605.201 Mini-Project 1:

Please do the following to complete this assignment.

Purpose:

The purpose of this project is to provide non-trivial practice in the use of Java programming constructs discussed from the beginning of the course through Module 05 and have a bit of fun doing it.

Resources Needed:

You will need a computer system with Java 8 or greater SE edition run-time and JDK. You may optionally use a Java IDE for example NetBeans, Eclipse, etc. However application builders are not allowed.

Submitted Files:

Design and Analysis:

This word-processed document is required to be written in APA style minus the Abstract section. The file format can be ODT, PDF, DOC, or DOCX. The length of the document should be between 1.5 and 2 pages plus a reference section. The following subjects should be discussed in this order:

1. General program design. How is the program organized? What major data structures were used? How are commands processed? How is the PacMan's state maintained? Etc. This section should take about ½ to 2/3 of the paper content. Do not repeat the project specifications (assume the reader is knowledgeable of the project specifications).
2. What alternative approaches were considered and why were they rejected?
3. What did you learn from doing this project and what would you do differently?

Source file:

All Java source will be in a single text file containing a *single public class* which can be compiled and executed in a standard Java 8 or later SE environment. Multiple methods can be used but all user-created methods must be in a single Java public class contained in the single Java text file.

The format of the Java source must meet the general Java coding style guidelines discussed so far during the course. Please use course office hours or contact the instructor directly if there are any coding style questions.

Submit file:

The submit file is to be a Zip file containing both your design and analysis document and your single Java source text file. Name your Zip file MiniProject1_<Blackboard ID><Section Number> for example: MiniProject1_jdealjr182.zip (note with Windows the .zip extension is added automatically).

Collaboration:

It is encouraged to discuss technical or small design parts of this project with your fellow students.

However the resulting design and implementation must be your own. For example, it is acceptable to discuss different ways of maintaining the PacMan state but not detailed design or implementation information on processing the Move command. When in doubt, ask during office hours or contact your instructor.

Program Specification:

1. Create a new Java program which implements a simple PacMan-type text game which contains the following functionality:
   A) At program startup, constructs and displays a *2-dimensional grid using standard array(s)* (no collection classes allowed) with the size dynamically specified by the user (X and Y sizes can be different). Places the PacMan in the upper-left corner of the grid facing left All grid cells should have the empty cell character of '.' except for the start position of the PacMan which will have the appropriate PacMan symbol (see below). Also 8% of your grid (rounded down if necessary) should contain cookies randomly located on the grid except for the initial PacMan position. The grid must be displayed after each command.

   B) Use these symbols for the grid:
      1. Cookie symbol – shows were cookies are in the grid ('O')
      2. Empty symbol – shows empty unvisited grid cells ('.') (dot)
      3. Visited symbol – shows grid cells where the PacMan has visited (' ') (space)
      4. PacMan symbol depends on the current PacMan facing direction.
         1. Left '>'
         2. Up 'V'
         3. Right '<'
         4. Down '^'

   C) The following menu of commands *must* be provided and must be displayed when appropriate. The command number is what the user should enter to execute the command. Just display the command number and text (ex. 1: Menu), not the explanation of what the command does:
      1. Menu – Display the menu of commands.
      2. Turn Left – turns the PacMan left (counter-clockwise) but the PacMan stays in its current location
         1. Current: up, new: left
         2. Current: right, new up
         3. Current: down, new right
         4. Current: left, new down
      3. Turn Right – turns the PacMan right (clockwise) but the PacMan stays in its current location
         1. Current: up, new: right
         2. Current: right, new down
         3. Current: down, new left
         4. Current: left, new up
      4. Move – Moves the PacMan one grid location in the facing direction if possible. Adds one to the count of move commands if successful or not. If the move command is successful, the previous location is replaced with Visited Symbol (see above). If the move command results in the PacMan moving to a cell where a cookie is located, the cookie is "eaten" and the number of cookies eaten is increased by one.
      5. Exit – exits the program displaying the game statistics of the number of total moves and the average number of moves per cookie obtained.

2. The main processing cycle is the following:
   A) The grid must be displayed after each command showing the effects of the command.
   B) Optionally display the list of commands
   C) Display the grid
   D) Accept user input. Code will be provided for reading user input.
      1. If an invalid command number is entered, an appropriate error message should be displayed and the menu of commands and grid gets redisplayed. An invalid command does not count as a command in the statistics.
      2. Process the command and add one to the number of move commands entered if it is a move command.
      3. If the user enters the Exit command, the program will display the number of commands and the average number of commands per cookie.
   E) If the resulting move places the PacMan over a cookie, indicate the cookie was eaten and add one to the number of cookies eaten for the program statistics.

3. Observe the PacMan demonstration for an example of one implementation.

4. Create a compressed zipped folder containing your Design and Analysis document and your Java source file.

5. Submit your compressed zipped folder as directed by your instructor.

Assessment:

	60%	**70%**	**80%**	**90%**	**100%**	**Weight**
Design & Analysis Document	Majority document subjects covered with accurate information. Some fluff. Document is over half the specified length with 2-4 minor APA style or other minor issues.	All but one document subject covered with accurate information. Maybe some fluff. Document is close to the specified length with 2-4 minor APA style or other minor issues.	All document subjects covered with accurate information. Maybe some fluff. Document is close to the specified length with 1-3 minor APA style or other minor issues.	All document subjects covered with insightful and accurate information. Document is of specified length with 1-3 minor APA style or other minor issues.	All document subjects covered with insightful and accurate information. Document is of specified length and properly formatted to APA style.	**15%**
Program Correctness Note: Compilation	All but two major specified features work but a noble effort is made.	All but one major specified features work but a	All but two minor specified features work. For	All but one minor specified features work. For	All specified features work with neat and easy to understand	**60%**

errors and warnings are part of this evaluation.	For example the PacMan will not move up or down the grid and the PacMan can move off the grid.	noble effort is made. For example the PacMan will not move up or down the grid or the PacMan can move off the grid.	example the PacMan facing is not always correct or the display of the grid is a little off.	example the PacMan facing is not always correct or the display of the grid is a little off.	information display.	
Code Style	Most functionality not properly segmented into methods. Some variable use proper names and types. Some commenting. Some use proper indentation line	Most functionality properly segmented into methods. Most variable use proper names and types. Some commenting. Some use proper indentation line	Most functionality properly segmented into methods. Most variable use proper names and types. Appropriate commenting. Mostly use proper indentation line	Functionality properly segmented into methods. Most variable use proper names and types. Appropriate commenting. Mostly use proper indentation line	Functionality properly segmented into methods. All variable use proper names and types. Appropriate commenting. Proper indentation line continuation, etc.	**25%**

If you have any questions about the specification of this project, contact your instructor *before* the project is due.