

# Comp3021 Java Programming

## Spring 2018, Programming Assignment #2

### (Wars of the cities – GUI mode)

**(Deadline: 11:55pm, 12 May 2018)**

#### Note:

- This is an individual assignment; all the work must be your own.
- Download the project skeleton at <https://course.cse.ust.hk/comp3021/assignments/assignment2/PA2.zip> and finish the implementations using the skeleton.
- Zip your complete project directory to PA2.zip and submit it to CASS through <https://course.cse.ust.hk/cass/student/>. Refer to <http://cssystem.cse.ust.hk/UGuides/cass/index.html> for the details of uploading assignment using CASS.
- You can submit the assignment for as many times as you wish, we will only mark your latest submission.
- This PA accounts for 10% of the total course grade. **Cheating/plagiarism will be caught and punished HEAVILY!**

University's Honor code: <http://tl.ust.hk/integrity/student-1.html>

University's Penalties for Cheating: <http://tl.ust.hk/integrity/student-5.html>

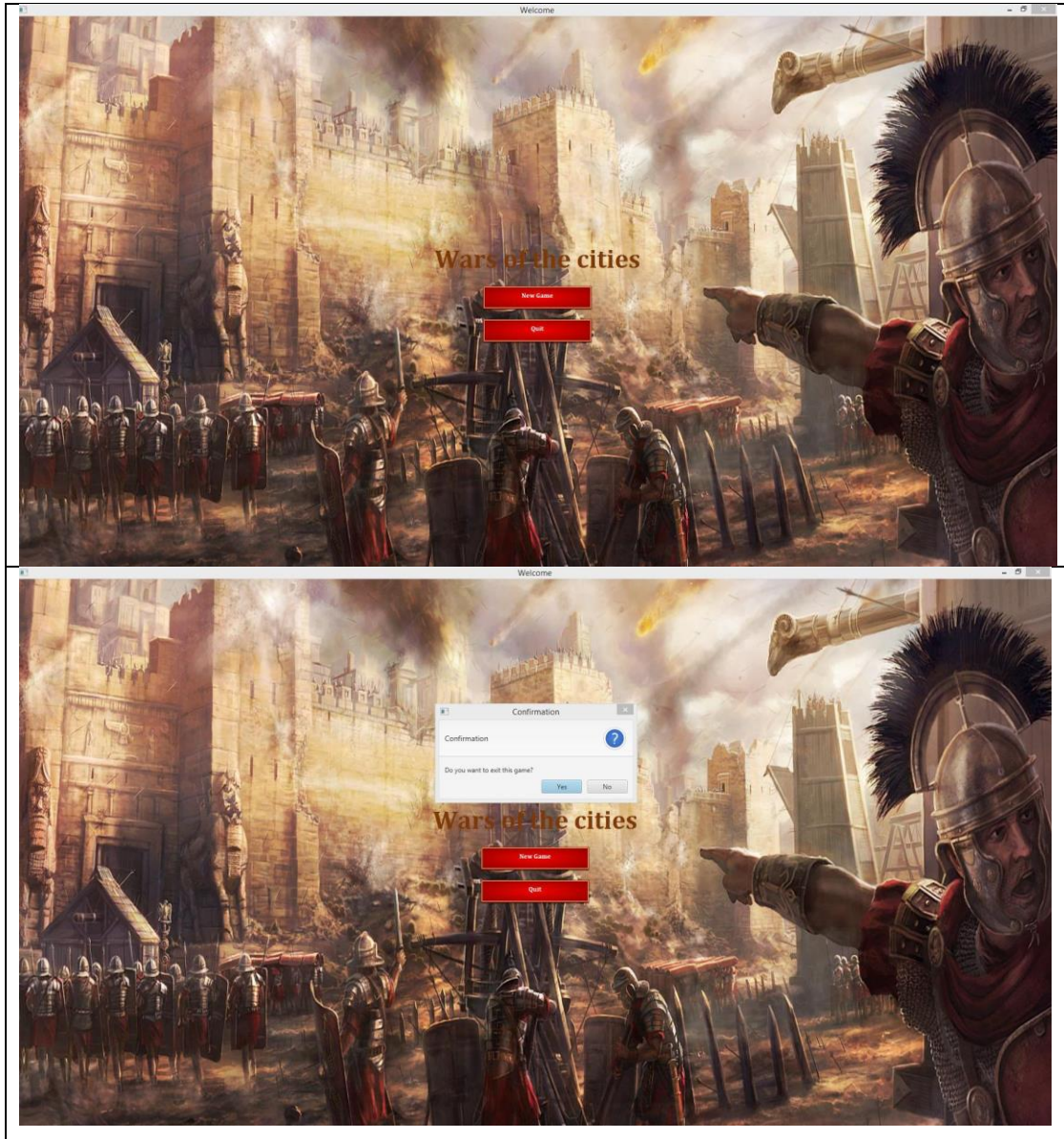


## Description

The PA2 is an extension of PA1. It is a turn-based strategy game involving up to three players. One of the players is a human player, while the others are computer players. The main extension you will make is to extend the game's text interface into graphical user interface, using JavaFX and event handlers. The PA2 will need you to apply the knowledge you have learnt in lab9 (Javafx), lab10 (event-driven), and lab11 (thread programming).

## GUI Introduction

### (1) The Welcome screen

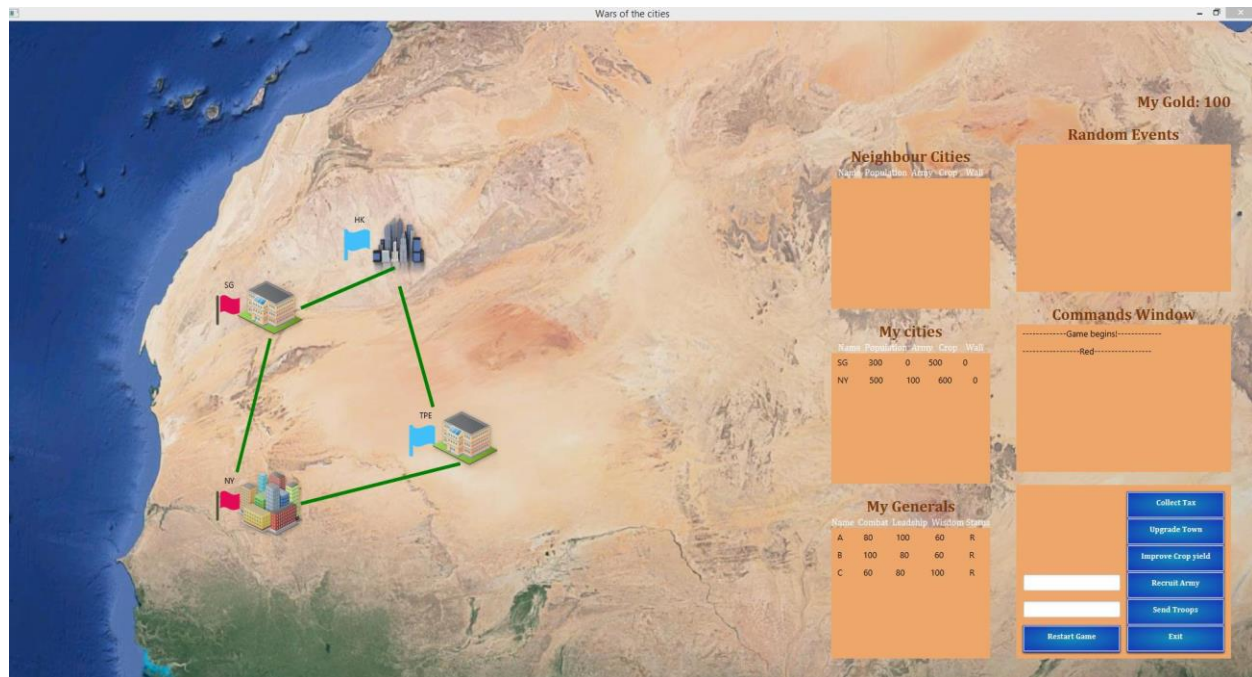


The above picture shows the start screen of the game. There are two buttons in the interface:

**New Game:** When the New Game button is clicked, another GUI for game star will be shown, and the game will be started.

**Quit:** When the Quit button is clicked, there will be a dialog box for the player to decide whether to exit the game.

## (2) The Player Interface



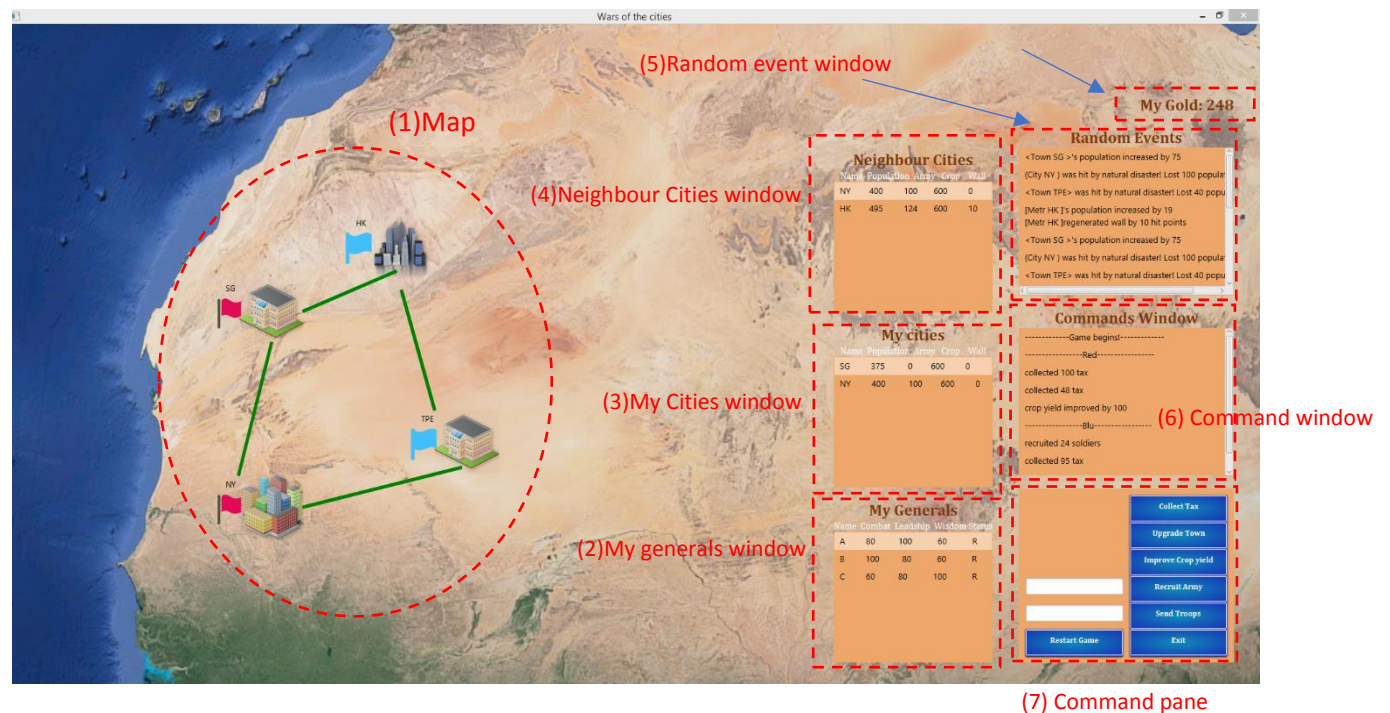
The player interface is being shown above.

- There are several computer players and one human play in the game. We assume the human player is always the first player in the game object (i.e. `game.players.get(0)` will return the human player). We also assume that including the human player, there could have a maximum of 3 players.
- To play the game, the human player has to select a ready general from the “listViewGeneral” (which is a `ListView<String>` object that contains the list of all the generals and their attributes), and then the human player needs to select a town/city/metropolis from the “listViewCity” (which is a `ListView<String>` object that contains the list of cities). The player can only select a ready general (with a “R” in the Status). After executing any of the following commands, the status of the general will be changed to “D” (“D” refers to “Done” of PA1).
- Then the human player can select a command by clicking an appropriate blue button at the right.
  - If the player wishes to collect tax, he could press the “Collect Tax” blue button and then the selected general will collect tax from the selected town/city/metropolis. “My Gold” at the upper right corner will be incremented according to the tax collected.
  - If the player wishes to upgrade a town/city, he/she could press the “Upgrade Town” blue button, and the just like in PA1, the town/city will be upgraded and 50 gold will be used. If the selected city is already a metropolis, then no upgrade will be done and an error message will be displayed in the “Commands Window” at the right.
  - If the player wishes to improve crop yield, he/she could press the “Improve Crop yield” blue button and then the crop yield will be improved accordingly.

- If the player wishes to recruit army, he/she needs first to input the budget (gold) he/she is willing to spend on recruitment at the white textfield, and then presses the “Recruit Army” button.
- If the player wishes to send troop to another town/city/metropolis either to transfer his/her troop among his/her own towns, or to attack another town/city/metropolis, he/she will need to enter the troop size at the white textfield and then selects the “Send Troops” blue button. The troop size could be sent must be no bigger than the “Army” size of the city being shown under “My cities” listview. If the player is attacking another city/town/metropolis, he/she should further select a victim town/city/metropolis from the “Neighbor cities” listview at the top left part of the command interface.
- The computer players will not execute any commands before the human player has finished all the commands.
  - The computer player will select the general, town/city/metropolis and command randomly. You can design some strategies for the computer player so that it can be smart. Design strategies for computer player will be the bonus for the PA2.
  - The action performed by the computer player could be viewed from “Commands Window”.
- After all the players have completed their commands, some random events will happen for each town. These events are shown at the “Random Event” window.
- Then the generals will become “Ready” and the human player can begin another round.
- Game will be over when one of the players owns all the town/city/metropolis.



### (3) The structure of the player interface

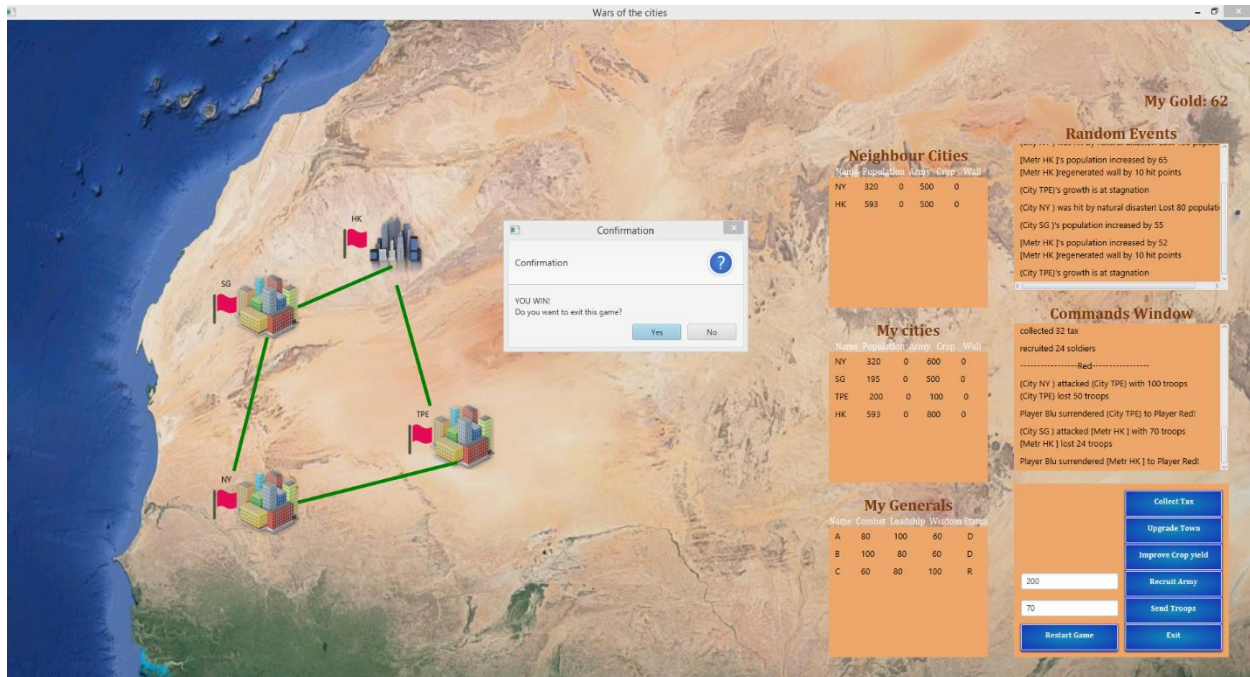


The player interface consists of the following parts:

- (1) **Map:** The map shows the location of the town/city/metropolis, information of their owners and the connection between towns/cities/metropolises.
- (2) **“My generals” window:** shows the information of the generals for the human player.
- (3) **“My Cities” window:** shows the information of the cities of the human player.
- (4) **“Neighbour Cities” window:** shows the information of the neighbor cities of the selected city in the “My cities” window.
- (5) **“Random Events” window:** shows the results of the random events for each town.
- (6) **“Command Window”:** shows the results of the commands from the players
  - a. **Collect Tax:** Execute the command to collect tax from a selected city by a selected general.
  - b. **Upgrade Town:** Execute the command to upgrade a selected town by a selected general.
  - c. **Improve Crop yield:** Execute the command to improve crop yield for a selected town by a selected general.
  - d. **Recruit Army:** Before executing the command of recruiting army, the human player has to fill in the budget (gold) in the textfield box.
  - e. **Send Troop:** Before executing the command of sending troop, the human player has to fill in the troop size in the input box and then select the general to be responsible for sending the troop, and then a city he/she owns, and then the neighbouring city the player wishes to send the troop to. If the selected neighbouring city belongs to the player, then this will be a transfer to troop from one town to another. If the selected neighbouring city belongs to another player (i.e. computer player), then this will be an military attack.
- (7) **Command pane:** It consists of several command buttons.

(8) **Gold account:** shows the amount of gold that the human player has.

When the game is over, a dialog box will show a dialog box indicating either you have lost the game or you have won the game, as shown in the figure below.



#### (4) The coordinate System

Different from PA1, we provide a pair of (x,y) coordinates for each of the cities. The coordinates are stored in the "PlayersData.txt", and are the last two numerical items of a city:

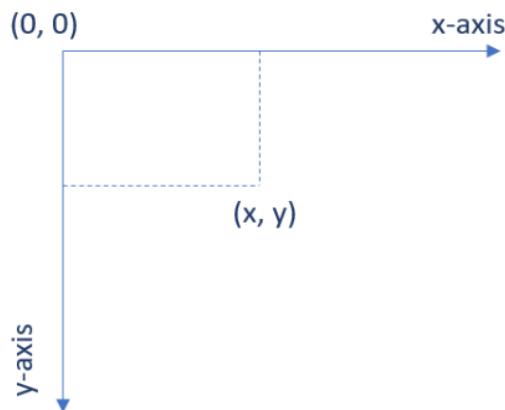
```
0 SG Town 300 0 500 200 200
```

For example the above row in the "PlayersData.txt" file describes a "Town" with index 0, named "SG", and has a population of "300", an army/troop size of "0" and a crop yield of "600" is located at the coordinates (200,200) on the map canvas

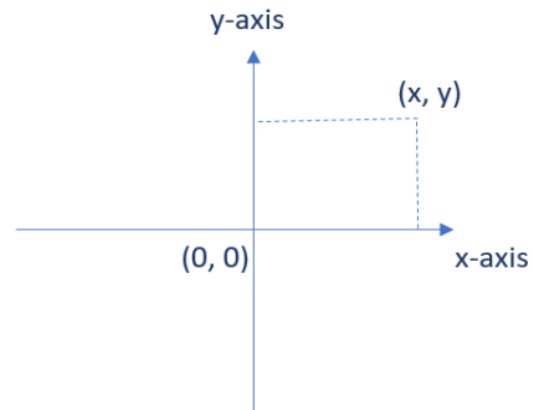
**You will need the concept of the Java Coordinate System described on the next page for PA2.**

# Java Coordinate System

- Java coordinate system is different the conventional coordinate system
  - ▶ The origin  $(0, 0)$  is at the top-left corner
  - ▶ x-coordinate value increases towards the right
  - ▶ y-coordinate value increases downwards



Java Coordinate System



Conventional Coordinate System

## Tasks

You will have to finish the following TODOs in the class `GameApplication` in the skeleton code downloaded. In the following we are providing you hints for implementing the methods, but it is just showing you briefly the crucial steps; the hints are not necessarily complete. To implement the methods correctly, you may need to make judgements based on knowledge you have learnt from Comp3021 lectures/labs (and also your common sense). Moreover the hints just show one of the ways to implement the game. You may need to refer carefully to the executable Jar and the descriptions in this document for the details of this game. You can implement the game in your own way and you are encouraged to add more functions to the game if you wish, as long as the game works exactly the same as we have expected. We hope you will enjoy the game.

```

private Pane paneWelcome()
{
    /**
     * To do:
     * 1. This function is to create the welcome scene
     * 2. There are two buttons and a label in this scene
     * Hint:
     * 1. To set the background of the pane, you can use
     *    welcomePane.getStyleClass().add("pane-welcome")
     * 2. To set the style of the button/label, you can use
     *    lbMenuTitle.getStyleClass().add("menu-title");
     *    btNewGame.getStyleClass().add("large-button");
     *    where lbMenuTitle is a Label and btNewGame is a button.
     */
}

private Pane paneStartGame()
{
    /**
     * To do:
     * 1. This function is to create the game start scene.
     * 2. There are two parts in this scene:
     * 2.1 the canvas where you can draw the map
     * 2.2 the control part that consists of four listview, some
     *     buttons and two TextField
     * 3. Please think carefully on how to put the button/listview
     *     /label in the right places
     * Hint: To set the background of the control pane, you can use
     *    pane.getStyleClass().add("pane-game-start");
     */
}

private void handleExitGame()
{
    /**
     * Todo:
     * 1. The handler for the Button exit and quit
     * 2. When the button exit or quit is clicked, this function will
     *    be called.
     * 3. When the function is called, there will be a dialog box
     *    showing
     *        "Do you want to exit this game?"
     * 4. If the user choose yes, then you can call Platform.exit();
     *
     * Hint: user Alert:
     * Alert alert = new Alert(AlertType.CONFIRMATION, "Do you want
to exit this game?", ButtonType.YES, ButtonType.NO);
     */
}

```



```

private void loadTownMap()
{
    /**
    * TODO:
    * 1. Draw the town/city/Metropolis in the canvas according to
    *    their coordinate
    * 2. Draw the flags next to the town/city/Metropolis's owners.
    *    For the 1st player, use the "flag_0.png", for the 2nd
    *    player, use the "flag_1.png"
    * 3. Draw the line between the connected town/city/Metropolis
    *    according to the data in the "MapData"
    *    You can use the drawPath(int x1,int y1,int x2,int y2) to
    *    draw the line between two points (x1,y1) and (x2,y2).
    */
}

private void handleNewGame()
{
    /**Todo: the handler for the button "New Game"
    *         when the button "New Game" is clicked, this
    *         function will be called
    *
    * Hints:
    *     You may assume there is only one human player, and the
    *     human player could be retrieved by game.players.get(0);
    * 1. load the map information from the file "MapData.txt":
    *     game.gameMap.loadGameMap("MapData.txt");
    * 2. load the player information from the file
    *     "PlayersData.txt":
    *     game.loadPlayersData("PlayersData.txt");
    * 3. call the loadTownMap() to plot the Map;
    * 4. initialize the humanPlayer and the computerPlayers:
    *     the first user in the "PlayersData.txt" is the humanPlayer;
    * 5. change the scene: putSceneOnStage(SCENE_STARTGAME);
    * 6. call the startTurn to start the Game: startTurn()
    */
}

private void updateListViewCityItems()
{
    /**
    * TODO:
    * To update the listview of the information of the human
    * player's cities.
    * 1. It is similar to the function updateListViewNeiItems()
    * 2. Use listViewCityItems.add() to add the information of

```

```

        *   cities
        */

    }

    private void updateListViewGeneralItems() {
        /**
         *   TODO:
         *   To update the listview of the information of the human
         *   player's generals.
         *   1. It is similar to the function updateListViewNeiItems()
         *   2. Use listViewGeneralItems.add() to add the information of
         *       generals.
         */
    }

    private void initWelcomeSceneHandler()
    {
        /**
         *   TODO:
         *   To initialize the handler for the two buttons in the welcome
         *   scene
         *   1. handleNewGame() for button btNewGame
         *   2. handleExitGame() for button btQuit
         */
    }

    private void initStartGameSceneHandler()
    {
        /**
         *   Todo:
         *   To initialize the handle for all listView and button in the
         *   start game pane.
         *   1. call the initListView(); to initialize the listview
         *   2. call setOnAction for each button
         */
    }
}

```

```

private void showGameOver()
{
    /**
     * Todo:
     * TO Show Game Over using Dialog box.
     * When the game is over, this function will be called.
     * Hint:
     * 1. Use another thread to handle it. You may use
     * Platform.runLater in this function.
     * 2. Show "YOU LOSE!" if the human player loses the game,
     * otherwise "YOU WIN!"
     * 3. You can use
     *Alert alert = new Alert(AlertType.CONFIRMATION, result,
ButtonType.YES, ButtonType.NO);
     *         where result is the String to be presented
     */
}

private void handleRestartGame() {
    /**
     * Todo:
     * The handle for the button of restart game.
     * When the button "Restart Game" is clicked, this function will
     * be called.
     * Hints:
     * 1. Clear all the players
     * 2. call game.gameMap.loadGameMap("MapData.txt");
     *         game.loadPlayersData("PlayersData.txt");
     *         loadTownMap();
     * 3. Update all the listview
     * 4. call the startTurn()
     */
}

private void startTurn() {
    /**
     * TODO:
     * 1. This function is to start the thread playerThread.
     * 2. the function processPlayerTurns() will be run in the
     *     playerThread.
     */
}

public void processPlayerTurns()
{
    /**
     * TODO:
     * 1. This function is for processing the commands of computer
     *     players.
     * 2. All the computer players have to wait until all the

```

```

*     generals of human player are done.
* 3. When the game over or when the human player chooses to
*     restart the game, the function will be terminated.
* 4. After all the computer players finish their commands, all
*     the generals of human players will be ready.
* 5. Each computer player in each round will randomly select a
*     general, a town/city/metropolis and a command.
* 6. The same as the PA1, there are five command"
*     (1)collect tax;(2)upgrade town;(3)improve crop yield;
*     (4)recruit; (5)send troops
* 7. All the output message will be added in the arraylist
*     printResult and print in the log window.
* 8. When the game is over, there will be a dialog box to print
*     the result and ask whether to exit the game.
*     To achieve this goal, you can call showGameOver();
* 9. When the button "Restart" is clicked, this function will be
*     over. The variable gameOver can be used here.
*/
}

```

```

private void handleUpgrade() {
    /**
    * Todo:
    * When the button upgrade is clicked, this function will be
    * called.
    * 1. call selectGeneral() to obtain the selected general
    * 2. call selectTown() to obtain the selected town
    * 3. call upgradeTown(selectedTown, selectedGeneral) to upgrade
    * 4. if upgrade successfully, call loadTownMap() to redraw the
    *     map
    * 5. update all the listview and myGold
    */
}

```

```

private void handleRecruit() {
    /**
    * Todo:
    * When the button recruit is clicked, this function will be
    * executed.
    * 1. Obtain the input budget from the TextField using
    *     tfRecruit.getText();
    * 2. call selectGeneral() and selectTown() to obtain the
    *     selected general and town
    * 3. call selectedTown.recruitArmy(humanPlayer, selectedGeneral,
    *     budget) to execute the command
    * 4. Update all the listview and myGold
    */
}

```



```

private void handleCollectTax() {
    /**
     * Todo:
     * When the button collect tax is clicked, this function is
     * called.
     * 1. selectGeneral() for obtaining the selected general
     * 2. selectTown() for obtaining the selected town
     * 3. call selectedTown.collectTax(humanPlayer, selectedGeneral)
     *    to execute the command
     * 4. update all the listview and myGold:
     *     updateListViewCityItems();
     *     updateListViewGeneralItems();
     *     updateListViewMessageItems();
     *     updateMyGold();
     */
}

private void handleImproveCrop() {
    /**
     * Todo:
     * When the button improve crop is clicked, this function will be
     * called.
     * Hints:
     * 1. selectGeneral() for obtaining the selected general
     * 2. selectTown() for obtaining the selected town
     * 3. call selectedTown.improveCropYield(selectedGeneral) to
     *    execute the command
     * 4. update all the listview and myGold:
     *     updateListViewCityItems();
     *     updateListViewGeneralItems();
     *     updateListViewMessageItems();
     *     updateMyGold();
     */
}

```

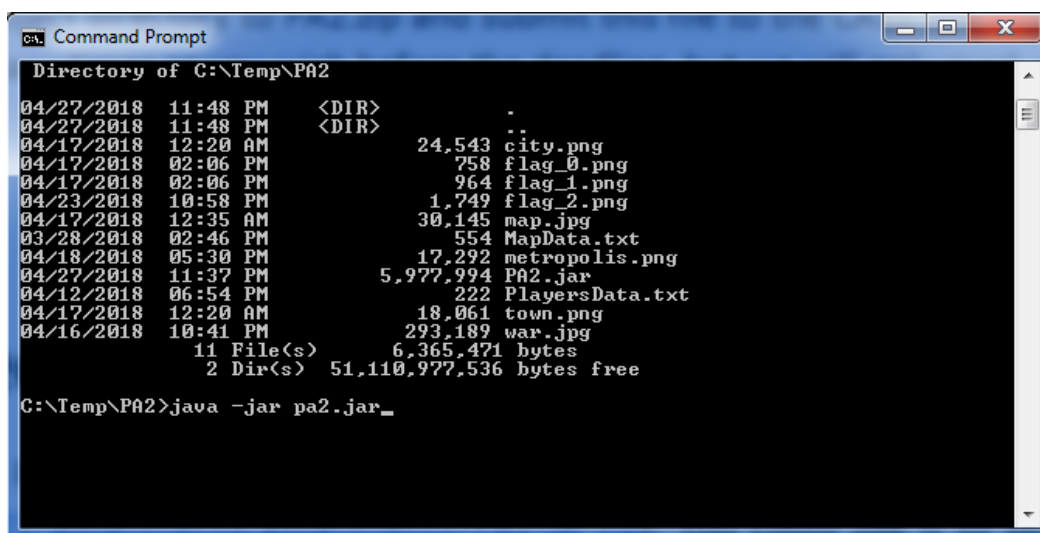
# Submission

Congratulations! You are done with the strategy game. Hope you have enjoyed implementing it! Also hope this game has provided you a fun and painless way to learn Javafx UI, event-driven programming and Java threads! You are very likely to use them in your future! To submit your PA, you can zip your complete project directory, and then name it "PA2.zip", and submit this file to the CASS link shown on page 1. You can submit for as many times as you wish before the deadline, but we will only mark you latest submission.

## Running the executable jar file

An executable jar file for the PA could be downloaded from the link at the course web (<https://course.cse.ust.hk/comp3021/assignments/assignment2/PA2-jar.zip>). You could use it to verify the correctness of your finished PA. To run the executable jar file under Windows (sorry again, we only have Windows machine to compile it), unzip the zipped file into a single directory. Now make sure the two .txt files (MapData.txt and PlayersData.txt files) and the 3 flag pictures (flag\_0.png, flag\_1.png, flag2.png files), the map.jpg, the town pictures (town.png, city.png, metropolis.png), the war.jpg, the map.jpg, and PA2.jar files are all in the same directory. Open a Windows command window, and issue the command:

```
java -jar pa2.jar
```



```
Command Prompt
Directory of C:\Temp\PA2
04/27/2018 11:48 PM <DIR> .
04/27/2018 11:48 PM <DIR> ..
04/17/2018 12:20 AM      24,543 city.png
04/17/2018 02:06 PM       758 flag_0.png
04/17/2018 02:06 PM       964 flag_1.png
04/23/2018 10:58 PM      1,749 flag_2.png
04/17/2018 12:35 AM      30,145 map.jpg
03/28/2018 02:46 PM       554 MapData.txt
04/18/2018 05:30 PM      17,292 metropolis.png
04/27/2018 11:37 PM    5,977,994 PA2.jar
04/12/2018 06:54 PM       222 PlayersData.txt
04/17/2018 12:20 AM      18,061 town.png
04/16/2018 10:41 PM      293,189 war.jpg
               11 File(s)      6,365,471 bytes
               2 Dir(s)  51,110,977,536 bytes free

C:\Temp\PA2>java -jar pa2.jar_
```

## **Bonus: +6 points**

As mentioned in the Game Process Introduction, the actions of the computer players are given randomly. To improve their performance, you can design some strategies for the computer players. Please attach a report in your submission to brief introduce your implement of the strategies for computer players, please be precise in the report. We will not be able to consider your bonus work if you do not attach a report.