Software Engineering

Assignment 3

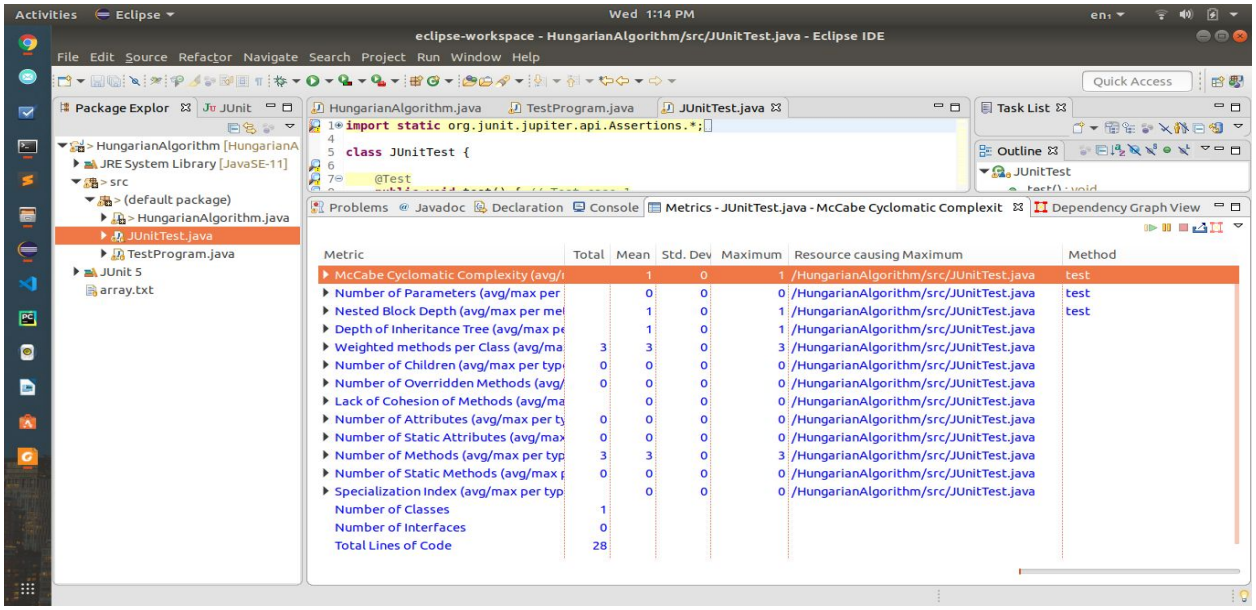
Part B

Kunal Kumar

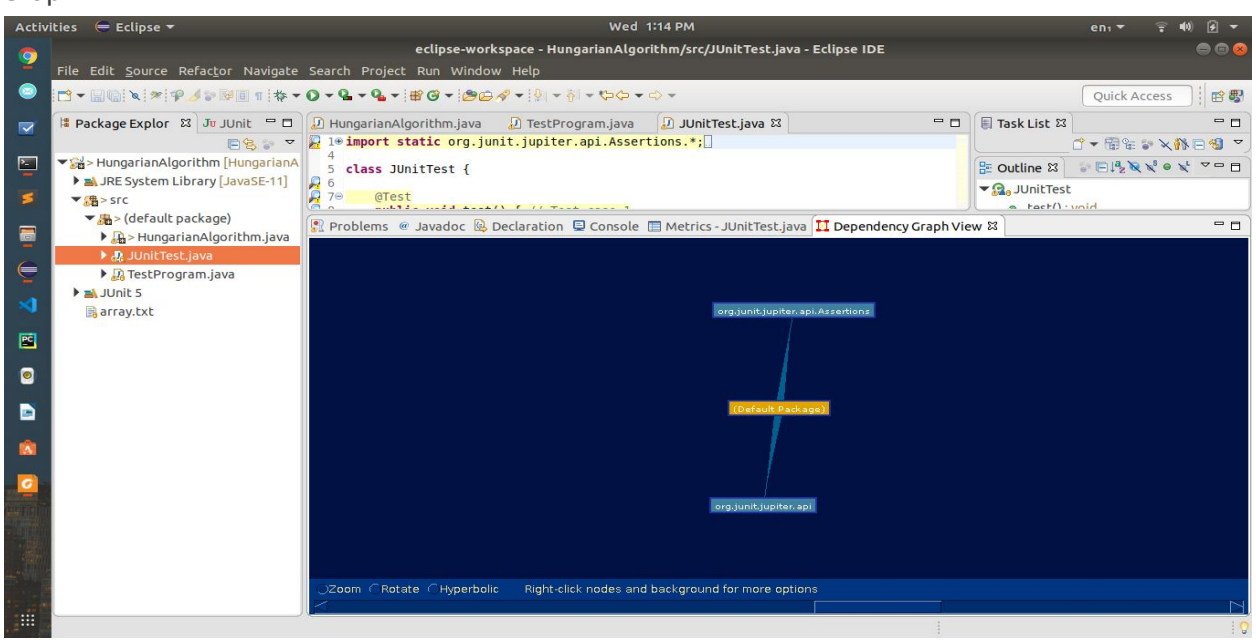
170010012

Metrics

1. Applied metrics plugin to code
2. Output of Metrics



Graph:



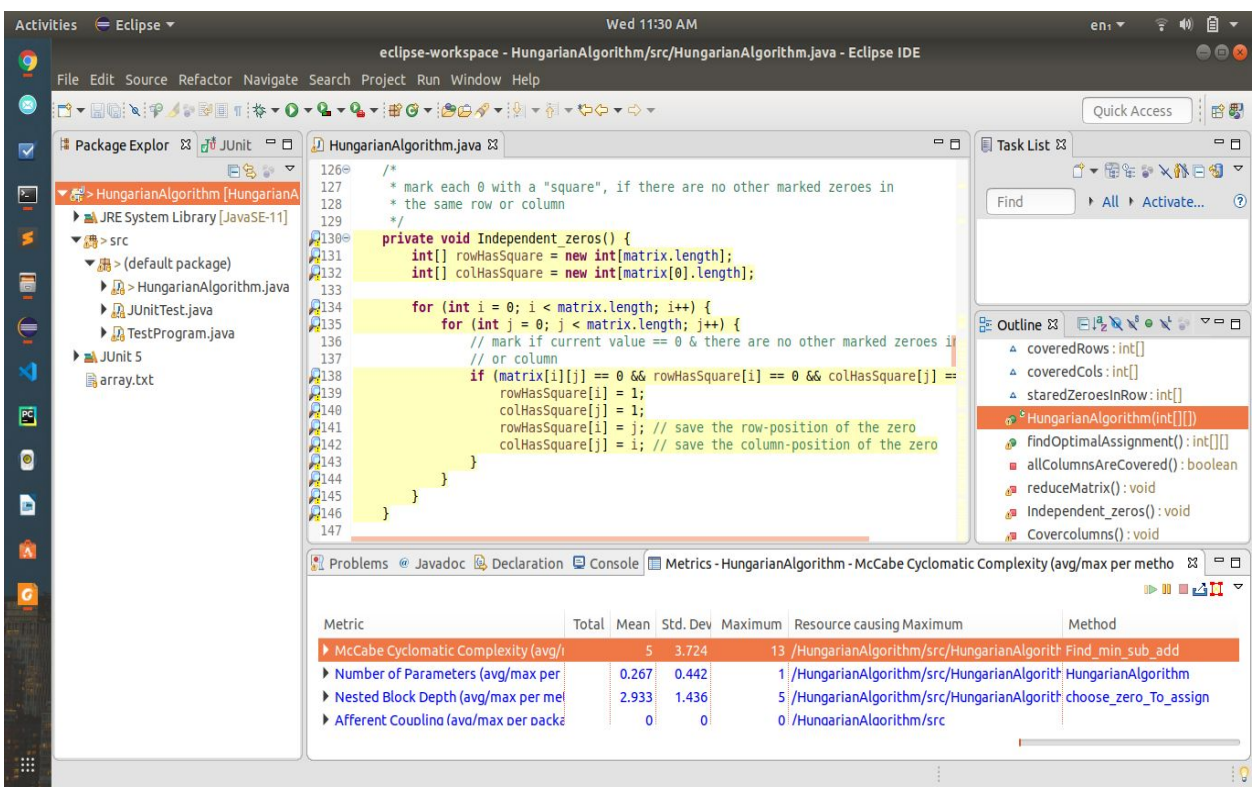
3. **Brief Description of Metrics Report:**

   The metrics view indicates the progress of the metrics calculations as they are being performed in the background. It basically shows statistics of each part of the code. This plugin calculates various metrics for our code during build cycles and warns us, via the Problems view, of 'range violations' for each metric. This allows us to stay continuously aware of the health of your code base.

   The view is *started from the metrics view menu or toolbar and it only works when a source folder or entire project is selected.* It shows a dynamic hyperbolic graph of the package dependencies, which can be zoomed and rotated. Use the radio buttons and scrollbar to manipulate the graph. The red and blue rectangles represent packages, and the edges their dependencies.

Checkstyle

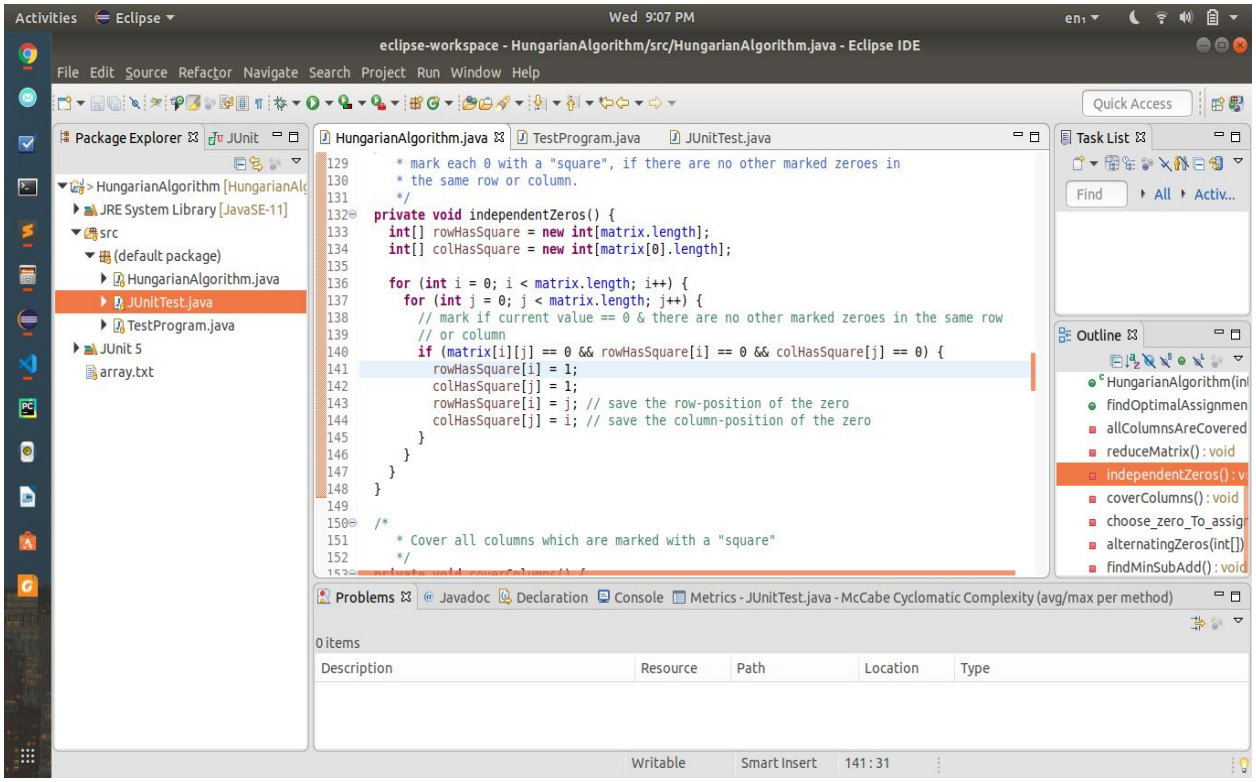
1. Applied Checkstyle plugin to code
2. output of checkstyle plugin



3. **Brief Description of CodeStyle report:**

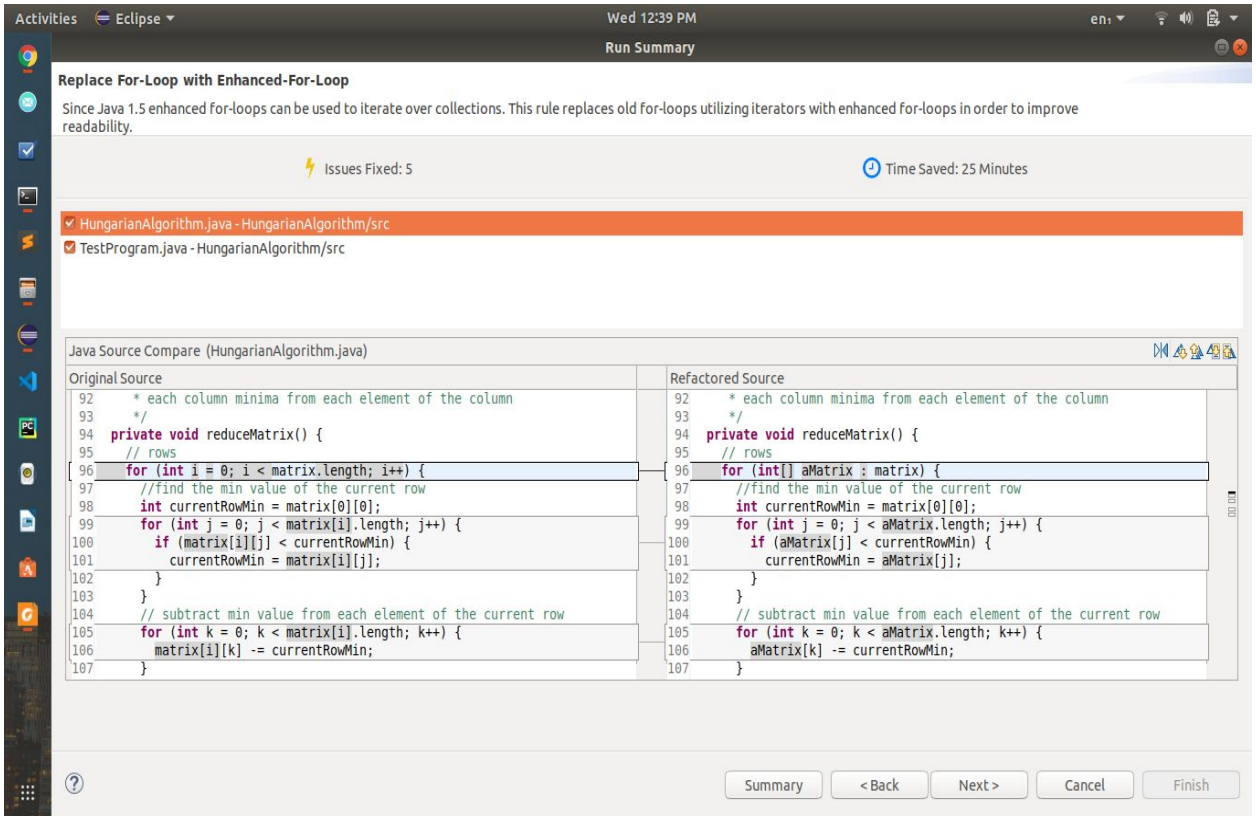
The Eclipse Checkstyle Plugin integrates the static source code analyzer CheckStyle into the Eclipse IDE. With the Checkstyle Eclipse Plugin our code is constantly inspected for coding standard deviations. Within the Eclipse workbench you are immediately notified of problems via the Eclipse Problems View and source code annotations similar to compiler errors or warnings. If we work in a development team consisting of more than one person, then obviously a common ground for coding standards (formatting rules, line lengths etc.) must be agreed upon - even if it is just for practical reasons to avoid superficial, format related merge conflicts. Checkstyle (and the Eclipse Checkstyle Plugin for that matter) helps us define and easily apply those common rules. This ensures an extremely short feedback loop right at the developers fingertips.

4. Modified code



Refracting (jSparrow)

1. Applied refractor plugin to code
2. output of refractor plugin



3. **Brief Description on refractor**

   jSparrow detects and automatically replaces bugs and code smells in Java sources with a rule-based approach. jSparrow finds bugs and other code smells and replaces them with clean and modern Java code. This keeps your system healthy and strong. jSparrow automates refactoring of Java code thus it makes repetitive work unnecessary when upgrading to a new Java version. jSparrow improves our Java source code. Removal of potential bugs and code smells is one of the main purposes of jSparrow. jSparrow is very efficient in disposing of these kinds of threats with its rules. You can group the profiles into individual rule profiles, which also can be exported and imported to share them within your development team. After the run of jSparrow, it shows us how the quality of our sources can be improved. jSparrow delivers a solution for each found old language construct or threat and shows exactly how to solve it. We can run the rules

through our whole sources and review issues and their solutions according to best practices.

4. Modified code