Programming Logic & Techniques (PLT)

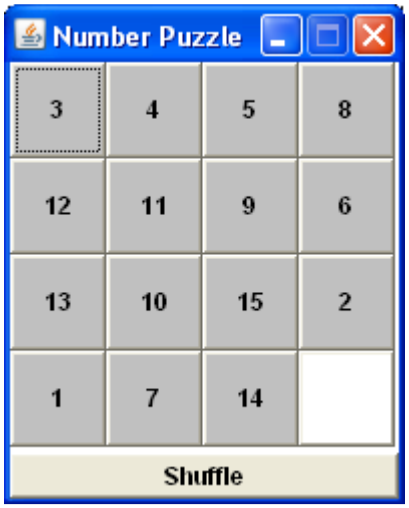
Assessment ID: SAAF-AS-I0-1 Iteration #: 0

Name: _____ EMP ID: _____

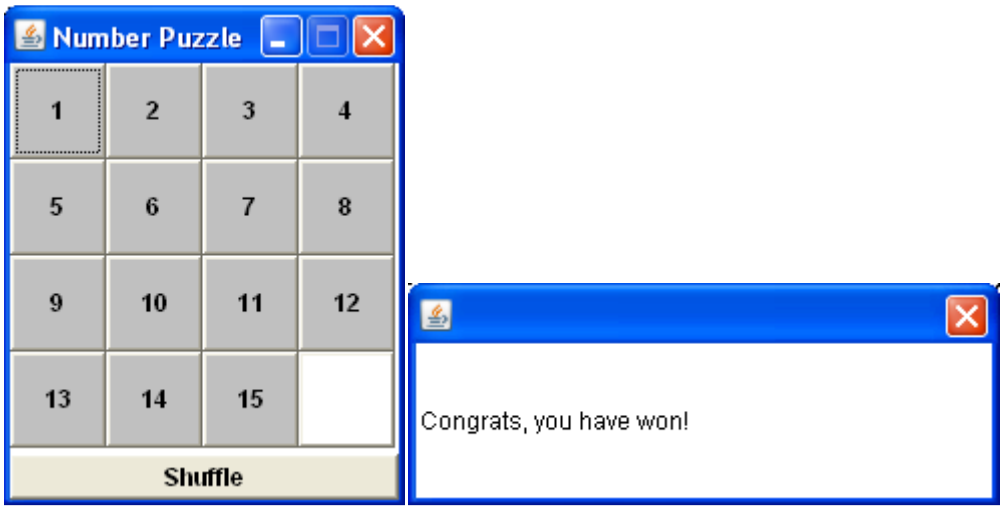
Date: 04th Aug, 2010

1. Write the code to solve the Number Puzzle game. The numbers are Randomly generated and you can shuffle the grid at any point in time.

40



Rules: Arrange the numbers in the grid from 1 to 15 as shown below.



Click on the numbered button adjacent to the empty cell to move the numbered button. The buttons that are diagonally positioned to the empty cell cannot be moved. Only the buttons to the left, right, top and bottom can be moved. Essentially this means that you end up swapping the adjacent numbered button and the empty cell. Here both are buttons. The shuffle button allows you to shuffle the numbers in the grid.

The UI and all the required classes have already been written. You are required to write the logic for the following three methods in the class **CustomNumberPuzzleControl. All these methods will be called by the UI component. If your code is correct, then the entire application works fine and you can play the NumberPuzzle game.**

Happy coding!

1. **Method: getRandomNumbersForGrid()**

```
public int[] getRandomNumbersForGrid() {
	int arr[] = new int[15];

	// write the logic for the method

	return arr;
}
```

Hint:

The base class has a method called getRandomNumber() which will return some random integer value.

Usage:

```
int randomInt = getRandomNumber();   // how to call this method…
```

Use this method to generate a random integer value. This can be any value and make sure you trim it down to some value between 1 to 15.

Fill the array 'arr' with random numbers from 1 to 15 without repeating any of the numbers. This will help in populating the grid with randomly generated numbers

2. **Method: handleButtonClicked()**

```
public int handleButtonClicked(NumberPuzzleGame game){
	int emptyCellId = game.getEmptyCellId();
	Button buttonClicked = game.getButtonClicked();
	Button[] buttons = game.getButtons();

	// write the logic here…

	return (emptyCellId);
}
```

Hint:

The base class provides a method void swapButton(Button emptyButton, Button clicked). Use this method to swap the button that is clicked and the button that is empty.

Usage:

swapButton(buttons[emptyCellId],buttonClicked); // How to call this method

Ensure that you set the emptyCellId with the new value and return the same. Do not bother about the code already written in the NumberPuzzleGame class and how it is used. Please focus on the logic.

3. **Method: checkForWinner()**

```
public boolean checkForWinner(Button[] buttons) {
	boolean winner = true;

	// write the logic for the method.

	return winner;
}
```

Hint:

The base class provides a method called `getIntegerArrayOfButtonIds`(). This method takes the array of Buttons and returns an int[] array of buttonIds.

Usage:

int[] arr = getIntegerArrayOfButtonIds(); // How to call this method

If the numbers in this int[] array are already arranged from 1 to 15 then, we have won the game.

Accordingly write the logic and return the value indicating whether you have won the game or not.

Scoring Pattern:

1. **Method: getRandomNumbersForGrid()** **20 marks**
2. **Method: handleButtonClicked()** **15 marks**
3. **Method: checkForWinner()** **5 marks**

Get all right and score another 10 marks as BONUS.