Homework 6: Civilization JavaFX

Marc Marone, William Syre, Tian-Yo Yang

Introduction:

Civilization is a turn-based game centered around founding and building a civilization. More commonly referred to as Sid Meier's Civilization, the game has been around since 1991 and has since developed somewhat of a cult following. This semester we are going to be implementing our own version of this game for your homework assignments and by the end of the semester you will have a working game you can show off to your friends! If you are not familiar with Civilization, there is a free version that you can play in order to get familiar with the game.

For this assignment, you will be making a Graphical User Interface(GUI) using JavaFX. This assignment will be:

- Assessing your ability to understand/utilize objects from the javaFX library.
- Test your ability to understand and implement event handlers to handle certain actions.

Problem Description:

Now that you know the ins and outs of Java, it's time to get some practice writing some JavaFX code. We'll do this by using JavaFX to create a GUI for the Civ game.

Before we begin ……

- Like all homework descriptions, it is **IMPERATIVE** that you read and understand this entire document.
- It is important to note that you will not need to create a new repository for this homework. You will be working in the same git repository you set up for the first homework.
- This homework will be very focused on the display and functionality of your GUI. So make sure you read the description carefully and implement the functions and display described!
- Most of this code is revised from HW04 code, so you will need to replace any duplicated files in your repository with the new versions.
- Check out Helpful Hints at the end of this document for some tips on the homework.

Solution Description

For this assignment, you will be implementing classes in the runner folder and the view folder of your application to create a GUI for the Civ game that you guys have been working on throughout the semester.

runner.CivilizationGame.java

This class is the class that get run upon running your civilization_fall2016 application. This class is responsible of launching and displaying your javafx application upon running the program. When the program begins running, this class should first display the layout view of the *StartScreen.* Upon

clicking the “Start” button in *StartScreen* there should be a input text dialog box that pops up and prompts the user to enter the name of their first settlement. After entering the name of the first settlement, the scene of the application should switch to display the layout view of *GameScreen*.

For this class, you are responsible for implementing:

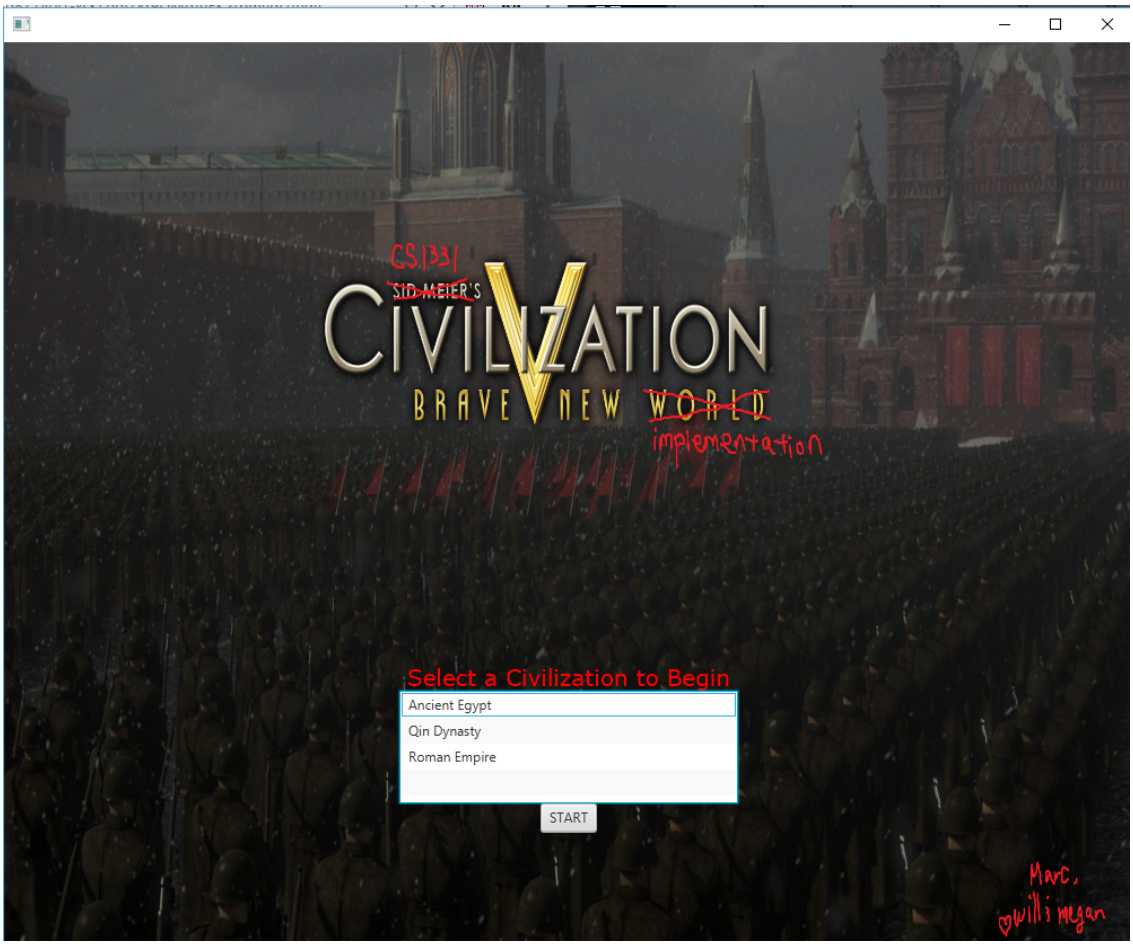
- void start(Stage primaryStage)
    - this method is called upon running/launching the application. This method should display a scene onto the stage.
- Scene startGame()
    - This method handles the logic for switching scenes from *StartScreen* to *GameScreen* upon clicking the start button after selecting a Civilization.
    - When the start button is pressed after selecting a Civilization, an input text dialog box should pop up and prompt the user to enter the name of their first settlement.
    - After the user enters the name for the first settlement, your method should create a new instance of the Civilization that the user has chosen and add the civilization and the provided name to the *Map* of your game.Your method should also set the civilization of the *Game*.
    - You will also need to add a new *Bandit* to your *Map* and update your *GameScreen* when the user presses the start button.
    - It should then set the scene to display the layout view of *GameScreen* and update the *GameScreen*.
    - Note: if the start button from the *StartScreen* has NOT been pressed, then the program should still display the layout view of StartScreen

Hints:

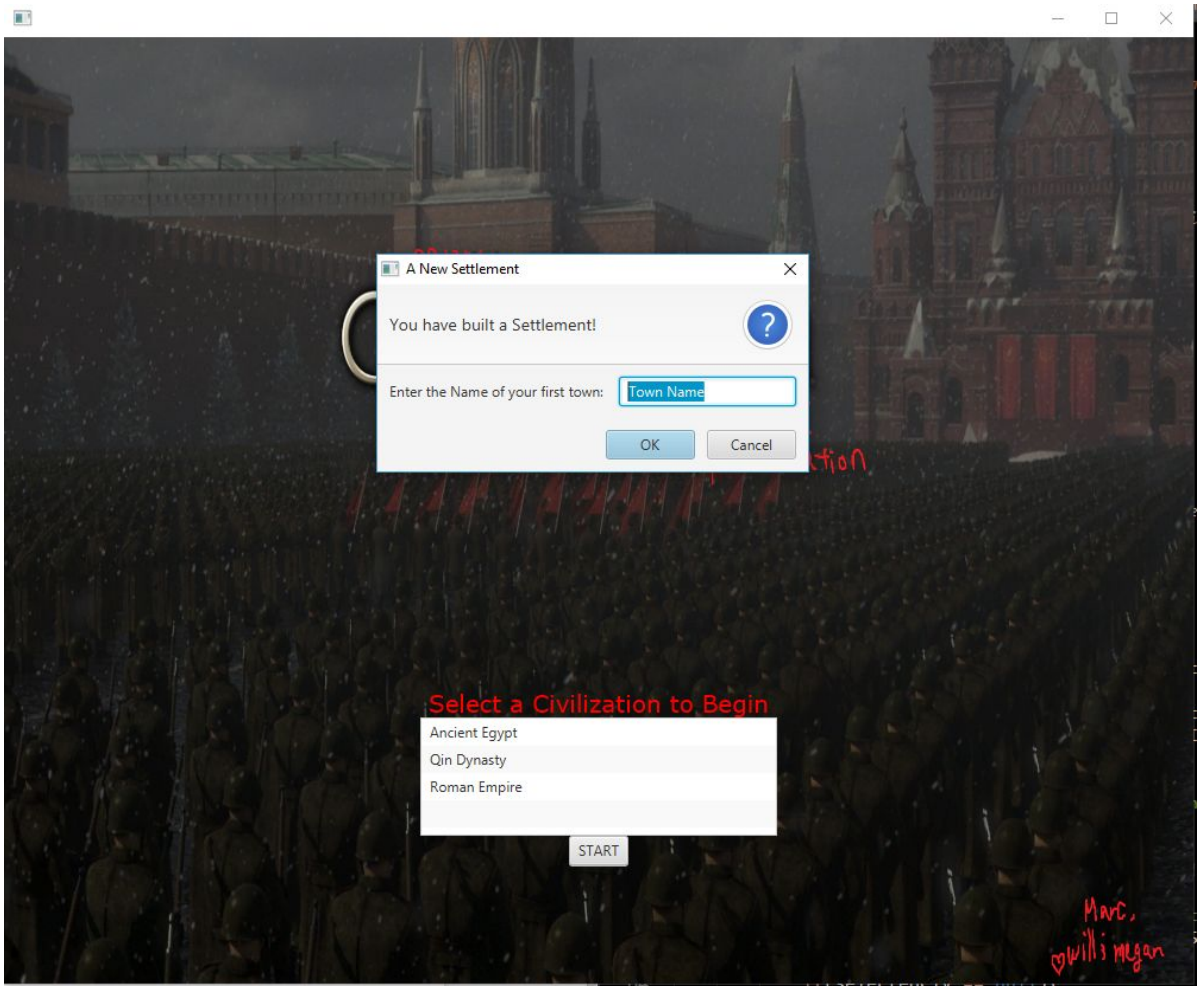
- Check out the classes listed in the import statements. You might wanna check out some of the methods within those classes for your implementation.
- TextInputDialog

Here is what the flow of your program should look like:

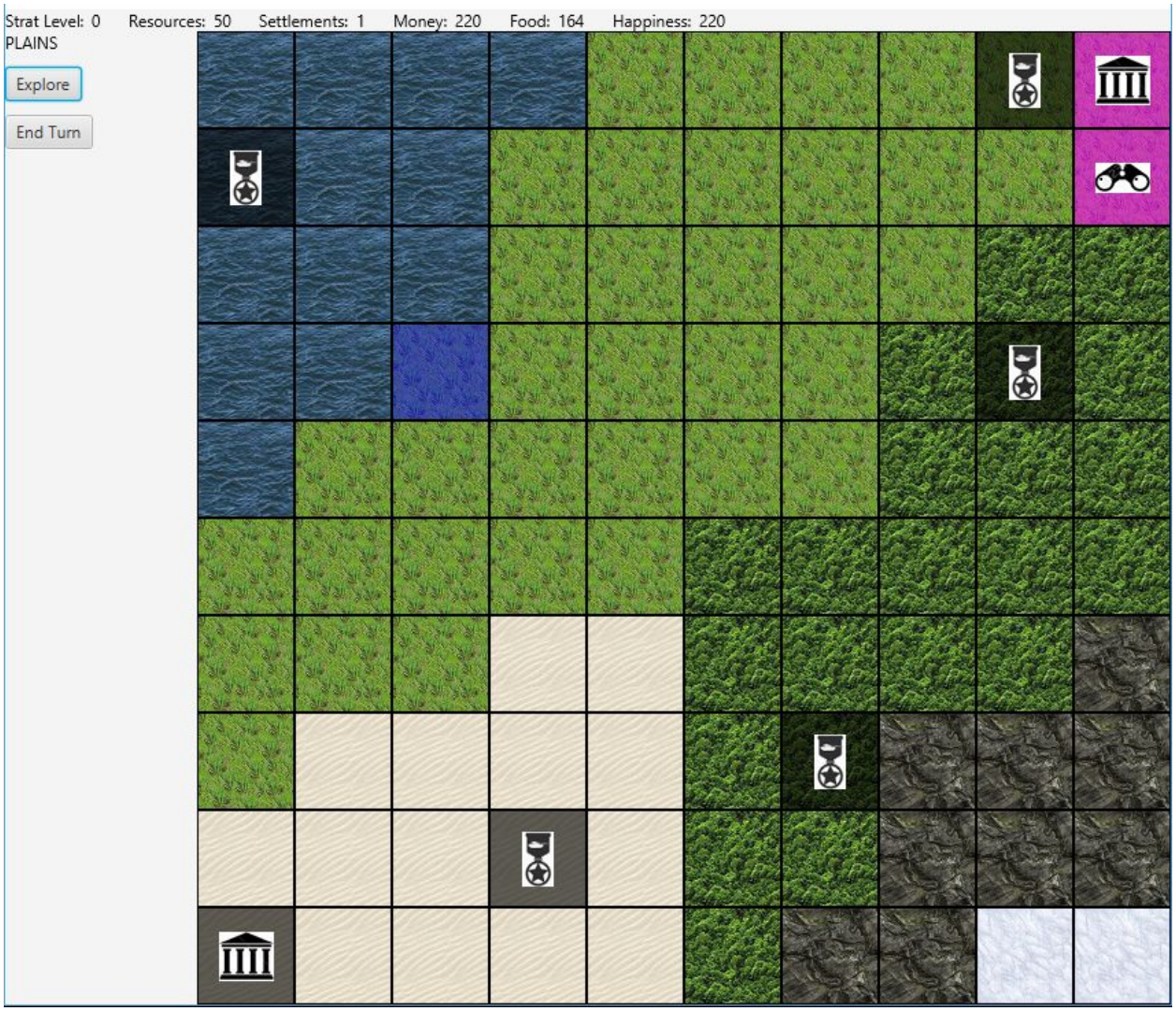
a.) Upon starting your application, it should display the *StartScreen*



b.) after choosing a Civilization and upon pressing the start button, there should be an input dialog text box that pops up to prompt the user to enter the name of their first settlement.



c.) After the user enters the name of their first settlement and click the ok button, the scene should switch to display the *GameScreen*



view.StartScreen.java

This class represents the layout of the start screen, the screen that is displayed upon launching the application.

Your StartScreen layout must have the following:

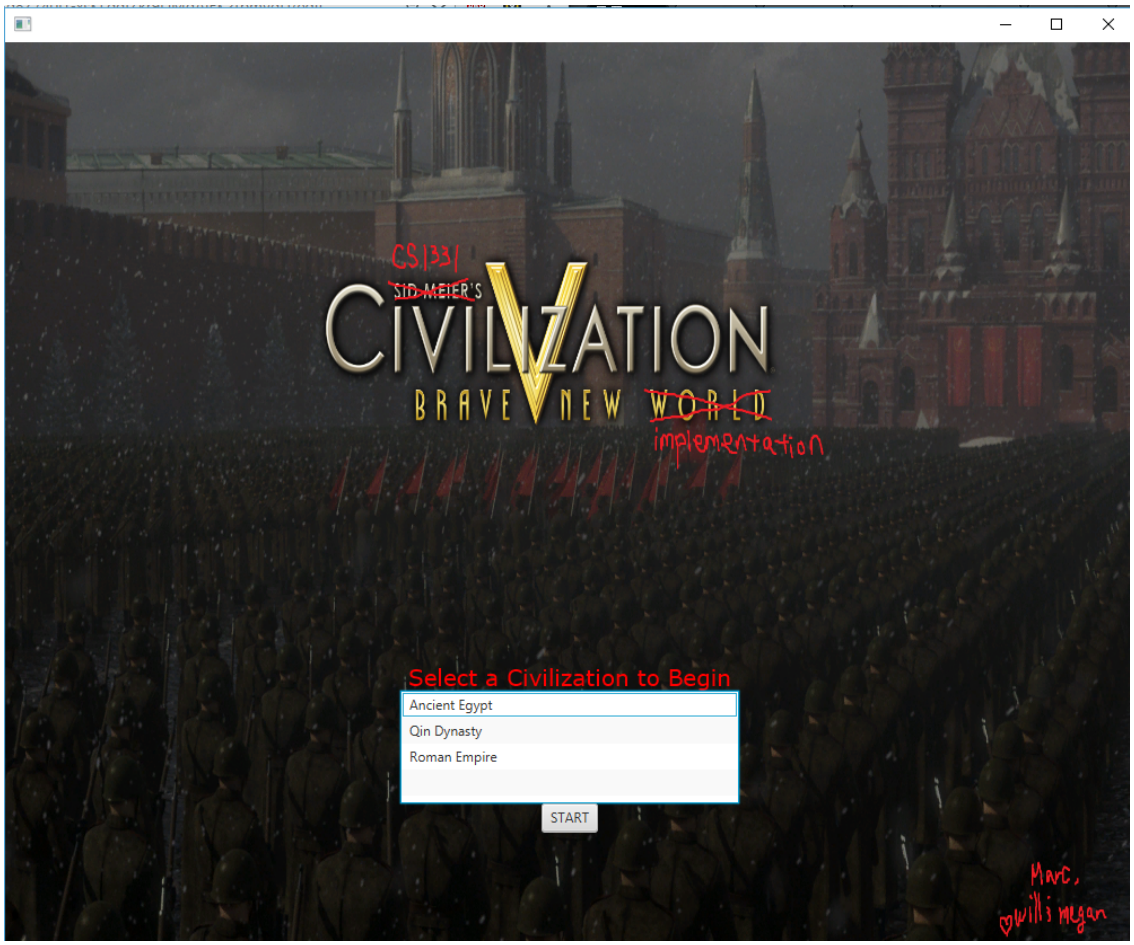
- A background image. The image itself has been provided to you and the url of the image is "File:./src/main/java/view/civ_background.png"
- A list of civilizations for the user to select from. This is where the CivEnum should be used.
- A start button

Hint: Here are some classes that you may want to look into. In addition, check out the classes listed in the import statements.

- StackPane

- ObservableList
- ListView
- ImageView

The display of your StartScreen layout should roughly like this:



view.GameScreen.java

This class represents the layout of your Civilization game once the game has begun. This layout includes the logic of displaying the different action menus as well as the map and resource bar. Upon clicking a tile that a worker/unit/building is occupying, there should be a corresponding menu action that should be displayed the actions that the selected MapObject is able to make. The *GridFX* map of your game should be set in the center of the *GameScreen* layout. The action menus should be set at the left of *GameScreen* layout. The resource bar should be set at the top of your *GameScreen* layout.

void update()

- This method should update the *GridFX* of your game

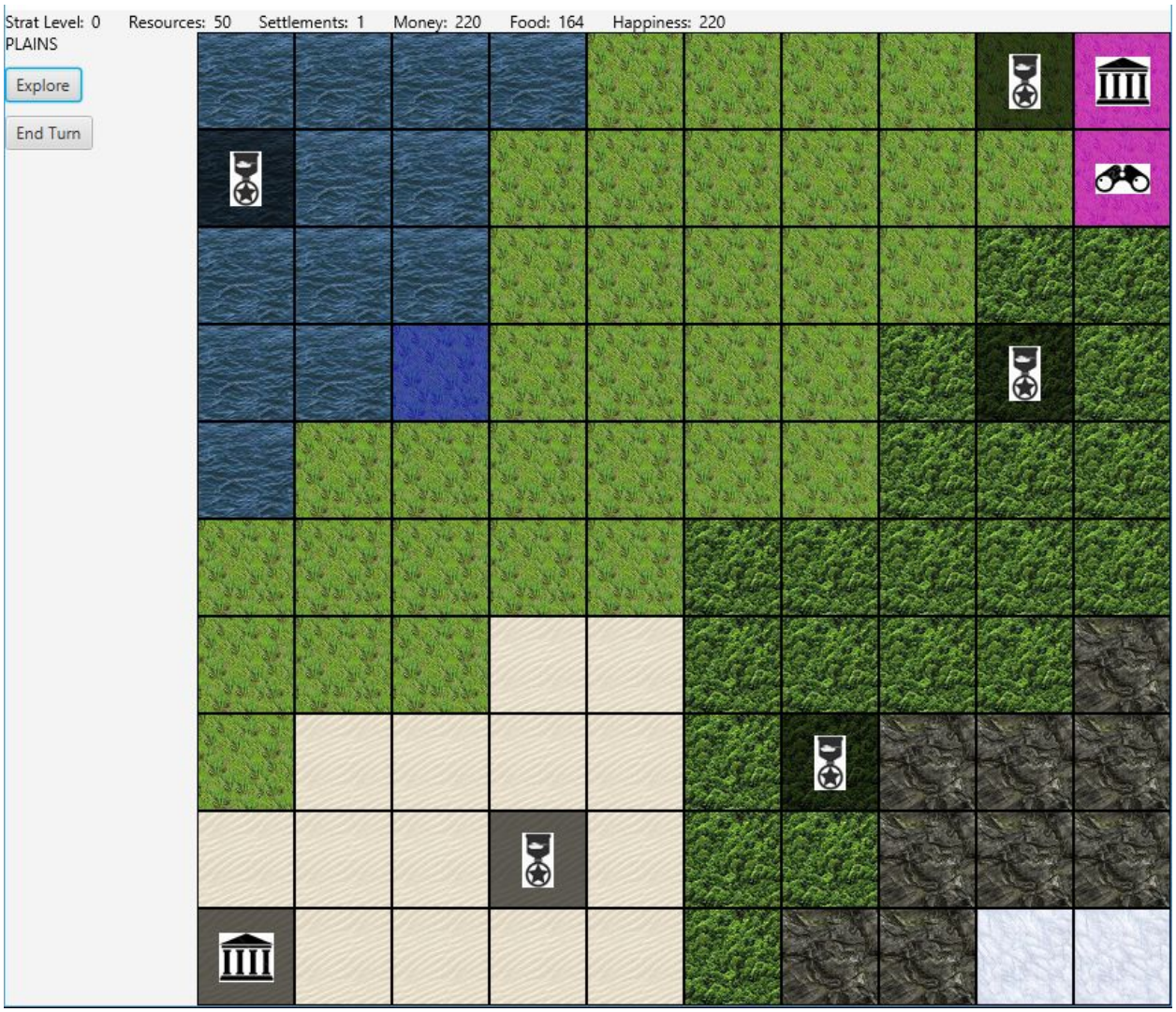
Void switchMenu(Game.GameState state)

The argument taken in by the function, state, represents type of tile that has been selected.

If the tile selected is a:

- MILITARY: the action menu displayed should display the *MilitaryMenu*
- WORKER:the action menu displayed should display the *WorkerMenu*
- RECRUITING: the action menu displayed should display the *RecruitMenu*
- BUILDING: the action menu displayed should display the *BuildingMenu*
- NEUTRAL: meaning that there are no occupants on the tile, then action menu displayed should display the *StatusMenu*

This is what your *GameScreen* should look like when displayed:



view.MilitaryMenu.java

This menu is-a(n) *AbstractMenu*. When this menu is displayed on *GameScreen*, the tile selected is able to move or attack. Therefore, in the menu, there should exist a button for attack and a button for move.

- When the attack button is pressed, it should let the game know that it is in an attacking state. To actually attack an enemy unit, the user will first select their own military unit, press the attack button, and then select an enemy unit. You should also update the resource bar.
- When the move button is pressed, it should let the game know that it is in a moving state. To actually move a unit, the user will first select their own unit, press the move button, and then select the tile in which your unit wants to move to.

If any of the above functions cannot be done, ie. can't attack your own unit, can't move onto a tile, etc, then there should be an alert box that pops up letting the user know that the action cannot be done.

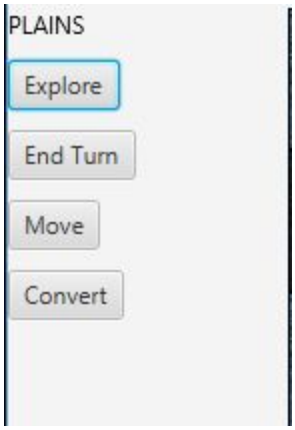
Here is an example of what it should look like

view.WorkerMenu.java

This menu is-a(n) AbstractMenu. When this menu is is displayed, the tile selected should be able to move or convert. Therefore, in the menu, there should exist a button for move and convert.

- The move button functions the exact same way as the move button in *MilitaryUnit*.
- When the convert button is selected, it should get the last clicked *TerrainTile* from the game and check if the tile is convertable. If it is convertable, then it should convert the tile and the view of the tiles should be updated. If it is not convertable an alert box should pop up indicating to the user that the tile selected cannot be converted.You should also update the resource bar.

This is what it should look like:



view.RecruitMenu.java

This menu is-a(n) AbstractMenu and should be displayed when selecting a tile that is a recruiting tile. This tile should display a list of possible workers and units that you can recruit. The list of worker and units include melee unit, range unit, hybrid unit, siege unit, settlers, farmers, coal miners, anglers, and master builder.

When this menu appears upon selecting a recruiting tile, the user is able to select a worker/unit from the menu list to place onto the tile. The selected worker/unit should be set as the occupant of the tile and the view of the tile should be updated. You should also update the resource bar for recruiting a unit/worker.

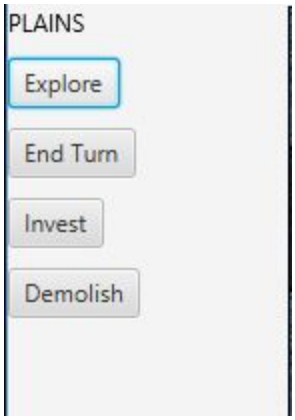
If for any reason the user is unable to perform this action, there should be an alert box that pops up indicating that this action cannot be performed.

view.BuildingMenu.java

This menu is-a(n) *AbstractMenu*. When this menu is displayed on *GameScreen*, the tile selected is able to be demolished or invested. Therefore, there should be an invest and demolish button for this menu.

- When the invest button is clicked, you should get the the occupant of the of the tile and invest in the occupant. Investing costs 25 gold, and should only take place if the owner civ is able to invest. You should also update the resource bar.
- When the demolish button is clicked, you should first check to see if there is more than 1 settlement. If there is more than 1 settlement, check to see if the occupant of the tile is a Settlement. If it is a *Settlement* then the occupant of the tile from the game should be demolished

If the invest and demolish functions cannot be performed for the selected tile, there should be an alert box indicating that the action cannot be performed and provide the reason as to why the action cannot be performed. The actions should also update any other parts of the UI that are affected.



view.StatusMenu.java

This menu is-a(n) *AbstractMenu*. When this menu is displayed on *GameScreen*, the tile selected is able to be explored or the user is able to end turn. Therefore there should be an explore and end turn button. Hint: this has been given to you in *AbstractMenu* but you need to be able to display those buttons

view.ResourceMenu.java

This class should create a resource bar that can display the current state of your resources. It should display the strategy level, amount of resources, settlements, money, food, and happiness.

update()

- This method should update all the older resource values shown on the resource bar to the current state of your resource values

getRootNode()

- This method should first update the resource bar and then return the resource bar.

view.TerrainTileFX.java

void updateTileView()

This method is responsible for highlighting, adding map object icon, and updating the images displayed on the tile. For this method you need to write the following:

- Check if the *TerrainTile* is empty. If it is not empty, fill the overlay with the color of the occupant on the terrain tile . Otherwise set the color of the overlay to be transparent.
- If the tile has been selected in the GameController, highlight it. Otherwise, make sure that it is has only the occupant color.
- If the *TerrainTileFX* contains an icon, remove it
- If the tile is not empty, get the image of the occupant of the tile and add the image of the occupant to the tile.

Extra Credit

The extra credit for this assignment is up to 100 extra points. Meaning that you can potentially get a 200 on this assignment. To earn extra credit for this assignment, you must implement the following features in your working game.

- Improve UI (+20pts)
    - As you can see, right now the UI is extremely simple and just contains all the necessary function for the game. Implement some way of arranging all the layout to display in an organized manner.
- Add 3 New Civilizations (+10pts)
    - Add other Civilizations for the user to choose from
    - These should be different from each other and the provided Civilizations
- Sound Effects (+15pts)
    - When an action is performed, there should be sound effects associated with it.
    - ex.) when the player successfully invests in a Building your game can make "*cha-ching*" sound effect
- Animation (+30pts)
    - When moving a worker/unit to another tile, animate the movement so that the player can see that the unit is moving
- Dynamically Sized Map (+40pts)
    - When the game starts, give the user an option as to which size map they can play on. With a bigger map you could also introduce more Enemy Civilizations!

Compiling and Running

- This project makes use of several different packages to manage its many files. As a result, compiling and running this project is a bit different from projects you have worked on in the past.

- Since our files are in many different packages, we have to make sure they get linked together properly. We do this by setting the class path when we compile and when we run.
- To compile the program in its totality run this command:

```
$ javac -cp src/main/java src/main/java/runner/*.java src/main/java/model/*.java
src/main/java/view/*.java src/main/java/controller/*.java
```

- That command instructs java to compile all of our files while considering src/main/java to be our classpath.
- To run the program, you should run:

```
$ java -cp src/main/java runner.CivilizationGame
```

Checkstyle

- Remember that you may lose checkstyle points for this homework! So, make sure that you are frequently running checkstyle on your code to ensure that you don't lost unnecessary points.
- **The checkstyle cap on this assignment is 100 points, where you lose one point per error.**

Verifying Your Submission

Please be sure that any code you push compiles and runs through the command line! Pull from your repository and make sure everything is working how you want it!