ALGONQUIN COLLEGE

ASSIGNMENT 1.2 – GUI for Game (NumPuz[1])

General View

Due Date: prior or on **Oct 2nd 2022 (midnight)**

- **2nd Due date** (until Oct 9th) - 50% off.

Earnings: 12% of your course grade.

Purpose: Create the interface and basic organization of the Game NumPuz.

- This is the second task in JAP. Now, you are moving to the implementation of the **NumPuz game** application. Remember that this project will be updated to incorporate new functionalities.

Note 1: About Teams
As mentioned, new teams are acceptable up to the end of for assignment. It means that after this period, no new team can be composed. Students from different sessions can be accepted, since there is only one lab / lecture professor.

- **PART I:** What you need to create the interface considering the model:
  - A basic controller (to be increased in the next assignments).
  - The main window for the game (NumPuz).
- **PART II:** Additional elements are required to define:
  - **Code** (using code conventions).
  - **Javadoc** files (created by documental comments in the code).
  - **Executable** Jar (remember to be sure that it is not a common archive file).

Note 2: About scripts
Students must create **scripts** to perform all activities. See, for example, the script provided by your professor in the labs and/or the following example (using local variables to be adjusted to your own configuration).

[1] Check the A11 specification for more details.

Example:

```
:: ------------------------------------------------------------------------
:: JAP COURSE - SCRIPT
:: ASSIGNMENTS - CST8221 - Fall 2022
:: ------------------------------------------------------------------------
:: Begin of Script (Assignments - F22)
:: ------------------------------------------------------------------------
CLS
:: LOCAL VARIABLES ..................................................
SET JAVAFXDIR=/SOFT/copy/dev/java/javafx/lib
SET SRCDIR=src
SET BINDIR=bin
SET BINOUT=game-javac.out
SET BINERR=game-javac.err
SET JARNAME=Game.jar
SET JAROUT=game-jar.out
SET JARERR=game-jar.err
SET DOCDIR=doc
SET DOCPACK=game
SET DOCOUT=game-javadoc.out
SET DOCERR=game-javadoc.err
SET MAINCLASSSRC=src/game/GameBasic.java
SET MAINCLASSBIN=game.GameBasic
SET MODULELIST=javafx.controls,javafx.fxml
@echo off
ECHO " _____________________________________ "
ECHO "|      __    _  ___    ___  _          |"
ECHO "|     |  |  / \ \  \  /  / / \         |"
ECHO "|     |  | /   \ \  \/  / /   \        |"
ECHO "|     |  |/  _  \ \    / /  _  \       |"
ECHO "|  __ |  |  / \  \ \  / /  / \  \      |"
ECHO "|  \____/_/   \__\ \/ /__/   \__\ |"
ECHO "|                                     |"
ECHO "| .. ALGONQUIN COLLEGE - 2022F .. |"
ECHO "|_____________________________________|"
ECHO "                                       "
ECHO "[ASSIGNMENT SCRIPT ----------]"

ECHO "1. Compiling ....................."
javac -Xlint -cp ".;%SRCDIR%;%JAVAFXDIR%/*" %MAINCLASSSRC% -d %BINDIR% > %BINOUT% 2> %BINERR%

ECHO "2. Creating Jar ................."
cd bin
jar cvfe %JARNAME% %MAINCLASSBIN% . > %JAROUT% 2> %JARERR%

ECHO "3. Creating Javadoc ............."
cd ..
javadoc -cp ".;%BINDIR%;%JAVAFXDIR%/*" --module-path "%JAVAFXDIR%" --add-modules %MODULELIST% -d
%DOCDIR% -sourcepath %SRCDIR% -subpackages %DOCPACK% > %DOCOUT% 2> %DOCERR%

cd bin
ECHO "4. Running Jar ................."
start java --module-path "%JAVAFXDIR%" --add-modules %MODULELIST% -jar %JARNAME%
cd ..

ECHO "[END OF SCRIPT -----------------]"
ECHO "                                       "
@echo on

:: ------------------------------------------------------------------------
:: End of Script (Assignments - F22)
:: ------------------------------------------------------------------------
```

Part I – Creating the Game

1.1. BASIC GAME

The **NumPuz** is a simple visual game in which you need to be able to fulfill tiles by using sequence of digits (or any different pattern[2]).

Some references for this game can be found easily on the internet. For instance, you can play an interesting version using internet resources (ex: https://play.google.com/store/apps/details?id=com.dopuz.klotski.riddle).

- In short, we use square boards with N=dim^2. For example, if we have dim=3, our board will have 9 tiles, and at least one of them is empty and used for movements. So, you can see random distribution from numbers from 1 to N-1.
- The game must have two modes: "design" and "play", where you can respectively, create and starting playing the game.
  - In the design mode, the initial board is created with basic components.
    - The board can be completely "blank";
    - Or can show the initial solution;
    - Or even a random value created and ready to be played).
    - ***Important**: No events are really necessary to be created now.*

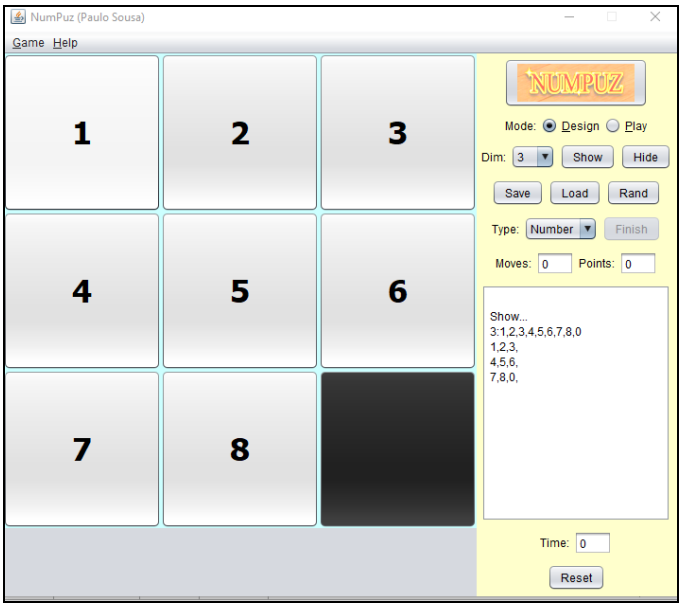

***Fig. 2** – Proposal for NumPuz interface*

  - In the play mode (to be implemented in the game), ideas about the logic and movements are supposed to be created.

[2] NumPuz means "Number Puzzle". However, the game itself can be played using different orders/sets.

1.3. GAME CONTROLLER

The game itself can be defined by one basic top-level container (ex: JFrame) and one single class – basically a kind of *simple controller*, showing the game interface. The basic functionalities required right must include:

- Define the dimension (at least from 3 to 5);
- Define the type of game ("Number" or "Text").

Class GameController

The class GameController will assemble the UI, primarily in its default constructor. Here are some recommendations (you can consider these ideas as "*suggestions*"):

- You can consider it as an **inner class** called **Controller**.
  - The **Controller** class will have very basic code. It reads the action command (or JavaFX equivalent) and will develop the **NumPuz** in the two different modes above described ("design" or "play").
    - Once one mode is selected, a logic function will be invoked to check if it is ok and will let it be included in the configuration. Some basic behaviors:
- All components can have a unique action command, but the way that you can select the mode as well as the strategy to be used in the game can vary freely.
  - For example: You can use menus or choices or dialogs to define the mode initially. The **NumPuz** itself can be composed by buttons, or labels, or text fields.
- Some notes:
  - **NOTE 1**:
    - If you are using Swing, **GameController** can extend JFrame. You may have to define your JFrame parameters in GameController instead of the **NumPuz** class.
    - If you are using JavaFX, **GameController** may extend Scene, Pane or any descended Pane subclass as you see fit.

1.4. BASIC GAME INTERFACE

1.4.1. General Interface

The **Game** must let you perform some actions:

- **Select the mode**: "Design" or "Play".

- **Define the dimension**: This value can vary according to one specific range (ex: 3 to 5). Remember, that it means that the interface must be dynamically configured.

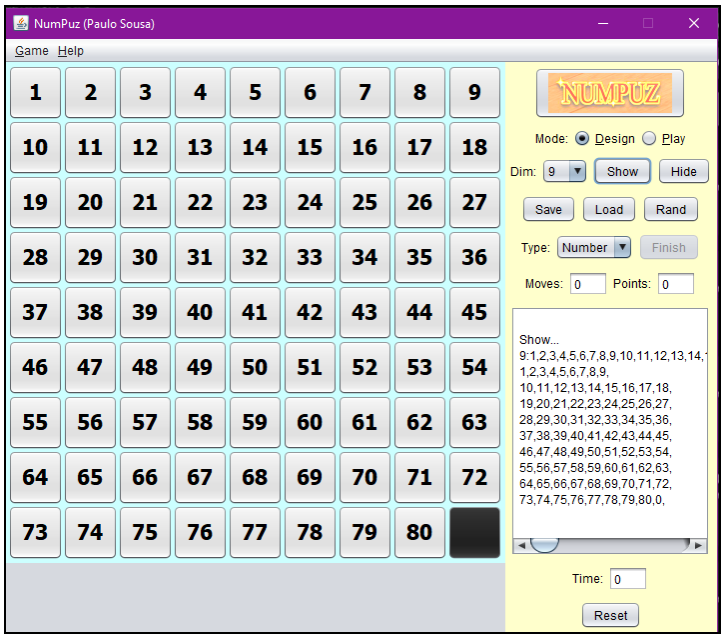

Fig. 2 – Changing the NumPuz dimension (in this case, 9x9).

- **Define the type**: You can select basically:
  - **Numerical values**: Use a sequence from 1 to N (=dim^2-1);
  - **Text input**: You can define the game by using an input string (and each cell will contain one char).

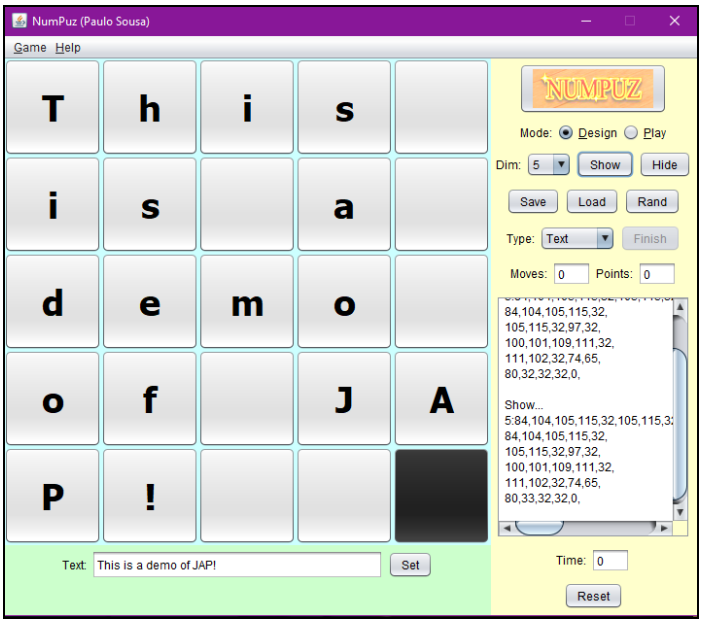

Fig. 3 – Using text for NumPuz

- **Additional elements**: Your interface must also contain:
  - **Visual Elements:**
    - **Image logo**: Small button with image that can open a dialog with "About" message.

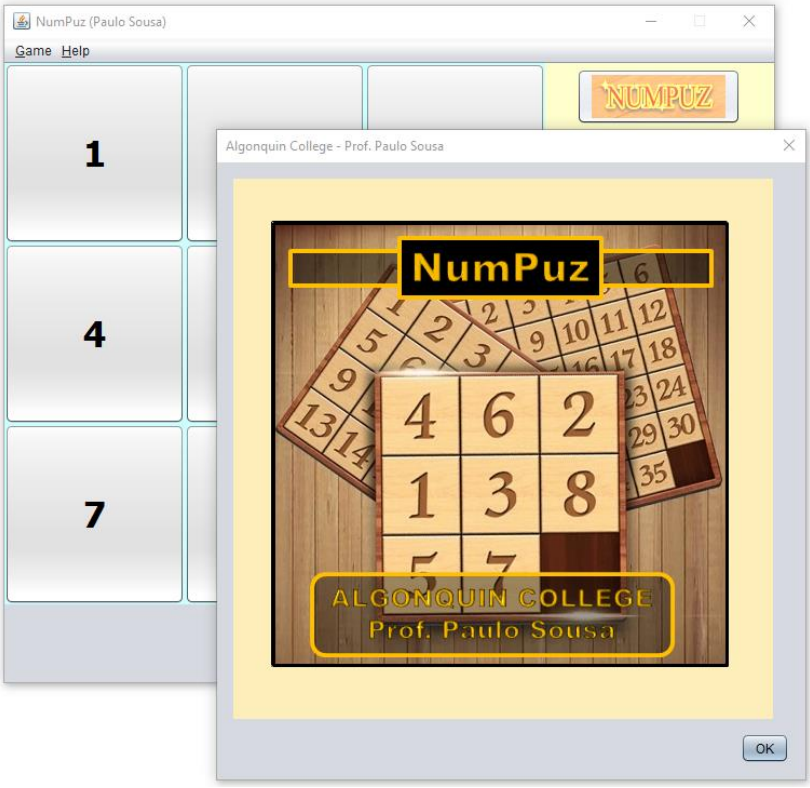

Fig. 4 – Using Dialogs with images ("About")

  - **Execution info (only to be shown[3]):**
    - **Points**: Showing the points at this moment.
    - **Movements**: Displaying the amount of movements done by the user.
    - **Time**: To show the number of seconds since the beginning of game (to be implemented later).
  - **Additional elements (optional):**
    - **History Area**: To describe the sequence of actions performed by the user.
    - **Reset button**: To restart the game.

[3] It means that it is not necessary to implement actions for these elements (required only for the assignment A21/A22).

1.4.2. Initial Behavior

Once selected the Design mode, the user must be able to define a correct configuration for **NumPuz**.

- For while, the only actions are related to prepare the game to be played (with the configuration in the design mode).
- Remember, user can select **dimension** and the **type of input** (numerical or text).
- Features to be further implemented include the functionalities while playing and some additional elements, for ex:
  - *Color* management;
  - Input / output (ex: *load* and *save* properties).

1.4.3. GUI Recomendations

TIPs: The definition for all colors and sizes are free (not predefined here), since the general layout is respected.

- This layout uses Java's default look and feel.
  - It is better to consider that the top-level container (frame / stage) are not resizable.
- You may use BorderLayout, FlowLayout and GridLayout (or the JavaFX equivalent).
  - ***Remember**: The use of UI builders is not forbidden but it is definitively not recommended!*
- **About code:** Code conventions are really recommended.

Part II – Documentation and Packing

2.1. BASIC DOCUMENTATION

CODE ELEMENTS (*Basic criteria*)

- Code well organizes, methods and functions grouped logically, classes in order, etc.
- Label naming (variables, constants, methods, functions, etc.) consistent and meaningful.
- Code consistently indented and spaced (define a consistency /organization).
- Good commenting which includes block comments.

JAR (*Executable file*)

- It is required to create and include JAR files (that can open the binary using Java)

JAVADOC (*following Java Standard*)

- Finally, **javadoc** must be generated (you do not need to submit).

Evaluation

- Please read the Assignment **Submission Standard and Marking Guide**.
- **About Plagiarism**: Your code must observe the configuration required. Similarly, we need to observe the policy against ethic conduct, avoiding problems with the 3-strike policy...

Note 3: About Standards
The code and patterns to be used in submission must follow the Java Code Conventions, but also the elements described in the Assignment Standard.

Submission Details

- **OPTION 1: Digital Submission: Compress** into a **zip** file with the complete code (.java, images, .JAR, and **scripts**) that you created specifying the language and include as attachment in the *BrightSpace*.
  - ***IMPORTANT NOTE:*** The name of the file must be **Your Last Name** followed by the last three digits of your student number followed by your lab **section number**. For example: **A11_Sousa123.zip**.
    - If you are working in teams, please, include also your partner. For instance, something like: **A11_Sousa123_Melo456.zip**.
- **OPTION 2: GitHub Submission:** In the *BrightSpace*, include just the repository links (giving access to the professor if it is not public).

Note 4: About GitHub
This term, we are introducing the use of GitHub as a bonus activity (see the additional marks in the Rubric). So, you can simply submit a minimal instruction (ex: README.TXT) describing the access to your repository, as well as the hash code for your version.

- ***IMPORTANT NOTE:*** **Assignments will not be marked if there are not source files in the digital submission / GitHub repository.** Assignments could be late, but the lateness will affect negatively your mark: see the Course Outline and the Marking Guide. All assignments must be successfully completed to receive credit for the course, even if the assignments are late.

Marking Rubric

Maximum Deduction (%)	Deduction Event
	Severe Errors:
12 pt (100%)	Late submission (after 1 week due date)
12 pt (100%)	Plagiarism detection (code from previous versions)
6 pt (50%)	Does not compile with javac
6 pt (50%)	Compile but crashes after launch
PATTERN	**Non-compliance**
6 pt (50%)	The GUI appearance is not compliant with game functionality.
6 pt (50%)	Execution problems (check rule 1 - *UX*[4])
6 pt (50%)	Inadequate / inappropriate functionalities (check rule 2 - *KISS*[5])
3pt (25%)	Missing execution scripts
0.1pt each	Documental errors[6]
0.1pt each	Warnings[7]
0.5	Missing component (various text fields, buttons, labels, graphics)
0.5	Other minor errors
1 pt	GitHub
1 pt	Clean and elegant code, enhancements, discretionary points.
Final Mark	**Formula:** 12*((100- ∑ penalties + bonus)/100), max score 14%.

File update: Sep 18th 2022.

Good luck with A12!

[4] Rule 1 (UX): "*Put your self in the place of the user*".
[5] Rule 2 (KISS): "*Keep It Simple, Student*".
[6] During documentation by using the **scripts**.
[7] During compilation (ex: deprecated methods) by using the **scripts**.