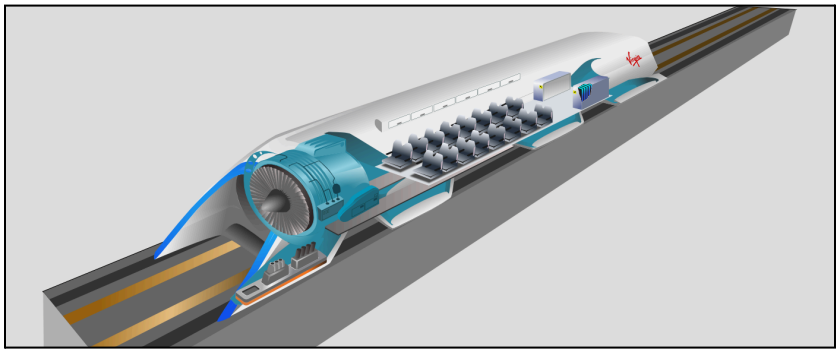
P07 Hyperloop

Overview

Welcome to The Future, where instead of convenient things like accessible public transport and trains, we have the gig economy… and the Hyperloop. In the Hyperloop you'll travel in one of a series of linked pods, which can be first class or economy and maintain a list of passengers (up to a hard limit). For realism, they also have a (low) probability of catastrophic malfunction!



The Hyperloop Manager you'll be completing provides the Hyperloop operators with the ability to track and control the active pods, remove malfunctioning pods, and ensure everything works as well as possible.

Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **11/10**.
15 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
20 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.
10 points	**Manual Grading Feedback**: TAs or graders will manually review your code, focusing on algorithms, use of programming constructs, and style/readability.
50 points	**MAXIMUM TOTAL SCORE**

Learning Objectives

After completing this assignment, you should be able to:

- **Explain** how data is stored in a doubly-linked list, including how additional elements are added in locations other than the head or tail of the list
- **Identify** how an Abstract Data Type influences the structure of an implementation
- **Implement** meaningful unit tests to verify the correctness of a data structure

Additional Assignment Requirements and Notes

Keep in mind:

- Pair programming is **ALLOWED** for this assignment, BUT you must register your partnership ***before the autograder is released***. If you do not do so, you must complete this assignment individually or be subject to Academic Misconduct sanctions.

- The ONLY external libraries you may use in your program are:
  `Pod.java - java.util.Arrays, java.util.Random`
  All other classes - only relevant exceptions

- Use of *any* other packages (outside of java.lang) is NOT permitted.

- You are allowed to define any **local** variables you may need to implement the methods in this specification (inside methods). You are NOT allowed to define any additional instance or static variables or constants beyond those specified in the write-up.

- You are allowed to define additional **private** helper methods.

- Only `LoopStationTester` may contain a main method.

- All classes and methods must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide.

- Any source code provided in this specification may be included verbatim in your program without attribution.

- All other sources must be cited explicitly in your program comments, in accordance with the Appropriate Academic Conduct guidelines.

- Any use of ChatGPT or other large language models ***must be cited*** AND your submission MUST include **screenshots** of your interactions with the tool ***clearly showing all prompts and responses in full***. Failure to cite or include your logs is considered academic misconduct and will be handled accordingly.

- **Run your program locally before you submit to Gradescope**. If it doesn't work on your computer, *it will not work on Gradescope*.

Need More Help?

Check out the resources available to CS 300 students here:
https://canvas.wisc.edu/courses/427315/pages/resources

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Appropriate Academic Conduct, which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - How do I cite my sources?
  - and more!
- Course Style Guide, which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project in Eclipse, called something like **P07 Hyperloop**.
   a. Ensure this project uses Java 17. Select "JavaSE-17" under "Use an execution environment JRE" in the New Java Project dialog box.
   b. Do **not** create a project-specific package; use the default package.
2. Download five (5) Java source file(s) from the assignment page on Canvas:
   a. **ListADT.java** (abstract data type)
   b. **LinkedNode.java** (instantiable class)
   c. **Pod.java** (instantiable class)
   d. **MalfunctioningPodException.java** (custom CHECKED exception)
   e. **LoopStationTester.java** (skeleton code for tester, includes a main method)
3. Create two (2) Java source file(s) within that project's src folder:
   a. **Track.java** (instantiable, implements ListADT)
   b. **LoopStation.java** (instantiable, manages several Tracks)

1. Testing Your Code

The provided **LoopStationTester** file contains only FOUR tester methods, focused on the high-level methods in the **LoopStation** class. LoopStation requires that the methods in **Track** be working correctly!

- Test as you go, and add private tester methods to your tester class to help ensure that Track is working as expected. If methods in Track are not correct, LoopStation will not work correctly either.
- The crux method in this assignment is likely to be the `clearMalfunctioning()` method in LoopStation, which will rely heavily on Track's `remove()` method. Draw diagrams and ensure you know how to remove from all possible locations in a doubly-linked list BEFORE you begin implementing!
- If you haven't already, consider spending some time learning how to use your IDE's debugger tool. It is very helpful for following the execution of your code, especially as we get into more complex programs. There is a tutorial for Eclipse's debugger linked on our resources page.

2. Pod objects

The Hyperloop you will be implementing contains a series of Pod objects. We've provided these for you in their entirety, but you might want to add a `toString()` method to help with debugging. There is no specified format for this, as it's not a required method, but you might consider something like the `Arrays.toString()` of the passenger list, followed by a `*` if the Pod is currently not functional.

Note that the Pod object's `isFunctional()` accessor method has a 1-in-20 chance of CAUSING the Pod to malfunction, which in turn will cause every other Pod method to throw an exception. You're free to use this method if you like, but be aware that using it may cause other problems that you'll need to take into consideration in your tester methods!

It's *very much* in your interest to figure out a more creative solution to determining whether a Pod is currently functional.

3. Track - a doubly-linked list

In class you implemented a singly-linked list. Now you'll implement a doubly-linked list yourself! This variety of linked list is composed of nodes which maintain not only a reference to the next element in the list, but also the previous element, so that you can move through the list in either direction.

NOTE: Track contains three (3) methods NOT defined in ListADT. ***Don't forget to implement these!***

4. LoopStation and management tasks

Once you have a working Track class, move on to the LoopStation and its associated tester methods. This will maintain a set of three different Tracks and move Pods between them. It will also have the ability to detect and remove malfunctioning pods.

Remember to protect against exceptions! No exceptions except those EXPLICITLY stated in the Javadocs should be thrown at any time.

Assignment Submission

Hooray, you've finished this CS 300 programming assignment!

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct and style guide requirements, make a final submission of your source code to Gradescope.

For full credit, please submit ONLY the following files (**source code**, *not* .class files):

- `Track.java`
- `LoopStation.java`
- `LoopStationTester.java`

Additionally, if you used generative AI at any point during your development, ***you must include screenshots*** showing your FULL interaction with the tool(s).

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default this will be your most recent submission.

Students whose final submission (which must pass ALL immediate tests) is made before 5pm on the Wednesday before the due date will receive an additional 5% bonus toward this assignment. Submissions made after this time are NOT eligible for this bonus, but you may continue to make submissions until 10:00PM Central Time on the due date with no penalty.

Copyright notice