University of Stirling
Computing Science and Mathematics

CSCU9P6 — Software Engineering II
Group project Spring 2019

For the purpose of this project each of you is required to operate as a member of a team. The task of the team is to implement a working system in Java satisfying the object oriented design described later in this document on page 7.

Lists of team members and monitoring officers, this description and project materials are on Canvas and the Courses file server `K:\CSCU9P6`

The *organization* of the work is largely up to each team. However, for each team we will appoint a *project monitoring officer*, and the team must meet with this officer at one project meeting each of two separate weeks. The monitoring officer will attend for up to half an hour (the group meeting might well continue for longer), and we appreciate that once the implementation gets properly under way, shorter meetings may be appropriate with the agreement of the monitoring officer. Attendance at these meetings is **compulsory**, and non-attendance may result in No Mark being awarded for the course. You should give your monitoring officer a *brief, informal*, progress report at the start of each meeting. Their normal role will be to observe silently, in the background — they are not there to solve design problems for you. If you wish to ask your monitoring officer for clarification of the problem, then they may need to consult "higher authorities" for the expansion of details in the requirements — a response will be provided as soon as possible, and, if appropriate, will be disseminated publicly for the attention of all teams.

Sommerville's and Pressman's texts on software engineering provide guidance on organizing teamwork. We recommend that each team appoints one of its members as *project manager*, but this is up to you, and that you plan your work schedule using a *Gantt chart*. We also recommend that, wherever possible, you think about and define very clearly the interfaces of each Java class (the public operations offered) and work as individuals on separate classes (or groups of classes).

Each group has been allocated a communal, shared folder on a Computing Science file server. If your group number is *nn* (*01, 02, etc*) then your shared folder is in `\\wsv\Groups\P6\`*nn*. Each member of the group has full access to all files created there; you might like to create a subfolder per group member as a place to keep work but which is also accessible to other members of the group if necessary (e.g. in case of illness).

Each group has also been allocated a Subversion repository on our server: `svn://svn.cs.stir.ac.uk/p6-`*nn* where we recommend that you manage your central versions of your project files. Each member of the group can checkout a copy of the project into their own file space, or into personal sub-directories of the shared directory, regularly committing any changes that they make, and synchronizing to fetch updates committed by other members of the group.

Note: The best way for Eclipse/Together to access a project in the shared folder, is to *Map a Network Drive letter* to the shared folder .

Remember: You should never have the same project folder open with Together or Eclipse on more than one computer simultaneously, and there should be no need for this! [But several copies separately checked out from Subversion by different programmers is fine!]

I recommend that you use the Together project ***only*** *for* ***reference*** *to the documentation,* *and that for development you use the Java already extracted from the Together project, and work in Eclipse, with the the most up to date project files held in the Subversion repository. Please note that the versions of Together and Eclipse in the labs **are not compatible with each other**, and you should **not** open one's projects with the other!*

While you are working, it will be important to synchronize your work with the repository on a regular basis, and **in particular at the start and end of each session of work**.

The Division has two rooms that you can use for team meetings: 4B96 and 4X6. They can be booked via Linda, Grace, Gemma or Kimberley in the Divisional Office (4B112) from where keys can also be collected. The rooms are to be left tidy and may be claimed as a higher priority for Divisional uses (4B96 is in heavy use for many meetings). The Library also has a collection of bookable meeting rooms — see `http://stir.ac.uk/1wz` Further, you can now book free teaching rooms through the ResourceBooker at `https://resourcebooker.stir.ac.uk/` : log in with your Portal username and password, choose Book a Teaching Room, and then look at Cottrell Central.

Assessment arrangements

Each group will make a **joint software development submission**, and a **single agreed statement of individual contributions**. Each student will submit an **individual essay**.

Group software development submission

Each group will produce the following assessable output:

1. The final Java files, **emailed** as one Zip file to *sbj@cs.stir.ac.uk* named `Group-`*nn*`-Java.zip`
2. **You are not required** to make any substantial alterations to the design/architecture of the system. **However, if you do,** then you should submit a brief but clear description of the alterations and the rationale for them, with relevant parts of the new class diagram.
3. **Hard-copy documentation** of the *JUnit testing* that you would carry out to demonstrate the correctness of an implementation of the `Permit_List` class — for simplicity assume that the classes that it depends on are themselves all correct and so may be used in the testing. The documentation must include the full code of the JUnit tests, and a description of the **rationale** for the test cases chosen (well written comments in the JUnit test code would be fine).

   *You may wish to carry out unit testing of other classes, but you should **not** hand in details of those tests.*
4. **A demonstration** by the group of the working system.

Each group should hand in to the labelled box outside 4B89, or directly to Dr Jones, one hard-copy of items 2 and 3 described above, on or before 17.00 on 29 March 2019. The cover of the document should clearly identify the *number* of the group, but should *not* give the names of the members.

The demonstrations will be arranged in due course.

Statement of individual contributions

When assessing group projects, care is taken to ensure that the marks awarded fairly reflect individual contribution and effort. The marks awarded for this assignment will take into account the individual essays, the individual contributions and effort as well as the group results. The individual contributions will be used to give each student an adjusted mark based on the group software development submission mark.

The group must submit **one printed copy** of the table provided attached to this document (page 5), completed to show the relative contribution of each team member: Please distribute 100 points between the group members according to what the group *agrees* their contribution to have been. Please double check that the sum of the row is 100.

Note that each member of the group must sign the form to indicate that it has been discussed and agreed.

The group must attach to the form a brief statement listing the contributions of each member. *For this purpose the group will need to keep a record of who does what.*

The statement of individual contributions should be posted into the assignment box by the same deadline as the group submission,17.00 on 29 March 2019.

Individual essay

In addition, each individual group member must submit a short essay (maximum two sides of A4) discussing the *management of the project*:

- Describing how their group *organized* its collaborative work, with reference to the project management topics studied in CSCU9P5,
- assessing how effectively the group worked towards achieving its goals,
- and discussing, with hindsight, whether *and how* the collaborative work organization could have been different to give a better outcome or, if you think that no improvement was possible, then why the particular organization was so effective.

Note: This short reflective essay will be assessed on the *quality of the analysis, discussion, and writing, rather than than on whether an "approved" and effective organization was adopted*.

The individual essay must be submitted via Turnitin on Canvas by the same deadline as the group submission, 17.00 on 29 March 2019. The essay should have a title page giving the module code, the title of the assignment "Group project individual essay", your group number and student number, but not your name. Pay attention to the layout and clarity of your work.

Independent work as regards the individual essay

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on plagiarism which can be found at `http://stir.ac.uk/1x0`

Late submission

The standard University procedures dealing with non-submission and with late submission apply. Assessed coursework submitted late will be accepted up to seven days after the submission date (or expiry of any agreed extension) but the mark will be lowered by three marks per day or part thereof. After seven days the piece of work will be deemed a non-submission, and will result in the award of No Mark. This rule may be relaxed for students who can show good cause for failure to submit. Good cause may include illness (for which a medical certificate or other evidence will be required).

Assignment assessment

Although this is a group project, you will be given an individual mark for this assignment based upon the following 3 factors:

1. The Group Submission
2. The Individual Contributions Form
3. The Individual Essay

This assignment is worth 35% of the overall mark for CSCU9P6.

The Group Submission mark adjusted according to the Individual Contributions will account for 80% of your mark.

The Individual Essay provides the remaining 20% of your mark.

Group number:

CSCU9P6 Group project Spring 2019 Statement of Individual Contributions

Please quantify the contribution that each member of your team has given to the group work. Write the names of the members of the group in the first row below. Then, distribute 100 points between them, according to what the group agrees the contribution of each member was to the development of the project.

For instance, in a four person group, if all the members have contributed equally, then each member will receive 25.

Ensure that

1. the group number is include in the box above,
2. the names and contributions of your group members are given in the table below,
3. each member signs the form where indicated,
4. a brief state of the contributions of each member is attached.

	Contribution				
	Member 1	Member 2	Member 3	Member 4	Member 5
Name					
Distribute 100 points					

Signatures

1.

2.

3.

4.

5.

(This page deliberately left blank)

PACSUS project specification

In this assignment you will be implementing an information and control system for the **P**arking **A**ccess **C**ontrol **S**ystem at the **U**niversity of **S**tirling (PACSUS).

The project is an implementation of the way that the entry of vehicles to the campus *used to be controlled (around year 2000)* by a system of permits and automatic barriers that read number plates and denied or permitted access as appropriate. The design is based on ideas contained in documents written by the University administration at the time, principally a memo that accompanied permit application forms in July 2002. A copy of this document is given on the group project Canvas page, and in the group project folder in `K:\CSCU9P6`. [The system to be implemented does not cover all the detailed aspects of parking control described in those documents, it is just based on it but quite close in principle.]

The system is to be considered as implemented on a central computer with the various interfaces distributed around the University, and connected via special purpose networking. As far as the PACSUS software itself is concerned, the various interface devices could as well be in the same room as the central computer. [Of course, this is just an exercise, so the interfaces will be "only a simulation" and will appear as *separate windows* on a single PC screen — but this is actually very close to a real implementation where the separate windows might actually appear on *separate PCs* (or maybe simpler devices). This is like the DualDatabase and MVC practical exercises.]

You are provided with a Together model at an intermediate stage in the design process: use cases, a class diagram, and some sequence diagrams. Several of the classes could have had simple state diagrams associated with them, but they haven't. Some of the associations have not had final concrete decisions made about their implementation. Many reasonably likely standard operations for getting, setting, adding, deleting, finding have not been included. **You may wish to refine the class diagram with further operations and attributes as a way of generating more of the Java outline code, but submission of the more detailed design is NOT required.** No further use case nor sequence diagrams are required. State diagrams are not required.

The Together project itself contains a large amount of documentation on design decisions already taken. You need to explore and read it carefully. It contains a full set of use cases, with descriptions, and user interface boundary classes that indicate what should be implemented on the screen. These should be examined in conjunction with the University memo referred to above. The use cases are hyperlinked to the relevant boundary classes within the Together project.

- The Together model can be found in `K:\CSCU9P6\GroupProject\PACSUS`. You should copy this folder to your group working folder, remove the Read-only attributes, and then open it to investigate the design. For developing the software implementation, the outline Java files that Together has generated have been extracted into a separate folder. You can build a new Eclipse project using that folder and share it in your group repository.

  To make use of the Together model, you should launch Together, switch to the Modelling perspective and use **File** menu / **Import**. In the Import dialogue box choose "Existing projects into workspace" and then **Next**. Click **Browse** to *Select root directory*, and browse to locate your copied PACSUS folder. Then click Finish. This folder also contains the Java code generated by Together, plus a skeleton Main class in `Main.java`.

- More information/suggestions are given in the `README.txt` file — **START THERE, and READ IT!**

- Remember to survey the Descriptions in the Properties Pane when you click on use case diagrams, classes, methods, attributes, associations, messages... A lot of semantic information and design decisions are recorded there. In some cases there is no Description in the Properties, but there may be a Javadoc entry instead. A good way to view the Descriptions is to select the Description line in the Properties pane and then click on the ... that appears in the selected line — a separate window with the description will pop up.

- The design was constructed with the MVC architectural framework in mind. Read the design from that perspective.

  [**Caveat:** You do not need to split boundary classes into separate controller and view classes — this is the case when part of the "view" is used for the input of a "control" action, such as when the user can choose from a list of displayed items. However, the MVC principle can still be followed.]

- You will need to check on, and refine, the implementations that Together has implemented for the associations. New attributes and operations will certainly need to be designed. **However, no significant architectural changes to the design is required or expected.** Note: There are **no traps** where I have deliberately put in aspects of the design where I expect you to refactor the design.

- Various resources you might find useful:

  – On-line documentation of the Java class libraries at, say, URL `https://docs.oracle.com/javase/9/docs/api/index.html?overview-summary.html` (or Eclipse may look them up on the web for you).

  – The DualDatabase and MVC examples (from practicals) in `K:\CSCU9P6\Practicals`

  – The ListInterface example in `K:\CSCU9P6\GroupProject` shows how to use a JList widget – quite useful!

- The actual user interfaces **do not need to be beautiful, nor too clever** — just lists, buttons and text fields/areas, will be fine.

- **You should NOT use threads nor concurrency, nor are you required to store/retrieve the data managed by your software to/from files or a database.**

Good luck!

Simon Jones 19th February 2019