|  |
| --- |
|  |
| CS4218 Milestone 2 |
|  |

|  |
| --- |
|  |

**Team 05**

KOH CHENG GEE A0126199W

MUHAMMAD RAZALI A0133267H

PHOON QIANONN A0126232U

TANG WEI REN A0125531R

# Details about TDD process

* Write test cases for all applications in Basic Functionality (BF), Extended Functionality 1 and Extended (EF1) Functionality 2 (EF2)
* Add more test cases after debugging and the complexity of test cases is keep increasing
* Simple Test Cases
  + Test for basic errors
    - Display error message when there is no arguments for application which requires argument
      * echo, head, tail, sort
    - Display error message when there is argument for application which does not requires argument
      * pwd, date
  + Test for basic feature:
    - Pass test cases to pass basic features of each application
    - Only one argument is given
      * E.g. *sort text.txt*
* Advance Test Case
  + Test cases now contains multiple argument
    - E.g. *sort text.txt new.txt –n*
  + Test case now contains a combination of command
    - E.g. "cat Tests\pipeFiles\pipe\*|grep \"Hello World\

# Unit Testing

Unit Test for all functionalities

* **Basic Functionality (BF)**:
  + Shell: calling applications, quoting, semicolon operator.
  + Applications: cat, cd, pwd, echo, head, tail.
* **Extended Functionality 1 (EF1)**:
  + Shell: globbing, IO-redirection.
  + Applications: cal, grep, sort.
* **Extended Functionality 2 (EF2)**:
  + Shell: pipe operator, command substitution.
  + Applications: date, sed, wc.

Conducting of Unit Test for each shell and application for all functionalities

# Integration Testing

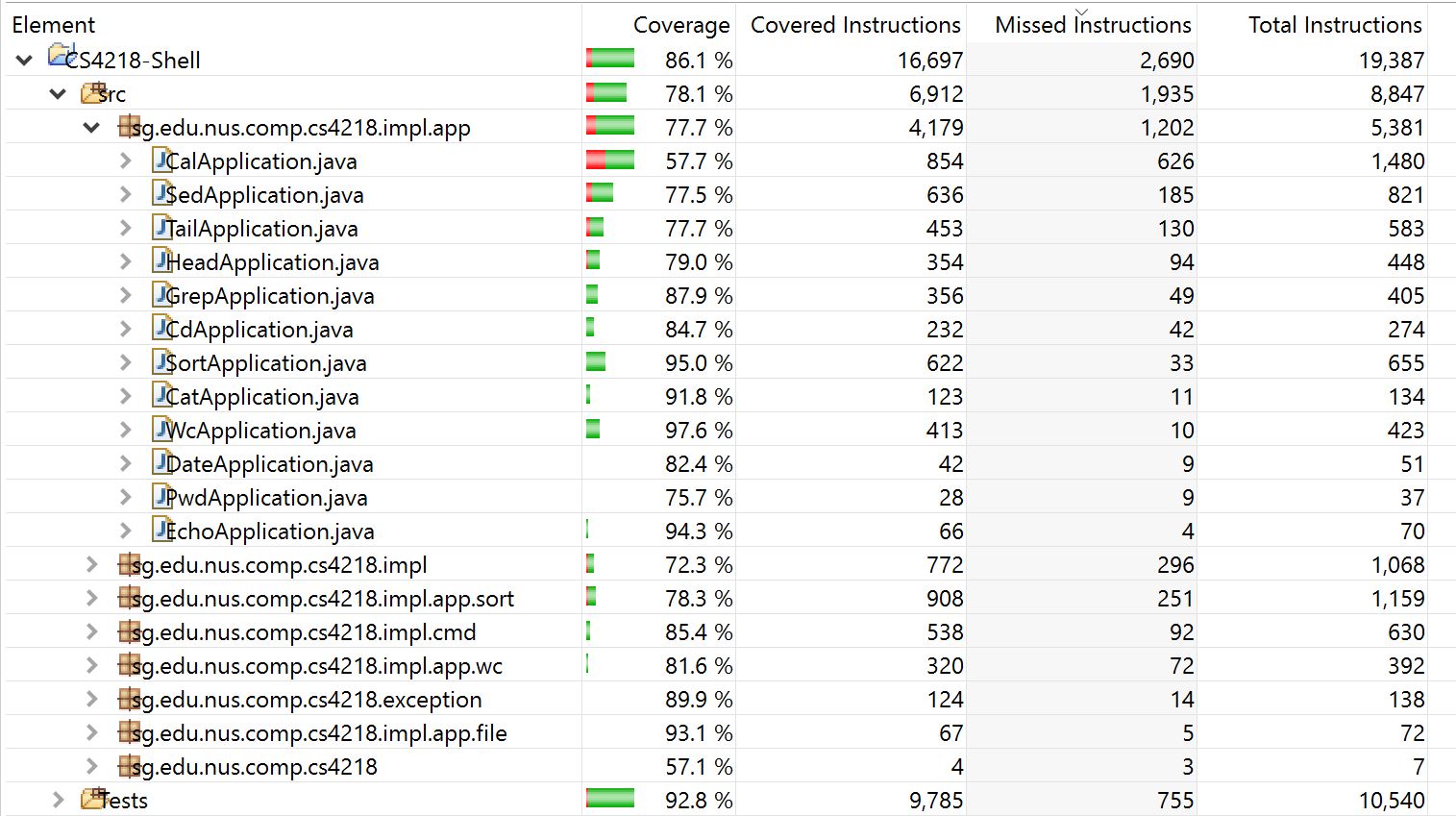
Integration Test for all functionalities

* **Basic Functionality (BF)**:
  + Shell: calling applications, quoting, semicolon operator.
  + Applications: cat, cd, pwd, echo, head, tail.
* **Extended Functionality 1 (EF1)**:
  + Shell: globbing, IO-redirection.
  + Applications: cal, grep, sort.
* **Extended Functionality 2 (EF2)**:
  + Shell: pipe operator, command substitution.
  + Applications: date, sed, wc.

Integration testing is done by executing command from “*ShellImp*l”:

* shellImpl.parseAndEvaluate(cmdline, stdout)
* *parseAndEvaluate(String cmdline, OutputStream stdout*) is use to execute all application
  + *cmdline* contains the command e.g. “*wc –lwm test.txt*”
  + *stdout* is the outputtream which will receive the results 1 10 20 tes.txt

## Code Coverage



EclEmma was used to give an overview of the code coverage of both unit and integration test

* Improves were made to the unit and integration test to try to cover as much code as possible
* However, it may not possible to give a complete code coverage as it was hard to simulate some errors such as *BufferReader* error.