# Project Plan - Team 20

Start date: 03 August  
Due date: 23 August

## **Iteration Objectives**

### Iteration 1 (03/8 – 12/8) ?\* multiple edits to iterations none to groups ?\*

Objective 1: Complete refactoring ?\* style guidelines group 1 and 2 to an acceptable level

Outcome 1: Code branches ready for review (meets code review entry criteria)

Objective 2: Complete code reviews and merges of refactored code base.

Outcome 2: All updated code merged back to master, master still works, ready to rebranch.

Objective 3: perform any non functional code refactoring as is convenient.

Outcome 3: reduce time to complete iteration 3.

### Iteration 2 (13-8 – 18/8)

Objective 1: Complete refactoring style guidelines group 3 and 4 to an acceptable level

Outcome 1: Code branches ready for review (meets code review entry criteria)

Objective 2: Complete code reviews and merges of refactored code base.

Outcome 2: All updated code merged back to master, master still works, ready to rebranch.

Objective 3: perform any non functional code refactoring as is convenient.

Outcome 3: reduce time to complete iteration 3.

### Iteration 3 (13-8 – 18/8)

Objective 1: Complete refactoring ?\* style guidelines group 5 and 6 to an acceptable level

Outcome 1: Code branches ready for review (meets code review entry criteria)

Objective 2: Complete code reviews and merges of refactored code base.

Outcome 2: All updated code merged back to master, master still works, ready to rebranch.

Objective 3: perform final review and testing of code base to be ready for submission for assessment.

Outcome 3: have assessable items ready for submission on time and to an acceptable standard

## **Task Assignment for Assignment 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team Member  Iteration | Semester Week | Hayley | Kenneth | Inconcert |
| Iteration 1  (03/08 - 12/08) | Week 3,4 | Refactor Group 2.  Review merged code base for conflicts and errors. | Refactor Group 1.  Review merged code base for conflicts and errors. | Review team members code refactoring.  Confirm team member code build.  Merge all refactored code branches.  Update morged code base to new version. |
| Iteration 2  (13/08 – 18/08) | Week 5 | Refactor Group 4.  Review merged code base for conflicts and errors. | Refactor Group 3.  Review merged code base for conflicts and errors. | Review team members code refactoring.  Confirm team member code build.  Merge all refactored code branches.  Update morged code base to new version. |
| intentionally blank spacer | | | | |
| Iteration 3  (19/08 – 23/08) | Week 6 | Refactor Group5.  Review merged code base for conflicts and errors. | Refactor Group 6.  Review merged code base for conflicts and errors. | Review team members code refactoring.  Confirm team member code build.  Merge all refactored code branches.  Update morged code base to new version.  Perform final code review and version commit. |

Appendix A

Group 1

1. All variable names are to be meaningful. Variable names should be nouns or noun phrases.

All variable names are to start with a lowercase letter and to be in camelBack.

4. The names of variables used as constants (static final variables) should be meaningful, all upper case, and underscore separated.

Group 2

2. All method names are to be meaningful. Method names should be verbs or verb phrases. E.g. isAvailable() rather than available(). All method names are to start with a lowercase letter and to be in camelBack.

3. All class (and enum) names are to be meaningful. Class (and enum) names should be nouns or noun phrases. All class (and enum) names are to start with an uppercase letter and to be in CamelBack.

Group 3

6. Anywhere curly brackets MAY be used, they MUST be used.

7. There MUST be a SINGLE consistent bracketing style used throughout the code:

Group 4

9. The argument list of any method calls MUST NOT contain any other method calls

10. Argument lists to method calls should start immediately after the opening bracket, be separated by whitespace, and the closing bracket should follow the last argument without whitespace.

Group 5  
11. Singletons MUST return their sole instance through a method called getInstance()

8. Indentation increments MUST be 4 spaces for every level of nesting. Not tabs, not 8 spaces, not 2 spaces; 4 spaces.

Group 6  
11. Singletons MUST return their sole instance through a method called getInstance()

5. All variables and operators MUST be separated by white space.