CSC 130
Programming Assignment #2

Due Date: ***by 11:55pm*** **Sunday, November 2nd**
Quiz: Monday, November 3rd

Objectives:

- Practice developing program logic
- Practice manipulating a stack and a queue
- Practice exception handling

Overview:

You will be expanding upon the code developed in Programming Assignment #1 by adding history tracking and a reuse pool to the Road class. History tracking will allow backtracking to a previous road state, and the reuse pool will store crashed vehicles, allowing the app to reuse vehicles instead of creating new ones.

Functionality Overview:

LinkedStack for Undo (History Tracking)

Use a ***LinkedStack*** to store the previous state(s) of the road so that you can "undo" the most recent simulation step(s). You will need to push a snapshot of the road onto a stack after each move or collision resolution. Input from the data file will indicate how many steps are to be undone. When data from the file indicates that you are undoing history, you will need to pop the appropriate number of states and restore the road to a previous state.

The data file will be modified to include a code as well as a number. For example, **m 1** would indicate that the current object is to move forward, and **m -1** would indicate that the current object is to move back, but **u 1** would indicate that the last "move" is to be undone, and **u 3** would indicate that the last 3 "moves" are to be undone. Note that **m 0** is considered a "move" even though the state of the road does not change. You can assume that each line of the input file contains a code and an integer in the format described above. Your program should run until the road is empty or it has processed all the data in the data file.

LinkedQueue for Crashed Vehicles (Logging or Reuse Pool)

When vehicles crash and are removed, add them to a ***LinkedQueue*** of "crashed vehicles." Whenever a new vehicle is needed, dequeue from the queue to "reuse" (recycle) a repaired vehicle, instead of creating a new one.

You must use the *LinkedStack* and *LinkedQueue* classes developed in lab as well as the exception classes (*EmptyStackException* and *EmptyQueueException*) associated with them. All the stack/queue classes should be part of this project. Recall that an *EmptyQueueException* is checked, thus you will be required to either acknowledge or handle them. You must decide when it would be appropriate to simply acknowledge that a method can throw an EmptyQueueException and where it would be appropriate to handle the EmptyQueueException.

Plagiarism Policy:

This assignment is to be completed **individually**. It is intended to help you understand the concepts of creating and manipulating stacks and queues, to practice exception handling, and to build your problem-solving skills. You can receive help from another person, but the kind of help that is acceptable should be verbal. You can explain a concept to someone else and use drawings to assist in this explanation, but you should never look at someone else's code, nor should you allow another student to look at yours. Additionally, GenAI tools can be used, but you must indicate what tool you used and how you used it. For example, maybe you used ChatGPT and asked it a series of questions. You will need to include the name of the tool and the prompts/questions posed in a text file submitted to Brightspace. Please note that these tools respond best with specific/detailed prompts. Additionally, you will want to make sure that you understand the code provided by the tool.

Style/Coding/Documentation Requirements:

- You should spend some time making your output as neat as possible. The overall appearance of the output will be graded.
- Neatness Counts!!! Proper indentation is required. Be consistent with your use of white space.
- Be sure to select meaningful variable names.
- There should be a Javadoc comment at the top that includes a title, description and your name. Javadoc comments should precede all methods and must be formatted according to Javadoc standards. Include line comments as needed to explain what your code does. You do not need to comment every line of code.

Submission:

Your submission will include: 1) your code; 2) a text file citing the resources used and/or the prompts you supplied to the genAI tool as well as the link to the video of you explaining your project.

Code (30%)

You must create a .zip file containing your Eclipse project (which must include the data file you used). I must be **able to download your zip file, extract the project, run it and have the code work as expected.** Output should be displayed after each collision and after each state change. Your output must be sufficient for me to determine who is moving, which vehicles have collided and that the result of the collision is as expected. Additionally, when undoing the state of the road, you must clearly indicate how many steps are being undone as well as the states being undone. I will not grade projects that do not meet these specifications.

Explanation Video (40%)

You will upload a link to a 5-minute video of you explaining the project on Brightspace. Your video should include the following:

Video Quality/Details:

- Record the video with both your face and your presentation/code.
- Create a text file that can be downloaded along with your project (.zip file)

  - The text file should contain either a link to the video in a **google drive folder** (that is public to anyone with the link) or to an Unlisted **YouTube** video.
- I will not grade a video that does not meet these specifications. Please be sure that your link works. I will not accept video links after the due date.

The 5-minute video should contain:

- A walk-through of the code that is executed when storing the state of the road after a movement has occurred and the code that is executed when restoring the state of the road
- A walk-through of the code that is executed when vehicles are added to the queue as well as the code that is executed when they are removed from the queue
- A brief description of any challenges you faced

NOTE: Any video content beyond the 5-minute mark will not be viewed and will not be graded.

Submitting Files:
You will be submitting 2 things to Brightspace

1. A zip file containing your Eclipse project (which must also contain your data file)
2. A text file containing a summary of the resources you used (for example, URLs to websites, the name of the GenAI tool used (including the prompts), etc.) and the link to your video.

Quiz (20%)
An in-class quiz will be administered to assess your understanding of the project. You will not be allowed to look at your code during the quiz.