CSCC01 Introduction to Software Engineering - Midterm Test

27 October 2017

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO

In the meantime, read and complete this front page.

Last name: _____

First name: _____

UTOR ID: _____

Student number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Rules:

1. When you receive the signal to start, please make sure the copy of your examination is complete.
2. Legibly write your name and student number on this page.
3. There are a total of 100 marks and 6 questions.
4. Total time is 100 minutes.
5. Write and submit all of your answers as instructed for each question and using the computer in front of you.
6. The exam is closed book, and no aids of any kind are allowed.
7. At the end of the exam, return this examination copy to the exam staff.

Exam Setup

There is no question in this part. However, read carefully the informations below regarding the exam setup.

In this exam, you will be asked to complete a project documentation and source code. You will use the computer in front of you to do this work.

Similarly to what you have done so far in this term, you will use `git` and a private exam repository for submitting your work. Your answers will be graded **if and only if they have been pushed to your exam repository**. No file will be retrieved from the computer once the exam is completed.

Using the exam computer

Once you have logged-in with your UTORID and password, an explorer window should open after few seconds. This window shows the content of your working directory `G:/`.

Make sure to clone your exam repository and store all of your ongoing work on the `G:` drive. This drive is automatically backed up in case your computer crashes during the exam. However, as specified earlier, all of your answers must be pushed to the given `git` repository to be graded.

For the purpose of this exam, you need to use three programs only:

- **Git CMD** to execute git commands. Start Git CMD from the program shortcut located on `G:/`. Do not start it from the Windows menu.
- **Notepad** (default) or **Notepad++** to edit all documents related to the product backlog and sprint backlog given as text files.
- **Eclipse** to edit the product source code written in Java. Start Eclipse from the program shortcut located on `G:/`. Do not start it from the Windows menu. It might takes few seconds the first time you start Eclipse.

Using `git`

You will use `git` and a private exam repository (see below) to 1) retrieve the initial documentation and source code and 2) submit all of your answers. Throughout the exam, here are some good practices that you should adopt:

- **Push early, push often.** It is strongly advised not to wait for the end of the exam for pushing your answers to the repository. You should push each of your answers as soon as they are ready and/or updated.
- All of your commit must have meaningful commit messages.
- Do not push automatically generated files.
- Make sure to adopt good branching and merging management as seen during this term.
- All of your answers must be on the branch `master` unless specified otherwise.

Here is a brief reminder of the most important `git` commands. For the commands below, all keywords surrounded by double underscores such as `__email_address__` must be replaced with a proper input.

description	git command
setup a global variable email	`git config --global __email_address__`
clone a repository	`git clone __repository_url__`
pull changes	`git pull`
staged a file	`git add __filename__`
commit staged files	`git commit -m "__commit_message__"`
push latests commits to the repository	`git push`
create a new branch	`git branch __branch_name__`
change branch	`git checkout __branch_name__`
push/pull on a new branch (first time only)	`git push origin __branch_name__` `git pull origin __branch_name__`
merge a branch	`git merge __branch_name__`

Part I
Product Backlog

The City of Toronto wants to develop a new app that gives information about live traffic conditions on its highways. At first, when a user enters a highway landmark given as a triplet: the highway name (ex: 401), a direction (ex: east) and an exit number (ex: 387), the application should return the traffic conditions as either *moving very slow*, *moving slow* or *moving well*. Later on, we would also like the application to give traffic conditions of the closest highway landmark based on either the user's current location or a selected location on a map. Traffic information should be available to regular users but also to third-party application developpers so that they can integrate trafic information into their apps. However, unlike regular users, third-party application developpers must register first because we want to limit the number of requests made per month by each third-party application.

Question 1. [25 MARKS]

Complete the file `product-backlog/version-00/personas.txt` by briefly describing up to four (but not necessarily 4 if it is not adequate) Personas that would be useful for the design and development of your system.

Question 2. [25 MARKS]

Complete the file `product-backlog/version-00/user-stories.txt` by writing up to ten (but not necessarily 10 if it is not adequate) user stories based on the description of the required system. Choose the most relevant system features for your user stories and order them by build priority (the first one being the first feature to be built). Include user stories for all Personas that you have described in the previous question.

Part II
Sprint Backlog

Recall that in class we talked about a moderately efficient team of three developers: Alice, Bob, and Charlie. Their sprint length is 5 days and 1 story point corresponds to 1 developer-hour. This time each of them contributes equally to the project — 3 story points per day.

Suppose they have the following tasks in their product backklog:

task ID	story points	dependency
T1	1	
T2	3	
T3	5	T1
T4	11	
T5	1	T2, T3
T6	15	
T7	9	T5
T8	16	
T9	2	
T10	12	

Question 3. [20 MARKS]

Complete the file `sprint-backlog/sprint-01/sprint-plan.txt` by producing an adequate sprint plan. Allocate each task by filling the necessary cells with the right allocation value. For instance, if you want to have `Alice` work for `2` hours on task `1` on day `1`. Write `A:2` in the corresponding cell for task 1 on day 1.

When allocating your task, you should follow these constraints:

- the tasks with higher priority should be completed first whenever it is possible
- each task should be completed by one developer and one developer only
- for a developer to be able to work on a given task, each dependency must have been completed either 1) at last the day before if the dependency has been assigned to a different developer or 2) the same day if the dependency has been assigned to the same developer

Please, keep the grid well formatted. Use spaces but no tabs.

Question 4. [10 MARKS]

Complete the file `sprint-backlog/sprint-01/provisional-burndown-chart.txt` by producing an adequate provisional burndown chart. Instead of drawing an actual chart, complete the given grid by filling it with:

- the units for the x-axis and the y-axis
- all x and y values for that sprint including the value at origin ($x = 0$)

You are allowed to extend the array to the right with as many cells as you need. Please keep the grid well formatted. Use spaces but no tabs.

Part III
Feature Development and Release

In this exercise, you are going to work on the source code of the product.

Currently, there is one new feature called `current-location` that has been completed but not released yet.

Question 5. [10 MARKS]

First, create a new feature called `gps-coordinate` from the latest release. Then, implement this feature by adding a new function called `fromGPS` to the file `src/ca/toronto/traffic/api/Api.java`. This new function should have the following Java documentation:

```
/**
 * Returns the nearest highway landmark given a pair of GPS coordinates
 * For the purpose of the exam, this function returns a dummy value
 *
 * @param  Double latitude
 * @param  Double longitude
 * @return HighwayLandmark
 */
```

For the purpose of this exam, you are not required to return the real corresponding highway landmark. Instead, make this function return the same highway landmark (of type `HighwayLandmark`) regardless of the GPS coordinate given as arguments. This returned highway landmark should be:

- highway: `"401"`
- direction: `"east"`
- exit: `387`

Question 6. [10 MARKS]

After fully completing the previous question (and only after), release the two features in that following order:

- First, release the `current-location` feature (the one that you did not implement yourself)
- Then, release the `gps-coordinate` feature (the one that you did implement yourself in the previous question)

Use this page for rough work.

Use this page for rough work.

Total Marks = 100