Java[IOS71a] Assignment

crossword puzzle design report

Jiangli Gui (r0867437)

1. Overview of the project and functionality of an individual class

The picture in the appendix (please read it first if necessary to have an overview of the whole structure) demonstrates the overview structure of this project. 3 PACKAGES are containing several CLASSES to implement the puzzle game.

① . **Package1** named "inputfile" contains the main entry of this project and also includes property file "config.properties" and puzzle "puzzle-1-adjusted.txt" that supposed to be read.

- Main: At the start of the program, read the property file first to know the name of file that is supposed to be read next and get the solution of this puzzle. Then an instance of ReadFile and FileContent are created to get the content of the puzzle file. These content are separated and stored in variables. Finally, a layout is created with those variables that contain the information of the puzzle as well as the solution of this puzzle.
- ReadFile: read the file line by line and organize all the information together.
- FileContent: read the first line to get the dimension of the puzzle(row number and column number) and store them in variables. Read the next several lines based on the row number and store the whole "matrix" in the variable. Finally, read the rest of the file and store them in variables.

② **Package 2** named "layout" intends to construct the whole layout of the game.

- Layout: an instance of JFrame is created and hold an instance of JPanel, which put these instance of Instructions, Playfield, and Solution together in "BorderLayout". This class also receives variables that contain puzzle information that is passed from the Main.
- Instructions: after receiving instruction information on the puzzle, different fonts are applied to different content.
- Playfield: after receiving the dimensions of the puzzle as well as the construction of it, x*y squares are generated in different colors with different functions. There is also a method that recognizes the character in the brackets H(*), to create instances of BlueSquares and pass the true answer to these squares.
- Solution: create a panel to contain the instances of InputSolution and Check. The solution of this puzzle is transferred from String to String array(separate the whole word into a single character) and pass them into InputSolution and Check.
- InputSolution: based on the length of "solution array", corresponding numbers of buttons are created and inherit the GerySquares directly.
- Check: a class for "check" button to receive the array of input buttons and the array of solution. It also contains an instance of a CheckListener to check whether the player's answer is correct by popping up a dialog with a corresponding message.
- CheckListener: Compare the "input answer" and "solution", to show two different dialogs by creating an instance of Popup.
- Popup: Show different messages based on the correctness of the input answer.

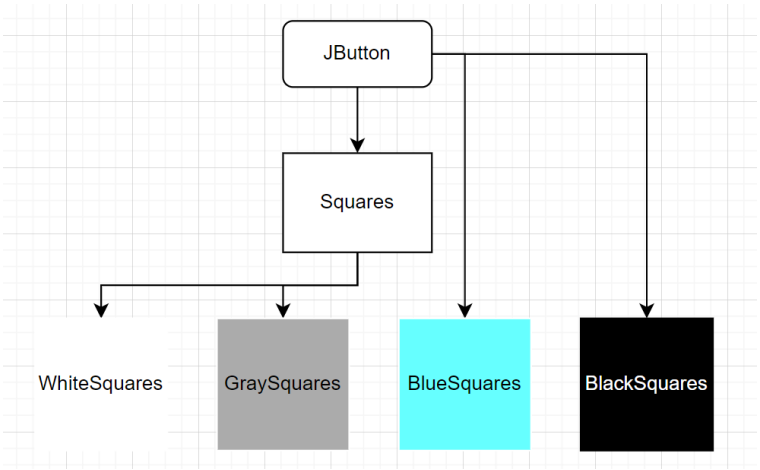
③ **Package 3** named "playfield" contains 4 different squares with different colors and functions.

- Squares: Set the basic function of an individual square: after clicking one specific square, an instance of KeyBoard will show up. The site of this specific square will be recorded at the same time.
- KeyBoard: When passing two arguments(JButton site, String title) inside, a dialog will show up at the center of the screen with 26 alphabet letters. Player can enter their answer by clicking the letter. An instance of ButtonListener was created and set on each character. When passing three arguments(JButton site, String title, String trueAnswer) inside, A dialog will show up at the center of the screen with 26 alphabet letters, while there only 5 random charters can be entered with a true answer of specific blue squares.
- ButtonListener: After clicking one of these letters, the owner (specific square) of this instance of keyboard will show the letter that was just clicked.
- WhiteSquares: Set the background color of the square to white, pop up an instance of KeyBoard after clicking it and change the text on the square when player enters a character.
- GraySquares: Set the background color of the square to gray, pop up an instance of KeyBoard after clicking it, and change the text on the square when player enters a character.
- BlueSquares: Set the background color of the square to blue, pop up an instance of BlueKeyBoard after clicking it, and change the text on the square when player enters a character.
- BlackSquares: Set the background color of the square to black with no other function.

2. Relationships

Inheritance relation:

Parent class	Child class
JFrame	Layout
JTextPane	Instructions
JPanel	Playfield, Solution, InputSolution
JDialog	Popu, KeyBoard
JButton	Check, Squares, BlueSquares, BlackSquares
Squares	WhiteSquares, GraySquares



Implement relation:

CheckListener and ButtonListener implement ActionListener interface.

3. Difficulties

a. Inheritance: (BlueSquare and BlackSquare)

In this project, there are 4 kinds of squares. I've tried to create a parent class (Squares) to construct the common properties for these four types for polymorphism and inheritance purposes. However, I didn't manage to do that for both blue square and black square, because in the non-parameter constructor of Squares, I've written codes to create a keyboard and add a listener to get the site of press square. In this case, due to the different functions of blue squares (to call a different version of keyboard) and black squares (no function at all). If BlueSquare and BlackSquare extend Squares, they'll call the non-parameter constructor of Squares, then, of course with the same function of white and gray squares. Therefore, BlueSquare and BlackSquare are set to extend JButton directly.

b. Keyboard(code reuse)

There are two versions of keyboard in this project. In the beginning, I've created two Class Keyboard and BlueKeyboard to achieve their function separately. Then I realized that since there are differences between these two keyboards when passing arguments in them, I compress these two classes into one class Keyboard. Although I found that there is a great part of code was duplicated in this class(create 26 alphabet buttons, add window focus listener, set the properties of keyboard panel, etc), I'm not able to reuse this code. I've tried to create a method in this class for purely creating a pop-up keyboard, but I have no idea how to add a listener to each (or 5-random) button inside.

4. Strengths and weaknesses

The Strengths of this project are :

a. expandability(as long as the same rules)

The code in ReadFile, Instructions, and Playfield parts are designed not only for this puzzle. As long as the puzzle file follow the same format(state dimensions of squares in the first line, followed by a matrix with O, H, S, and X that refer to a different type of squares, and finally the rest for the instruction). All uppercase words, serial number of cues, and cues that start with year will be set as blue color and bold font. No matter what character in the bracket after H, H(*), will be captured.

b. encapsulation, inheritance, polymorphism

All classes are well encapsulated by setting the characters as private as much as possible, and all setter and getter methods are included. Other than classes that extend the Swing, a good inherited relationship was constructed among different squares to fully achieve the code reuse. Polymorphism was used as much as possible when creating instances.

The weaknesses of this project are:

a. limited expandability of reading file content

The expandability of file reading and further decomposition is still not enough when a new formation of puzzle file is provided.

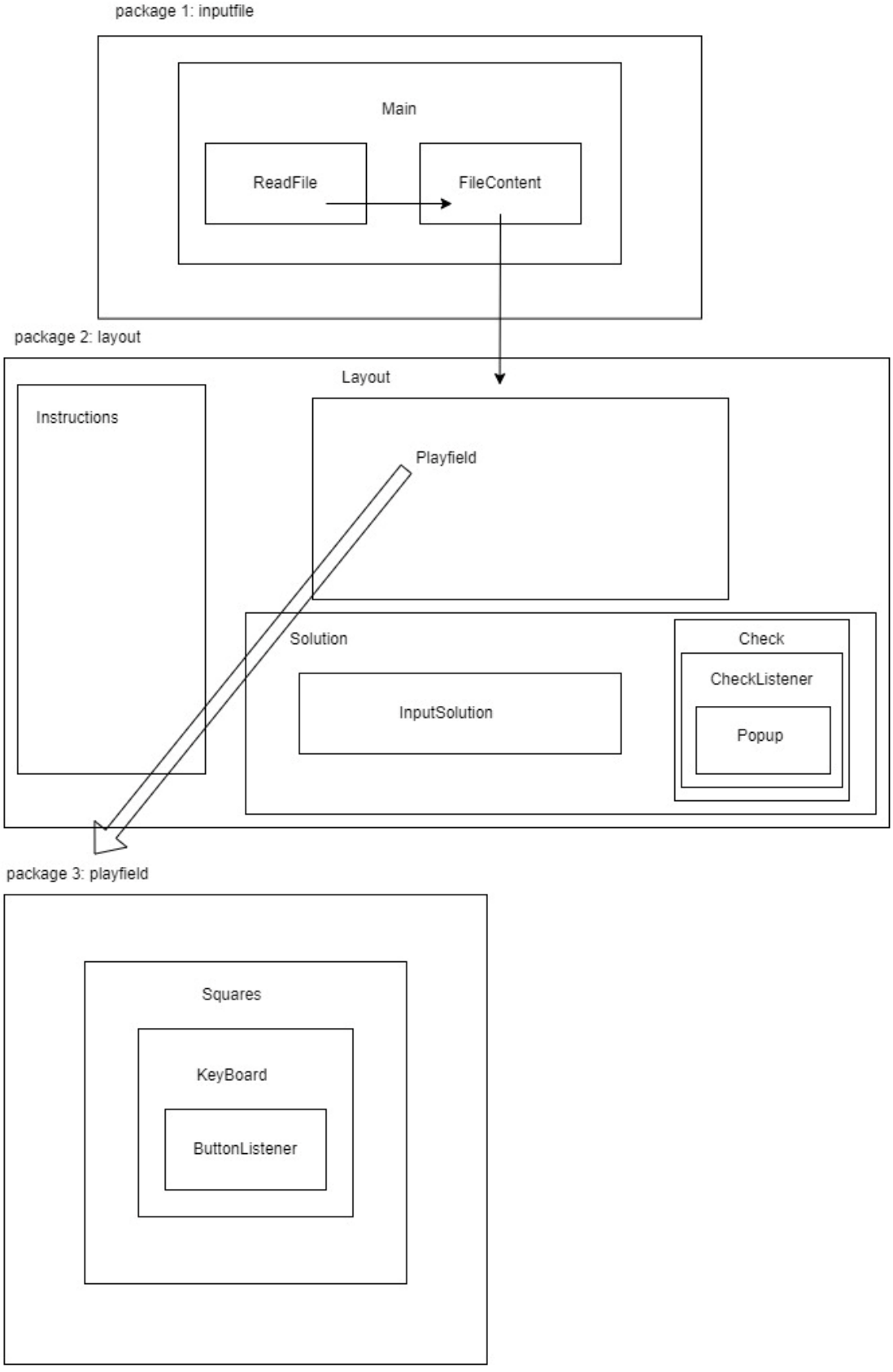
b. code reuse in keyboard

The codes in KeyBoard class are poorly designed, which leads to some duplicated parts.

c. inheritance of blue and black

It will be more ideal if all 4 kinds of squares can inherit Squares and be modified for the individual property.

Appendix


package 1: inputfile
Main
ReadFile
FileContent
package 2: layout
Layout
Instructions
Playfield
Solution
InputSolution
Check
CheckListener
Popup
package 3: playfield
Squares
KeyBoard
ButtonListener