Compiler Construction 2017

Project 3 - Java byte code generation and Optimization

- Implement arbitrary optimization techniques on your intermediate representation of Mini Java. The more, the better.
- Implement a code generation unit that transforms your intermediate representation to java byte-code. For the generation of Java byte code use the bcel library:
  (`http://commons.apache.org/proper/commons-bcel/`).
  Hint: Use the `javap` tool with argument `-c` (`javap -c Foo`).
- The provided test suite specifies the byte code generation rules and is mostly for you to get a feel for the number of features you have implemented. It will be used by us to do an initial evaluation of your work, but we reserve the right to use additional test cases and other methods for grading. You are not allowed to modify the test cases in any way.
- The provided test suite should be imported into your eclipse project.
- 4 weeks to complete this part of the project (by 26.5.2017 10:00h).
- Push your solution (the Eclipse project, including the parts provided by us) into the your bitbucket repository, shared with the user *scg-uko*.
- You are not allowed to use pre-made files from the internet, all parts of the project solution should be the authors own work.

Best Bytecode Generator Contest

Your optimization techniques will be evaluated from the bytecode size point of view. The best team (teams?) will be awarded with a small reward.

NB: Make sure you properly use method `addMethod(ClassGen cg, MethodGen mg)` *that counts the number of bytecodes.*