Assignment 1

CS 1027
Computer Science Fundamentals II

Due date: September 30 at 11:55 pm

Learning Outcomes

In this assignment, you will get practice with:

- Loops, arrays (int, boolean, and Object), conditionals
- Creating and using Strings and constants
- Creating and using classes and objects
- Implementing constructor, toString methods, and other methods
- Overloading a method

Introduction

Bingo is an exciting game of chance that involves a set of cards with an arrangement of numbers in a grid. Players will often have several cards to monitor and will mark the numbers that get called. A bingo caller will call randomly selected numbers slowly one after another until one of the players has a completed line on one of their cards. They indicate that they have a line by shouting out the word "Bingo!" If the player correctly has a line where each of the numbers was called, then they win some type of prize. Below is an example Bingo card. This card would be a winner if the numbers 6, 24, 37, 49, and 66 were called since that would form a line of 5 numbers in a row. Any row, column, or diagonal (the two main diagonals of 5 cells from one corner to an opposite corner) is a possible winning line. One thing to note of Bingo cards is that the middle of the grid contains a "free" square. So, the four lines that contain the middle square only need 4 other numbers to win instead of 5. To help direct players to where a number may be, the numbers a broken up into ranges of 15. The first range from 1 through 15 falls under the B and so instead of calling out the number 7, it would be announced as B-7. Similarly, 16 through 30 falls under the I so it would be called as I-16. There are variations of Bingo but this captures the game's essence.

We will be creating a Bingo Simulator.

Assignment 1

CS 1027
Computer Science Fundamentals II

Provided testing file BingoTests.java

This file will **help** to check if your java Classes are implemented correctly. A similar file will be incorporated into Gradescope's auto-grader. **Passing all the tests within BingoTest.java does not necessarily mean that your code is correct.**

Classes to Implement

For this assignment, you must implement two Java classes: **BingoCard** and **BingoSim**. Follow the guidelines for each one below.

In both these classes, you can (and should) implement more private (helper) methods, if you want to, but you may not implement more public methods. You may not add instance variables other than the ones specified below nor change the variable types or accessibility (i.e. making a variable public when it should be private). Penalties will be applied if you implement additional instance variables or change the variable types or modifiers from what is described here.

BingoCard.java

This class is used to represent a single Bingo Card.

The class must have these private instance variables:

- private int[][] card
- private boolean[][] taken

Also, this class must contain the following constant:

- public static final int[] MY_WINNER
    - This constant array can be used with the BingoCard's Constructor to create a valid Bingo card that is a winner if the following sequence is drawn 73, 20, 23, 7, 32, 1, 16, 29, 68, 38, 52, 17 (there are many solutions...Is the entire sequence required to before a winner is found?)

The class must have the following public methods:

- public BingoCard(int[] data) [constructor]
    - Takes the linear data and places it in the 2D array, card which should be a 5 by 5 array. (note: data only contains 24 entries since the middle square contains no number)
    - A 2D boolean array which is 5 by 5 stores whether their corresponding number in the card array has been drawn. Initially, these will be false except for the middle square which will be set to true.
- public boolean isValid()
    - Returns false if there is a duplicated value or if a number is placed in the wrong column: for example, is if 25 was in the first column (B) instead of the second column (I).
    - Returns true, otherwise

- public void drawn(int number)
  - Updates the taken array. If number is present in the card array, the corresponding entry in taken is set to true. There will be at most one updated value.
- public void drawn(int[] numbers)
  - Updates the taken array. If a value is present in the numbers array, is also present in the card, its corresponding entry in taken is set to true. Many values of taken may be updated.
- public void reset()
  - This method resets all the values back to their original values (see constructor).
- public boolean isAWinner()
  - Returns true if any row, column, or diagonal (twelve possible lines) in the taken array contains only true values
  - Returns false, otherwise
- public int minToWin()
  - Returns the minimum number of numbers that could be drawn for this card to be considered a winner.
- public String toString()
  - Returns a String with the contents of arrays, card and taken, that **exactly** matches the format of given below. Note numbers in square brackets indicate that that number has been taken.
  - Some specifics to focus on when constructing the String: each entry is 4 characters with a space after it. Spaces surround non-taken numbers, whereas brackets surround the taken characters. Thus, there are 25 characters per line. Also note that numbers less than ten have a space in front of them. You can better see the format within the code of BingoTests.java

```
 B     I     N     G     O
[14]   29    39    52    67
[ 2]   22   [35]   58    71
[ 6]  [28]  [FR]  [47]  [66]
[ 4]  [30]   32    [56]  [62]
 12    20    43    [48]   75
```

Assignment 1

CS 1027
Computer Science Fundamentals II

BingoSim.java

This class is used to represent a Bingo Game Simulator.

The class must have these private instance variables:

- private int numCards
  - Stores the number of bingo cards in the simulation
- private boolean[] taken
  - Stores whether the number has been taken or not
- private BingoCard[] cards

The class must have the following public methods:

- public BingoSim(int max) [constructor]
  - Sets up an array of BingoCards to store at most max cards
  - Initializes the taken array representing the numbers taken
- public void addCard(BingoCard b)
  - Puts a BingoCard into the array provided there is space. Otherwise, does nothing with b.
- public String simulate(int[] sequence)
  - Simulates the bingo numbers being drawn in the order of the parameter sequence. The simulation stops as soon as there is at least one winner.
  - Returns a string that includes all the drawn numbers in order and bingo card status indicating of each card's minimum number of numbers to win
  - Below shows the format of the simulation's output. The second line indicates the first number drawn was 42 (since it is the range 31-45, it's an N), and that two of the seven bingo cards must have contained an N-42 since their min-to-win went down to 3. The second number drawn was 64 etc...

```
Simulating ...
1. N-42 4 4 4 3 4 4 3
2. O-64 4 4 4 3 4 3 3
3. G-46 3 4 4 3 4 3 3
4. B-14 3 4 3 3 4 3 2
5. I-17 3 3 3 3 3 3 2
6. G-54 3 3 3 3 2 3 1
```

- public String showCardsWithMin (int min)
  - Returns a string containing the string representation (BingoCard's toString()) of all the BingoCards in this simulation that have a minimum number of numbers equal to min.

- public String toString()
  - Returns a String representing the current status of the simulation. The status includes a list of all the numbers (same format as the BingoCard's toString method), the number of numbers drawn, the min-to-win numbers for each bingo card in the simulation as well as the number of winners.

```
 B    I    N    G    O
  1   16   31  [46]  61
  2  [17]  32   47   62
  3   18   33   48   63
  4  [19]  34   49  [64]
  5   20   35   50   65
  6  [21]  36   51  [66]
  7   22   37   52  [67]
  8   23  [38]  53   68
[ 9]  24   39  [54] [69]
[10]  25   40  [55]  70
 11   26   41   56   71
 12   27  [42]  57   72
 13   28   43   58  [73]
[14]  29   44  [59]  74
 15   30   45   60   75
# Drawn: 17
mins:2 1 3 1 2 3 0
# Winners: 1 out of 7
```

Marking Notes

Functional Specifications

- Does the program behave according to specifications?
- Does it produce the correct output and pass all tests?
- Are the classes implemented properly?
- Does the code run properly on Gradescope (even if it runs on Eclipse, it is **up to you** to ensure it works on Gradescope to get the test marks)
- Does the program produces compilation or run-time errors on Gradescope?
- Does the program fail to follow the instructions (i.e. changing variable types, etc.)

Non-Functional Specifications

- Are there comments throughout the code (Javadocs or other comments)?
- Are the variables and methods given appropriate, meaningful names?
- Is the code clean and readable with proper indenting and white-space?

- Is the code consistent regarding formatting and naming conventions?
- Submission errors (i.e. missing files, too many files, etc.) will receive a penalty of 5%
- Including a "package" line at the top of a file will receive a penalty of 5%

Remember you must do all the work on your own. Do not copy or even look at the work of another student. All submitted code will be run through similarity-detection software.

Submission (due Friday, September 30 at 11:55 pm)

Assignments must be submitted to Gradescope, not on OWL. If you are new to this platform, see these instructions on submitting on Gradescope.

Rules

- Please only submit the files specified below.
- Do not attach other files even if they were part of the assignment.
- Do not upload the .class files! Penalties will be applied for this.
- Submit the assignment on time. Late submissions will receive a penalty of 10% per day.
- Forgetting to submit is not a valid excuse for submitting late.
- Submissions must be done through Gradescope. **If your code runs on Eclipse but not on Gradescope, you will NOT get the marks! Make sure it works on Gradescope to get these marks.**
- You are expected to perform additional testing (create your own test harness class to do this) to ensure that your code works for other dice combinations as well. We are providing you with some tests but we may use additional tests that you haven't seen before for marking.
- Assignment files are NOT to be emailed to the instructor(s) or TA(s). They will not be marked if sent by email.
- You may re-submit code if your previous submission was not complete or correct, however, re-submissions after the regular assignment deadline will receive a penalty.

Files to submit

- BingoCard.java
- BingoSim.java

Assignment 1

CS 1027
Computer Science Fundamentals II

Grading Criteria

Total Marks: [20]

Functional Specifications:

[2] BingoCard.java

[2] BingoSim.java

[10] Passing Tests (some additional, hidden tests will be run on Gradescope)

Non-Functional Specifications:

[3] Well-chosen helper methods

[1] Meaningful variable names, private instance variables

[1] Code readability and indentation

[1] Code comments