Programming Practice and Applications (PPA)
Major Piece of Coursework (Group Project), CWX
(15%, 100 marks)
Mark scheme

1 Overview

This coursework assignment is worth 15% of your final module mark for PPA. Part of this grade will come from a solution submitted collectively, and part of this grade will come from an individual report, submitted individually. Table 1 shows how the marks for this assignment will be awarded.

2 Mark Ranges

Each requirement is marked in the range of A – F, where A is full marks and an F is a fail. The meaning of these marks, in the context of different requirements, is listed in this section.

2.1 Marking Scheme for Requirements 1 - 5

(A) The requirement is completely implemented, there are no semantic errors or inefficiencies (logical bugs) that can be identified.

(B) The requirement is implemented, but some minor functionality may not have been completed. The parts that have been implemented, have no identifiable logical bugs. Alternatively, the requirement has been implemented completely, but with some minor logical bugs.

Requirement Number	Requirement Description	Marks Available
0	**Setting up Github**	**5**
1	**Main Window**	**5**
2	**Panel 1: Welcome**	**5**
3	**Panel 2: Map**	**20**
3.1	Interactive map	10
3.2	List of sightings	5
3.3	Sorting sightings	5
4	**Panel 3: Statistics**	**25**
4.1	Panel behaviour	6
4.2	Base statistics	8
4.3	Additional statistics	8
4.4	Saving preferences	3
5	**Panel 4: Surprise Me**	**5**
6	**Individual Project Report**	**15**
7	**Code Quality**	**20**
7.1	Documentation	10
7.2	Structure of code (including MVC)	8
7.3	Minimising API calls	2

Table 1: Mark scheme overview

(C) The requirement is implemented, but with some minor logical bugs and some missing minor functionality.

(D) While major parts of the requirement are missing, some minor functionality has been implemented. What has been implemented does not show any logical bugs.

(E) Major parts of the requirement are missing and there are some minor logical bugs.

(F) Major parts of the requirement are missing and there are some major logical bugs. Alternatively, nothing has been implemented at all.

2.2 Marking Scheme for Requirement 7.1

The use of javadoc

(A) All public classes, interfaces, methods, and attributes are provided with concise and

useful documentation comments. These comments make judicious use of additional tags such as @param, @return, @see, or {@link}.

(B) All public classes, methods, and attributes are provided with concise and useful documentation comments.

(C) While not all public elements are documented, those which are documented well. The elements which are not documented are of less importance to the design of the application than the ones which have been.

(D) Some elements are documented, some of these document comments may even use additional tags. The majority of elements, including key elements, is undocumented.

(E) Only very few elements are documented. In some places, these comments may be normal comments rather than javadoc comments.

(F) There are no javadoc comments.

Marks for inline comments will also be implicitly awarded.

2.3 Marking Scheme for Requirement 7.2

The structure of your code

(A) The code is well structured into a set of classes (including anonymous and inner classes, where appropriate) and methods. The code also avoids repetition and lengthy conditionals. At the same time, the code is not over-fragmented into overly small classes: for each class a clear responsibility can still be identified.

(B) The code is well structured into a set of classes and methods. There may be some code repetition or lengthy conditionals OR some over-fragmentation.

(C) The code is structured into a set of classes and methods. There may be some code repetition or lengthy conditionals AND some over-fragmentation.

(D) While all code may be contained in a single file, using separate classes only where mandated by the Java API, code repetition is completely avoided by using separate methods with appropriate parameters.

(E) While all code is contained in a single file, using separate classes only where mandated by the Java API, code repetition is partially avoided by using separate methods with appropriate parameters.

(F) All code is included in the main method of the main class except where this is impossible because of requirements of the Java API.

2.4 Marking Scheme for Requirement 6

The individual report

As indicated in Table 1, 85% of the major coursework mark is attributed to the solution submitted collectively by your group, for which each member will receive the same grade. 15%, however, is attributed to each group member individually, based upon a written report, for which each group member will receive a different grade. The report will also be marked using an A – F scale:

(A) The report is clearly structured into sections, which flow well as a narrative (i.e. introduction through to sets of analyses through to summary and conclusions). A domain analysis, hierarchical task analysis, virtual windows and a global navigation structure are all included. Each diagram is accompanied by a suitable section of text detailing what has been learnt during the construction of the diagram.

(B) The report is structured into sections, but the narrative they present isn't particularly intuitive. One of the analyses or one of the diagrams is missing. Each diagram is accompanied by a suitable section of text detailing why this analysis or design approach was taken.

(C) The report is organised into sections, but they are not in the correct order and therefore do not present a suitable narrative. Two of the analyses or diagrams are missing. Each diagram is accompanied by a suitable section of text detailing how the analysis or design approach was taken.

(D) The report lacks any clear sections and therefore has no narrative. Three of the analyses or diagrams are missing. Each diagram is accompanied by a suitable section of text that elaborates on the diagram shown.

(E) A report is submitted, with some vaguely relevant content, but all of the analysis or diagrams are missing.

(F) No report is submitted.

3 Exceptional Cases

1. All group members will receive a mark of zero if the API acknowledgement string is not shown.
2. All group members will receive a mark of zero if the Ripley JAR file is not used to access the Ripley API, and instead the API is accessed via other means.
3. All group members will receive a mark of zero if the required API data is acquired from a different source.
4. All group members will receive a 10% deduction to their grade if their submitted code does not run, and minor debugging is required to run the code without exception[1].
5. All group members will receive a 60% deduction to their grade (i.e. the mark will be capped at 40%) if significant debugging would be required to run their submitted code without exception.

3.1 Mark Moderation

1. A student will face a mark reduction of up to 60% if sufficient evidence is obtained from *all* group members, via `TeamFeedback`, that they did not contribute sufficiently to the submitted solution.
2. A student will receive a mark of zero if sufficient evidence is obtained from *all* group members, via `TeamFeedback`, that they did not contribute at all to the submitted solution.
3. A student will face a mark reduction of up to 60% if they do not push a comparable number of commits to the rest of their team members to their remote repository (hosted on `Github`). All team members are expected to have a resonable number of commits (> 10).
4. A student will receive a mark of zero if they do not push anything to their remote repository (hosted on `Github`).

[1] The definition of minor is as the discretion of the examiners.