefore, I found this question is irrelevant.