IT 179

Spring, 2022

Program 4

Double Linked Lists

**Objective:** Practice with double linked lists

In this program we are designing the game chutes and ladders using a double linked list. Although there are many other better ways to implement this game, this program will help you practice and get a better understanding of linked lists and how they operate.

The game is a turn-based game. All players start on the starting square. Players take turns rolling a single die to move by the number of squares indicated by the die rolled. If, on completion of a move, a player lands on the lower-numbered end of a "ladder", the player moves the up to the ladder's higher-numbered square. If the player lands on the higher-numbered square of a chute, the player must be moved down to the chute’s lower-numbered square. The player who is first reaches the last square of the track is the winner.

Below are the classes and methods you need for this game:

**Game class**

The game class will include the following

* A static inner class **Square** which includes:
  + A reference to the previous square.
  + A reference to the next square.
  + The current square number.
  + A jump value:
    - Either zero, + value or -value.
  + A constructor that will take the square number, the jump value, and the previous square reference.
* Instance variables:
  + A Square reference to the starting square in the board
  + An ArrayList of square references for the players positions
  + An integer to keep track of the current player
* A Constructor that:
  + Takes number of players and instantiates the players ArrayList with that number.
  + Generates a new board by creating 100 squares.
    - It should assign each square a number sequentially from 1 to 100
    - Make connections between the squares
    - Randomly assign a jump value. **Make sure that the player will not jump out of the board. The chances of getting a zero jump should be 75%.**
  + Positions all the players at the starting square.
* A play method:
  + Prompts the current player to press any key to roll the die
  + Rolls the die and generates a random number between 1 and 6.
  + Calls the move method
  + This method will keep running until a player wins
* A move method:
  + Takes the player number and the number of squares to move.
  + Move the player to the square.
  + Make the necessary jump if needed.
  + prints the starting and ending locations of that player indicating any jumps taken
  + returns true if the player won.

**Main Class**

* Main method:
  + Displays welcome message
  + Asks for the number of players
  + Generates the game board
  + Runs the game
  + Announces the winner

**Notes**

* You can format your output any way you like as long as it clearly shows the game play movements for players and accurately displays the winner.
* Good program design is expected. Consider using private methods to break up longer methods into smaller pieces of functionality. Make sure everything has appropriate access control (private or public).
* Good commenting is expected with correct Javadoc style comments.
* Make sure that your program runs as specified on the sample data as well as other test data which you should create yourself.
* Grading criteria:
  + MainClass(30%)
  + Game class with the inner class (70%)
* Deductions
* Late (-10% per calendar day)
* (-40) Syntax Errors
* (-30) Runtime Errors
* (-10) Style and Organization
* Reading material related to the assignment:
  + From Data Structures (IT179 book):
    - Chapter 1
    - Chapter 2

**Submission**

Zip your .java files and submit the .zip file to the Program 4 assignment on ReggieNet.

* + Email submissions will not be accepted
  + Follow the late policy in the syllabus
  + Corrupt files, empty files, invalid format files will result a zero