SCSJ3623 Mobile Application Programming
Semester 2, 2019/2020

Exercise 2
Basic Navigations

Exercise Material
- Program template is provided for this exercise. Please download from the elearning and extract the ZIP file to your local drive.
- The tasks to be completed are stated in the Problem section below.

Pair Programming
- You will be adopting the "Pair Programming" strategy to do this exercise.
- Arrange yourself with you partner for the pair programming session.
- Members for each pair are pre-assigned.
- Each pair must ensure to switch role (between Navigator and Driver) periodically.

Duration
- You may need 90 to 120 minutes to complete this exercise.
- Should you need more time, you can grant it yourself provided you do not exceed the submission deadline

Plagiarism Warning
- You are allowed to refer to any resources.
- You are allowed to have discussions with others during the exercise session.
- However, you and your partner must do the exercise on your own.
- Any kind of plagiarism (e.g., copying and pasting code by any means) would lead your submission dismissed. This applies to both parties, students that copy others' code and students that give their code to others.

Late Submission Penalty
- **10% deduction for every 10 minutes late**.
- For example: if the duration of the exercise is 60 minutes, and your submission is received on the 61st minute, you are only eligible to earn 90% at most of the total marks.

Other Penalties
- Program that cannot run will not be graded at all, thus will get a zero mark.
- Program that does not follow proper naming convention will get 10% penalty.
- Program that is not properly indented will get 10% penalty.

Problem

In this exercise you will be implementing a basic navigation using the class Navigator. Develop a simple Todo application consisting of two screens, the first screen (Todo List) and the second screen (Task List).

Todo List Screen

This screen shows the list of list of todo items along with the following information:

- The number tasks that each todo item has.
- The progress of each todo item shown by the percentage of tasks completed for the todo item.

Task List Screen

This screen will show up when the user taps on a todo item in the first screen. The Task List screen shows the list of tasks for the selected todo has. The user can perform the following operations on the task items in this screen:

- Toggle the status a task from “in-progress” to “completed” or vice-versa by tapping on the task. A completed task is shown in Strike-through (or line-through), whereas an in-progress task in normal text.
- Remove a task item by long-pressing on the task.

Model Classes and Mock Data

Both screens must be built dynamically (do not use hard-coded to build the screens). In this regard, you need to define the following classes and create some mock data by using these classes:

- class Task. Each task has a title and the status indicating whether it has been completed or not. This class is fully given in the base program.
- class Todo: Each todo has a title and a list of tasks associated to it. Add a convenience method with a getter to calculate the percent of tasks completed. You may also need to define a copy constructor for this class.
- Mock data: create your own mock data for a list of todo items. Each todo item then needs to have a list of tasks.

Watch the attached video for more details.

Preparing the Exercise Material

1. Download the zip file (exercise2.zip) and extract it to a folder with the same name (exercise2).
2. You should get the following materials:
    - A base flutter project for the exercise
    - A video of the expected result (expected_result.mp4)
    - Question script (question.pdf)

Initializing Git

1. Go inside the exercise folder (using Git Bash)
2. Type the following command to initialize git

```
git init
```

3. Identify yourself to git so that everytime you perform git commit, it will also log your name. Note: use your short name and your partner's.

```
git config user.name  "you and yourpartner"
```

Note: you can only use the above command once. Running it again will overwrite the first one. Thus, combine your name and your partner's to be used to the command.

4. Add files to be tracked by git

```
git add .
```

Committing Changes

1. Type the following command to commit (or, to record) any changes you have made.

```
git commit -a -m "write a concise message for each commit"
```

2. You must have the following commits:
    - Define model classes and create some mock data.
    - Build the looks of both screens dynamically – with the mock data.

- Implement the onTap in the first screen to Navigate to the second screen.
- Implement the onPressed of the "Update" button in the second screen to back to the first screen.
- Implement the onTap in the second screen to toggle a task.
- Implement the onLongPress in the second screen to remove a task.
- Add a "Cancel" operation in the second screen.

Notes: You can only commit **maximum 10 commits** for this exercise.

Creating A Git Bundle File

1. Once you have done with your exercise, prepare a git bundle for submission.

```
git bundle create exercise2.git HEAD master
```

2. If you want to test out the git bundle file (exercise2.git) before you submit, clone the repo to another folder. First, copy the git file (using Window explorer) to the other folder. Then clone the repo (using Git bash)

```
git clone exercise2.git exercise2
```

Submissions

- Submit your work in **a git bundle** file (i.e. **exercise2.git**)
- Submit on elearning.
- Deadline: **refer on elearning**
- Only one member from each pair needs to do the submission.