Computer Science 2A

Practical Assignment 06

Assignment date:	2023-04-25
Deadline	2023-05-02 12h00
Marks: 115	

This practical assignment must be uploaded to eve.uj.ac.za **before** 2023-05-02 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 the Visitor Design Pattern.

The **Milky Way Space Communication Board (MWSCB)**[2] is pleased with your progress but now require an addition to the **Graphical User Interface(GUI)** to your application from last week. You must do so making use of the newly provided data file, inside the data folder.

You are provided with a library (jar) and two text files, and need to create a JavaFX GUI application. In order to complete the practical,you have to implement the following:

- Create a **`SpaceShip`** class that extends the **`Ship`** class from the provided library (jar) and has the following additional properties:
  - **shipPosition** of type **`Point2D`** [3]
- Create a **`Planet`** class that has the following properties:
  - **name** of type String
  - **planetPosition** of type **`Point2D`** [4]
  - **color** of type **`Color`**
  - **radius** of type Integer
- Create a **`MyCanvas`** class that:
  - Extends **`Canvas`**
  - Sets the **`Canvas`** properties including **`ArrayList`** of **`Planet`** and **`SpaceShip`**
  - Has a ***`repaintCanvas`*** method that will get the ***`getGraphicsContext2D`***, then send the visitor to draw the shapes (**`SpaceShip`** and **`Planet`**) [5]

[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.

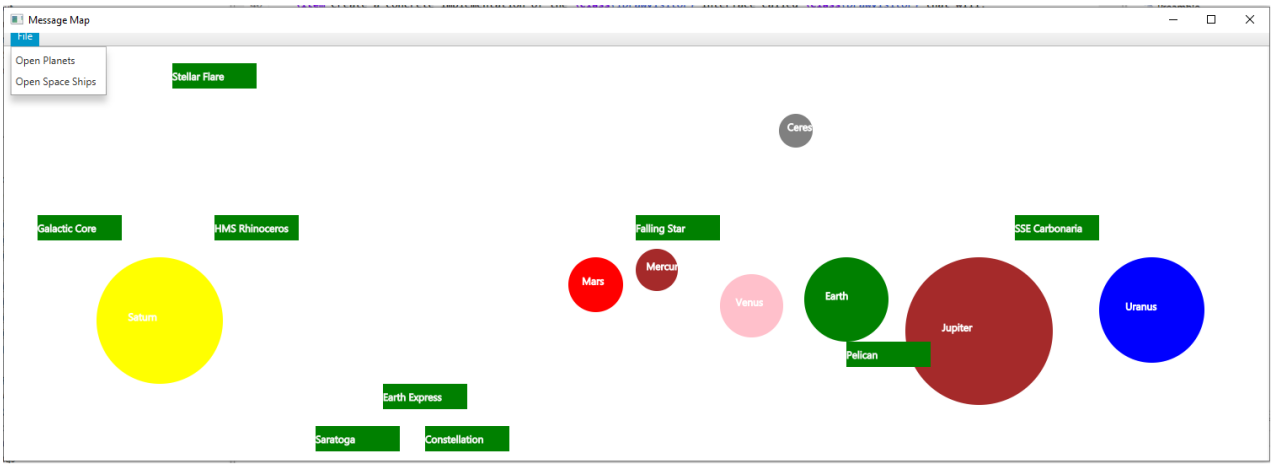
[3] Hint - JavaFX class

[4] Hint - JavaFX class

[5] Hint - The accept method

- Create a **MyPane** class that:
  - Extends **StackPane**
  - Has an **ArrayList** of **Planet** and **SpaceShip**
  - Has a **MenuBar** containing two **MenuItem** to open different files (**SpaceShip** and **Planet**)
  - An actionListener for each of the **MenuItem**s

You are required to complete the system by implementing a **Visitor Design Pattern** to display all the information as shown in the diagram below.



The **Visitor Design Pattern** will require that you do the following.

- Create a **IDrawable** interface with an ***accept*** method that accepts the **IDrawVisitor**
- The **Planet** and **SpaceShip** class should provide concrete implementations of the method from **IDrawable** interface
- Create a **IDrawVisitor** interface with ***draw*** methods that will draw instances of **Planet** and **SpaceShip** when implemented
- Create a concrete implementation of the **IDrawVisitor** interface called **DrawVisitor** that will:
  - Draw a Rectangle to represent the **SpaceShip**
    - Draw the **SpaceShip** in the Position provided
    - Set the Rectangle Color to GREEN
    - Draw (write) the name of the **SpaceShip** in the rectangle
    - Make sure it looks good
  - Draw an Oval to represent the **Planet**
    - The radius is included in the text file
    - Place the **Planet** in the Position (x, y) provided
    - The color of the **Planet** is included in the text file
    - Add the name of the **Planet** in the Oval
    - The size of the **Planet** is included in the text file

In addition create a **`FileIO`** class with:

- A static ***`readSpaceShip`*** that takes a **`File`** as a parameter and returns an **`ArrayList`** of **`SpaceShip`**
- A static ***`readPlanet`*** that takes a **`File`** as a parameter and returns an **`ArrayList`** of **`Planet`**

The files you are provided with is formated in the following way (Values are separated by a space):

Planet Data Format

```
PlanetName Colour x y radius
```

SpaceShip Data Format

```
x y SpaceShipID SpaceShipName
// SpaceShipName can be many strings long
```

Marksheet

1. Updated UML class diagrams for all classes. **[10]**
2. **`SpaceShip`**
   (a) Extends **`Ship`** **[01]**
   (b) Has property shipPosition of type **`Point2D`** **[01]**
   (c) Implement method from **`IDrawable`** interface **[05]**
3. **`Planet`**
   (a) Has the properties name, colour and radius **[03]**
   (b) Has property planetPosition of type **`Point2D`** **[02]**
   (c) Implement method from **`IDrawable`** interface **[05]**
4. **`MyCanvas`**
   (a) Extends **`Canvas`** **[01]**
   (b) Has ***`repaintCanvas`*** method **[04]**
   (c) ***`accept`*** method for **`SpaceShip`** **[02]**
   (d) ***`accept`*** method for **`Planet`** **[02]**
   (e) **`ArrayList`** of objects **[05]**
5. **`MyPane`**
   (a) Extends **`StackPane`** **[01]**
   (b) **`ArrayList`** of objects **[02]**
   (c) **`MenuItem`**s **[02]**
6. **`FileIO`**
   (a) ***`readSpaceShip`*** method **[02]**
   (b) ***`readPlanet`*** method **[02]**
7. **`IDrawable`** interface **[05]**
8. Concrete Visitor **`IDrawVisitor`** class
   (a) Implements **`IDrawVisitor`** **[05]**
   (b) Draw Oval (**`Planet`**) **[05]**
   (c) Draw Rectangle (**`SpaceShip`**) **[05]**
   (d) Add text to **`Planet`** **[05]**
   (e) Add text to **`SpaceShip`** **[05]**
9. **`Main`** **[10]**
10. 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**	P06

`Berners-Lee_TJ_209912345_ CSC02A2_2023_P06`

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.