Computer Science 2A

Practical Assignment 08

Assignment date:	2024-04-30
Deadline	2024-05-07 12h00
Marks: 100	

This practical assignment must be uploaded to eve.uj.ac.za **before** 2024-05-07 12h00. Late[1] or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of documentation. Marks will be deducted if these are not present. Every submission **must** include a batch file unless stated otherwise.

The **reminder page** includes details for submission. Please ensure that **ALL** submissions follow the guidelines. The reminder page can be found on the last page of this practical.

This practical aims to familiarise you JavaFXGUI

The **Firework Management Bureau (FMB)**[2] believes your **Graphical User Interface (GUI)** is ... umm ... somewhat modern? However, they feel it is missing something... Oh yes, I remember now: They would like to visualize the Firework Display to truly immerse themselves in the Lead Pyrotechnician's creativity.

Some layman has done most of the heavy lifting for you, enabling you to focus on your ST3 next Tuesday... yes, that's right... don't say you did not know X_X. You have been provided with a **p08.jar** file that contains particle systems. What you are required to do is the following:

In your `FireworkDisplayCanvas`:

- Maintain a reference to a `RocketFireworkSystem` and `FountainFireworkSystem`.
- When setting firework entities, ensure to add each firework to its corresponding `ParticleSystem`.
- Create a method called `startSimulation` where you use an `AnimationTimer` that will call the `updateAndShow` methods of the `ParticleSystems`.
- Since fireworks are best seen at night, set the background to black.

In your `FireworkDisplayPane`:

- Add a button named `Simulate`. When the button is clicked, it should call the `startSimulation` method of the `FireworkDisplayCanvas`. This button should be placed at the bottom of your `BorderPane`.

[1] Alternate arrangements for exceptional circumstances will been posted on eve.

[2] Disclaimer - This series of problem statements are a work of ficion. Names, characters, businesses, places, events and incidents are either the products of the author's imagination or used in a fictitious manner. Any resemblance to actual persons, living or dead, or actual events is purely coincidental.

Then make the following changes to your `Main` class:

- Make the class extend `Application` (*javafx.application.Application*)
- Remove unnecessary imports
- Implement the missing `start` method required by the JavaFX `Application`
- In your `main` method, launch the JavaFX Application (this should be the only code in the `main`)
- Your class you will need to instantiate an instance of the `FireworkDisplayPane`
- Add the `FireworkDisplayPane` instance to a `Scene` and load it onto the `Stage` provided by the `Application`
- Show the `Stage` (with the loaded `Scene`)

Remember to place the relevant classes into the `acsse.csc2a.fmb` subpackages[3]

Hints

- Look into the JavaFX AnimationTimer.
- Read your Marksheet!

[3] Hint: UI classes such as `FireworkDisplayPane` should appear in the `acsse.csc2a.fmb.gui` subpackage.

Marksheet

1. Updated UML class diagrams for all classes. [15]
2. `FireworkDisplayCanvas`
   (a) Reference to `RocketFireworkSystem` and `FountainFireworkSystem` [06]
   (b) Update `setEntities` method [10]
   (c) Implement `startSimulation` [10]
3. `FireworkDisplayPane`
   (a) Create `Simulate` Button [02]
   (b) Invoke `startSimulation` from `FireworkDisplayCanvas` [02]
4. `Main`
   (a) Extends `Application` [02]
   (b) Has `start` method [02]
   (c) `main` launches application [02]
   (d) Has `FireWorkDisplayPane` instance [02]
   (e) Adds `FireworkDisplayPane` to `Scene` and loads it onto `Stage` [02]
5. Packages [05]
6. Coding convention (structure, layout, OO design) [05]
7. Commenting (normal and JavaDoc commenting) [05]
8. Correct execution [30]

NB

Submissions which **do not compile** will be capped at 40%!

Practical marks are awarded subject to the student's ability to explain the concepts and decisions made in preparing the practical assignment solution. (Inability to explain code = inability to be given marks.)

Execution marks are awarded for a correctly functioning application and not for having related code.

Reminder

Your submission must follow the naming convention below.

`SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER`

Example

Surname	Berners-Lee	**Module Code**	CSC02A2
Initials	TJ	**Current Year**	2024
Student number	209912345	**Practical number**	P08

`Berners-Lee_TJ_209912345_ CSC02A2_2024_P08`

Your submission must include the following folders:

Folder	**State**	**Purpose**
`bin`	*Required*	Should be empty at submission but will contain runnable binaries when your submission is compiled.
`docs`	*Required*	Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. All files must be in **PDF** format. Your details must be included at the top of any **PDF** files submitted. **Do not include generated JavaDoc.**
`src`	*Required*	Contains all relevant source code. Source code must be places in relevant sub-packages! Your details must be included at the top of the source code.
`data`	*Optional*	Contains all data files needed to run your solution.
`lib`	*Optional*	Contains all libraries needed to compile and run your solution.

NB

Every submission **must** include a batch file that contains commands which will:

- Compile your Java application source code.
- Compile the associated application JavaDoc.
- Run the application.

Do not include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.

Multiple uploads

Note that only **one** submission is marked. If you already have submitted once and want to upload a newer version then submit a newer file with the same name as the uploaded file in order to overwrite it.