Object-Oriented Programming 2022/2023

Lab Sheet #13

Objectives

- Objective consolidation of the last labs.
- Advanced JavaFX properties.
- Windows and Modals.

Schedule

- Use of advanced properties.
- Use of windows.
- Management of multiple screens.

Implementation rules

- Use the BlueJ IDE with the provided code.
- Implement the necessary code as needed and test at the end of each level.
- Use the coding conventions adopted for the Java language (see **Notes**).

Implementation

Level 1:

Accept the Assignment GitHub Classrooms for this lab via the https://classroom.github.com/a/A3cQcQl_ link.

Open the template code provided in the lab repository and review the provided classes and understand how they work.

In the MenuPane class:

- Create a button called btnStart that has the text "Play Now" in the *init()* function.
- Change the size, background color and font color to your liking.
- Place it in the center of the BorderPane attribute of the MenuPane class.

Also add a new instance of the **InitialMenuBar** class and place it on top of the BorderPane attribute of the MenuPane class.

Fill the button with the action that is given below and understand how it works.

```
btnStart.setOnAction(e -> {
    primaryStage.setScene(new GamePane(primaryStage)
});
```

Level 2:

- Create a TileButton class that extends from a Button class from JavaFX, this class will be responsible for discovering the bombs and marking the flags.
- This class will have a **Tile** attribute that it will receive through its constructor.

Add the following *click()* method.

```
public void click() {
    if(MinesweeperLogic.getInstance().isFlaggingMode()){
        markFlag();
        return;
    }

    if(tile.isFlagged()) return;
    if(tile.isIsUnveiled()) return ;


    tile.setUnveiled(true);

    setText( tile.getValue() + "");

}

public void changeButtonSize(ImageView image, int xPixels, int yPixels) {
    image.setFitHeight(yPixels);
    image.setFitWidth(xPixels);
}
```

- Create the getTile() method
- Create a function similar to the *click()* function, that will not return values, and will be called *markFlag()*. This function will have to check if the tile attribute is flagged with the value **true** through the *isFlagged()* function.
- If it is flagged, change the button text to "", otherwise change it to "Flag".
- Finally invoke the *toggleFlagged()* function of the tile attribute after changing the Button text in the *markFlag()* function.

Level 3

To complete the **GamePane** that will be the Scene that will hold the Minesweeper game add a new instance of the **GameMenuBar** class at the top of your BorderPane. This class will have several options and features.

In the **GameMenuBar** class in the *init()* function

- Create a Menu of the **Menu** class, named "File".
- The "File" menu will have three **MenuItem** named "Help", "Home Menu" and "Exit".

The "Exit" MenuItem closes the application, the "Start Menu" will navigate to the "MenuPane" and the "Help" will create a new Alert with the text contained in the getHelpText() function.

Implement each of the MenuItems, and add them to the **Menu** menuFile. As you can see, the "Settings" Menu is already implemented.

Check the operation of the application. At this stage the application will be functional.

Level 4

As you can see, the application marks pumps as -1, empty positions as 0, flags as "flag" and numbers as themselves. But it does not present an appealing visual. For this reason:

- Create an attribute in the **TileButton** class that will be of type ImageView and will have the picture "Undiscovered.png" as is demonstrated in the following code:

```
private ImageView image = new ImageView(new Image("Images/Undiscovered.png"));
```

- Add the images contained in the ***Images** folder by replacing the *setText()* methods you used in level 2 of this lab with the *click()* and *markFlag()* methods of the **TileButton** class.

Add the necessary logic so that depending on the value of the tile, it will use the correct Image.

Don't forget to add the **ImageView** to the **TileButton**. Use the given function *changeButtonSize()* with the values of xPixels and yPixels at 50 to keep the buttons at a fixed size.

Level 5

To complete the game features, create a **GameOverStage** class that will extend the Stage class.

- Create two attributes in this class, the first being a **Scene** and the other a **BorderPane**.
- Initialize each attribute in the class constructor. Note: the **Scene** attribute will have a size of 150 pixels by 100 pixels.
- The constructor should be given *Stage primaryStage*.
- Create an init method that will take as parameter the primary Scene and will create a Text node with the string "Game Over", and a Button with the string "Try Again".
- Add the text and the button to the BorderPane class attribute
- Change the button's setOnAction behavior to use the *Singleton* MinesweeperLogic and execute the *setGame()* function. Still within *setOnAction()*, execute the *close()* function.
- Finally, change the window/Stage properties so that the window cannot be closed. Use the following functions:

```
initModality(Modality.APPLICATION_MODAL);
initStyle(StageStyle.UNDECORATED);
```

To conclude, add an instance of the GameOverStage class to the init method of the **GamePane** class where there is a console printout of "Bomb!"

Notes:

For identifiers follow the conventions adopted normally, in particular:

1. The **camelCase** notation for the name of local variables and identifiers for attributes and methods.
2. The **PascalCase** notation for class names.
3. Do not use the '_' symbol or abbreviations for identifiers.