Computer Science 2A

Practical Assignment 05

Assignment date: 2023-04-18

Deadline 2023-04-25 12h00

Marks: 115

This practical assignment must be uploaded to eve.uj.ac.za **before** 2023-04-25 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 with JavaFX.

The **Milky Way Space Communication Board (MWSCB)**[2] is pleased with your progress but now require a **Graphical User Interface(GUI)** for your application (to make it more modern).

Create a **ShipPane** class that extends **StackPane** (*javafx.scene.layout.StackPane*). This will act as the root node for your Scene. The **ShipPane** class will have the following components:

- A **MenuBar** containing a **MenuItem** to open a **FileChooser** to select the file to open
- A **TitledPane** control that provides an expandible container for **Ship**, including:
  - Properties of the **Ship** in a **GridPane** layout
- A **ListView** control to hold and display the **Message**s data, including:
  - **TitledPane** to display each of the derived **Message** class objects
  - Each derived **Message** class i.e. **SOSMessage**, **EncryptedMessage** and **NormalMessage** object is placed inside its own **GridPane** (for layout of content)
- You may use the following optional controls (or others that you find useful):
  - **ScrollPane**: Provides a scrollable container where necessary
  - **HBox** and **VBox**: Provide containers for horizontal and vertical stacking of nodes respectively (provided image)

You are provided with a library file (from the last practical) that contains the **Ship**, **Message**, **SOSMessage**, **NormalMessage** and **EncryptedMessage** model classes. In addition, it includes a **DataReader** class with a ***readShipFile*** method that takes a File as a parameter and returns

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

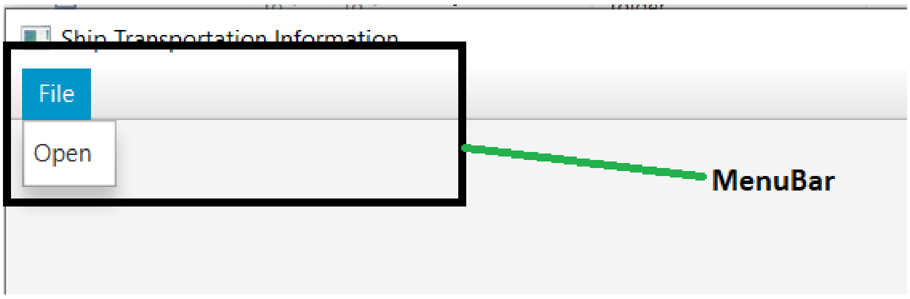
[2] Disclaimer - This series of problem statements are a work of fiction. Names, characters, businesses, places, events and incidents are either the products of the author's imagination or used in a fictitious manner. Any resemblance to actual persons, living or dead, or actual events is purely coincidental.

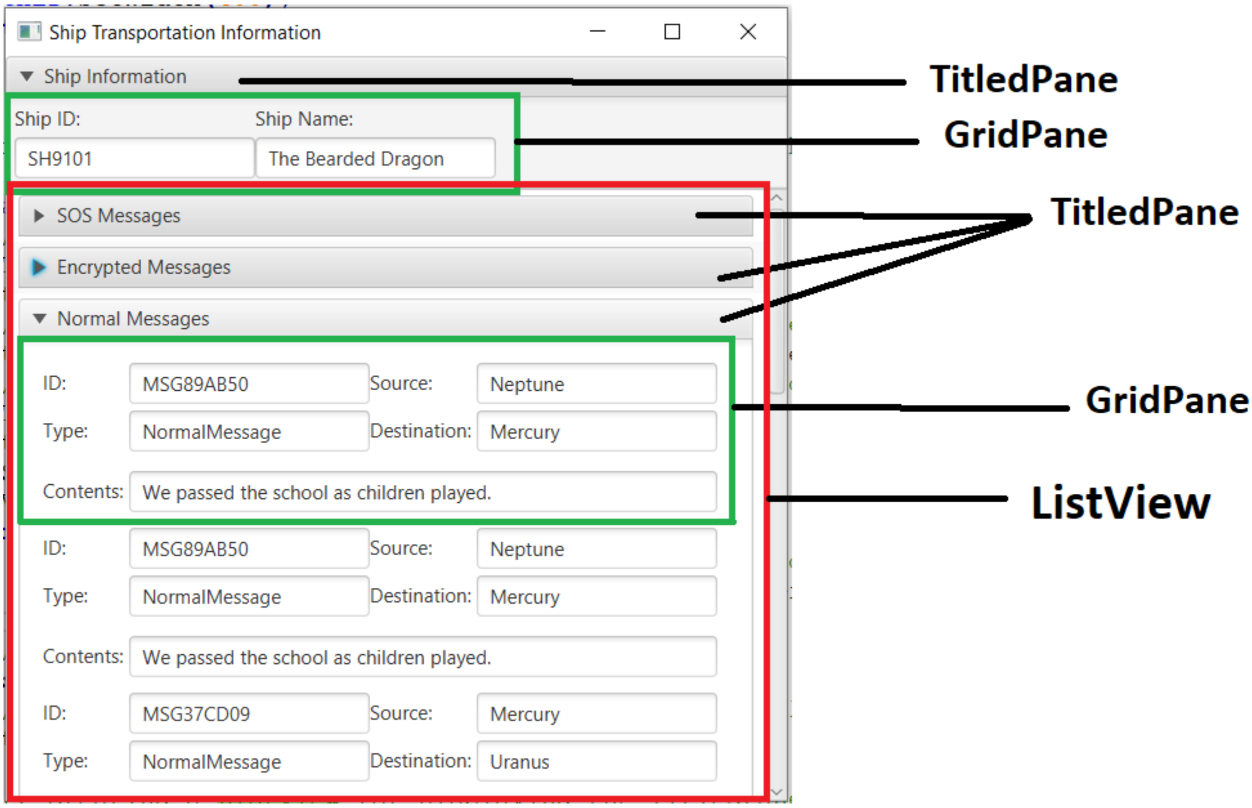
a **Ship** instance. The **Ship** class has a ***getMessages*** method that returns an array of **Message**s. You are also provided with a file named shipInfo.dat that you are required to read data from.

Your **Main** class should contain the following:

- **Main** must extend **Application** (*javafx.application.Application*), and you must implement the main method correctly[3]
- Implement the ***start*** method required by the JavaFX **Application**
- In your ***main*** method, launch the JavaFX Application
- Your class you will need to instantiate an instance of the **ShipPane**
- Add the **ShipPane** instance to a **Scene** and load it onto the **Stage** provided by the **Application** and show the **Stage**

Here is an *example* of how you can lay out your **Application** and its **ShipPane**.





Ship ID: SH9101
Ship Name: The Bearded Dragon

ID: MSG89AB50 Source: Neptune
Type: NormalMessage Destination: Mercury
Contents: We passed the school as children played.

ID: MSG89AB50 Source: Neptune
Type: NormalMessage Destination: Mercury
Contents: We passed the school as children played.

ID: MSG37CD09 Source: Mercury
Type: NormalMessage Destination: Uranus

Remember to place the relevant classes into the **csc2a.acsse** subpackages[4]

[3] Refer to Lecture Notes

[4] Hint: UI classes such as **ShipPane** should appear in the **csc2a.acsse.gui** subpackage.

Marksheet

1. Updated UML class diagrams for all classes created (Do not include **`Main`**). **[10]**
2. **`ShipPane`**
   (a) **`MenuBar`** with **`MenuItem`** **[02]**
   (b) **`FileChooser`** to select file **[02]**
   (c) **`Ship`** instance by calling **`DataReader.readShipFile`** **[02]**
   (d) **`TitledPane`** for **`Ship`** **[05]**
   (e) **`ListView`** for **`Message`** **[10]**
   (f) **`TitledPane`** for derived **`Message`** classes, i.e. **`SOSMessage`**, **`EncryptedMessage`** and **`NormalMessage`** **[15]**
   (g) Use of **`GridPane`**s to display **`Ship`** information **[05]**
   (h) Use of **`GridPane`**s to display **`Message`** information **[05]**
3. **`Main`**
   (a) Extends **`Application`** **[01]**
   (b) Has ***`start`*** method **[02]**
   (c) ***`main`*** launches application **[01]**
   (d) Has **`ShipPane`** instance **[05]**
   (e) Adds **`ShipPane`** to **`Scene`** and loads it onto **`Stage`** **[10]**
4. Packages **[05]**
5. Coding convention (structure, layout, OO design) **[05]**
6. Commenting (normal and JavaDoc commenting) **[05]**
7. Correct execution **[25]**

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**	2023
Student number	209912345	**Practical number**	P05

```
Berners-Lee_TJ_209912345_ CSC02A2_2023_P05
```

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.