Computer Science 2A

Practical Assignment 09

Assignment date:	2024-05-14
Deadline	2024-05-21 12h00

Marks: 140

This practical assignment must be uploaded to eve.uj.ac.za **before** 2024-05-21 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 Multithreading

The **Firework Management Bureau (FMB)**[2] has taken a peek at your **Graphical User Interface (GUI)** and finds it somewhat modern. However, it seems that the more the Lead Pyrotechnician thinks, the less creative the fireworks become. Perhaps a bit of spontaneity could ignite the spark of creativity?

In a bid to keep things lively and unexpected, the FMB is requesting the development of an on-demand **Firework Display System** within your GUI. This new system should be capable of generating up to 10 unique **Firework Displays**, each bursting with spontaneity and a flair for the dramatic. The idea is to reduce overthinking and increase creativity, ensuring that each firework display is not only distinct but also a delightful surprise to the senses.

While reviewing the digital remnants of a developer whose display was, shall we say, *somewhat not modern* (before we gave them the proverbial boot), we stumbled upon some intriguing code they were tinkering with. It appears this code can conjure up firework displays, provided it knows where to store the output:

- **FireworkDisplayGenerator.GenerateSmallDisplay(dataRoot)**: Zap! In just 5 seconds, you have a display.
- **FireworkDisplayGenerator.GenerateMediumDisplay(dataRoot)**: Hold tight! 15 seconds to showtime.
- **FireworkDisplayGenerator.GenerateLargeDisplay(dataRoot)**: A 30-second wait for a grander spectacle.

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

[2]Disclaimer - This series of problem statements are a work of fiction. 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.

- **FireworkDisplayGenerator.GenerateGoAllOutDisplay(dataRoot)**: The ultimate show needs 60 seconds of patience.

These methods are magical, but let's be honest, who has the time to wait even a minute for a firework display in this fast-paced world? We need to create **10** "Go All Out" displays, and we certainly don't have 10 minutes to spare! Thus, the **Firework Management Bureau (FMB)** tasks you with an urgent mission: Turbocharge this code! Make it multi-threaded and integrate it into our somewhat modern GUI. What you are required to do is the following:

Create a **`GenerateTask`** class in the `acsse.csc2a.fmb.threading` package:

- Make the class *Runnable*.
- The constructor should take a **`E_TASK_SIZE`** and a dataRoot String as parameters.
- When your GenerateTask is run, it should call the method that matches the **`E_TASK_SIZE`**.

In your `FireworkDisplayPane`:

- Add a Generate **`Menu`** to the **`MenuBar`**.
- Create a **`MenuItem`** for each **`E_TASK_SIZE`** and add them to the Generate **`Menu`**.
- When a **`MenuItem`** is clicked, use *Multi-threading* to run *10* GenerateTasks simultaneously.

After you finish creating the layout files, load them and enjoy the surprise of randomness. Good luck with running the Go All Out file!

You have been given a `P09 Starter Project` that includes a `p09.jar` file. Make sure to put the relevant classes into the `acsse.csc2a.fmb` subpackages.

Hints

- Read your Marksheet!

Marksheet

1. Updated UML class diagrams for all classes. **[15]**
2. **`GenerateTask`**
   (a) Make the class ***`Runnable`*** **[05]**
   (b) Constructor **[10]**
   (c) ***`Run`*** method **[20]**
3. **`FireworkDisplayPane`**
   (a) Create Generate ***`Menu`*** **[05]**
   (b) Add a ***`MenuItem`*** for each **`E_TASK_SIZE`** **[20]**
   (c) Use *Multithreading* to run *10* **`GenerateTask`**s simultaneously **[20]**
4. Packages **[05]**
5. Coding convention (structure, layout, OO design) **[05]**
6. Commenting (normal and JavaDoc commenting) **[05]**
7. 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**	P09

```
Berners-Lee_TJ_209912345_ CSC02A2_2024_P09
```

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.