Railyard

Team Name:

Manager:

Recorder:

Presenter:

Analyst:

This is a Process Oriented Guided Inquiry Learning (POGIL) activity. You and your team will examine a working program. A series of questions will guide you through a cycle of exploration, concept invention, and application. There is strong evidence that this is more effective (and less boring) than a traditional lecture.

By the time you are done with this activity, you and your team should be able to:

- explain and implement stacks.
- explain and implement queues.
- explain and use interfaces.
- work more effectively as a team.

Your team's recorder is responsible for writing your team's answers to the numbered questions on this form.

After you complete this activity, please fill out the short survey at

http://goo.gl/forms/HXjyuUb2ou

to improve this activity for future users.

Playing the game

Railyard is an original puzzle. Run Railyard.java to play. Each member of your team should play until they've solved it. The setup is different each time, so some players may get easier puzzles than others.

1. Is everyone done running the program and ready to pay attention to the team?

You may need to go back and play the game again to answer some of the questions to come, but you should do so *deliberately*, because your team's manager assigned one or more people to find something out, not merely because you got bored with the conversation or thought you could answer a question better on your own.

2. What two things happen visibly when you click on the > or < symbol?

3. Does your team reach a consensus before writing down an answer?

4. If the symbol is >, what happens when you click on one of the branches on the left side?

5. If the symbol is <, what happens when you click on one of the branches on the left side?

6. What happens when you click on the main line on the right side?

7. How do the top two branches differ from the bottom two?

8. Is there a limit to how many numbered boxes (“rail cars”) can be on one branch?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team’s presenter should be prepared to present any of your team’s previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have. If your team is done before other teams, discuss the following open-ended question:

9. What would be the worst possible starting configuration for this puzzle?

Branch

Examine Branch.java.

10. According to the first line after the Javadoc comment, this is not a class. What is it?

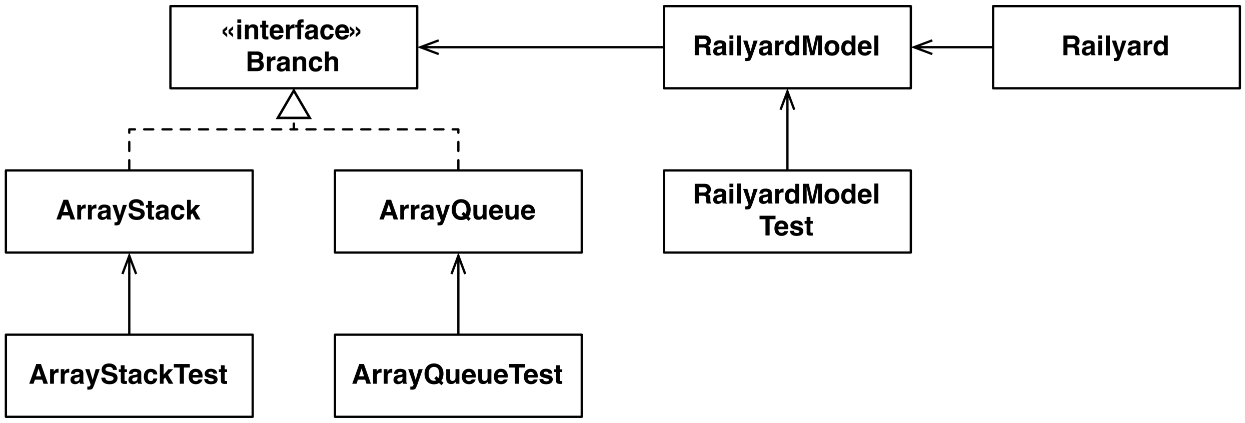
11. What does Branch contain that a class also contains?

12. What does Branch *not* contain that a class contains?

13. Are all members of your team included in discussions?

14. Can you make an instance of Branch? (Try it!)

The relationships between the classes in this project are shown in the UML class diagram below. The hollow-headed arrows from ArrayStack and ArrayQueue to Branch represent is-a relationships; they are dashed because Branch is an interface. They say that an ArrayStack is-a Branch and an ArrayQueue is-a Branch.



Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have. If your team is done before other teams, discuss the following open-ended question:

15. List as many pairs of classes of real-world objects as you can that have has-a relationships (e.g., feet and toes) and that have is-a relationships (e.g., dogs and animals).

ArrayStack

Examine ArrayStack.java. For this activity we'll use the terms "ArrayStack" and "Stack" interchangeably, but in the future you will learn about other kinds of stacks.

16. What special notation is at the top of ArrayStack.java indicating its relationship to the Branch interface?

17. What is the relationship between the methods in ArrayStack and those in Branch?

18. How does your team resolve conflicts about answers? Is this process working well?

19. Looking at RailyardModel.java, which Branches are represented by ArrayStacks?

20. In the game, if you put several things into a stack, which one is removed first: the oldest or the newest? In other words, are stacks "first-in, first-out" (FIFO) or "last-in, first out" (LIFO)?

21. A stack is often described as analogous to a stack of plates, with items added and removed at the top. In the branches in the game, is the top drawn facing the left or the right? What about in the main line?

22. If an ArrayStack contains several things, is the one at index 0 at the top or bottom?

23. If you add something to an ArrayStack whose `size` field is currently 3, where in the `data` array is the new thing placed?

24. If you remove something from an ArrayStack whose `size` field is currently 3, from where in the `data` array does the returned item come?

25. What is the order of the running time of all methods in this class?

26. Which methods modify the ArrayStack? Which methods return values? Are there any methods that do both? Are there any methods that do neither?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have. If your team is done before other teams, discuss the following open-ended question:

27. What would happen if you tried to add something to a full ArrayStack? What would constitute better behavior for this class?

ArrayQueue

Examine ArrayQueue.java. The word "queue" is pronounced like the letter 'Q'.

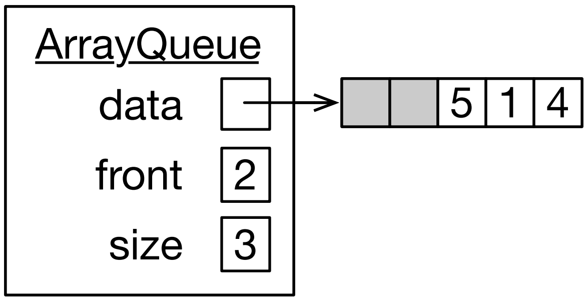
28. Looking at RailyardModel.java, which Branches are represented by ArrayQueues?

29. In the game, if you put several things into a queue, which one is removed first: the oldest or the newest? In other words, are queues "first-in, first-out" (FIFO) or "last-in, first out" (LIFO)?

30. A queue is often described as analogous to a line of people waiting for tickets, with items (people) added at the back and removed from the front. In the branches in the game, is the front drawn facing the left or the right?

31. Are your team members treating each other with respect?

32. Suppose an ArrayQueue `q` currently looks like the picture shown at right. What will it look like after `q.add(8)`?



33. What is the order of the running time of all methods in this class?

Stop here and wait for the other teams. If your instructor has given you a way to indicate that you have reached this point, use it now. Once all teams are ready, there will be a short discussion involving the whole class. Your team's presenter should be prepared to present any of your team's previous answers to the class. This discussion is also a good time for your team (through your presenter) to ask any questions you have. If your team is done before other teams, discuss the following open-ended question:

34. Do you treat your email inbox more like a stack or a queue? Why?

Please fill out the survey at http://goo.gl/forms/HXjyuUb2ou.

This work was supported by the Google Education and Relations Fund's CS Engagement Small Grant Program grant #TFR15-00411.

The principal investigator, Peter Drake, wishes to thank the following for their useful comments: the members of the CS-POGIL project, specifically Clif Kussmaul and Helen Hu; Maggie Dreyer; and various anonymous reviewers.