CSCI-605 Advanced Object-Oriented Programming Concepts

Homework 9 - The Zyxnine River Bridge

1 Introduction

This homework provides practice of techniques for synchronizing multiple threads. You will write a program using Java's thread synchronization mechanisms.

1.1 Problem Description

In the land of the Woolies, the towns of Merctran and Sicstine are on opposite sides of the Zyxnine River. There is a single bridge that connects these two towns. Any Woolie that wishes to go to the town on the other side of the river must cross this bridge.

The bridge between Merctran and Sicstine gets a lot of foot traffic because many Woolies need to go back and forth between the two cities on a regular basis. The bridge itself is very old and the municipal engineers have noticed many cracks developing in the structural members of the bridge. The state of the bridge is so bad that the engineers feel that they need to limit the bridge to carry only three Woolies at a time.

The Woolies are a rather impatient and unruly bunch. Even though they know that the bridge might collapse under the weight of more than three Woolies, plunging them into the Zyxnine River, they will not wait until it is safe to cross. Therefore the government has employed a mean-spirited troll to police the bridge and limit passage to a maximum of three Woolies at a time.

The mean-spirited troll will control the order in which Woolies are allowed to get on and cross the bridge. The troll was instructed to let Woolies get on the bridge in the order in which they arrive at the bridge (no matter which side they arrive on).

Each Woolie must ask permission to get on from the troll at the bridge. After the troll grants a Woolie's request to get on the bridge, the Woolie plods across the bridge for some number of seconds. This time it takes to cross defines it's particular speed, or lack thereof, crossing the bridge.

The bridge is a two-way thoroughfare. Woolies on the bridge can be going different directions at the same time. There is only one troll, who lives under the bridge. From this spot he keeps

track of the Woolies from both sides, and makes sure that the Woolies are allowed on the bridge in the order they arrived, no matter which side the Woolie is on.

2 Questions

1. Complete the following time table that simulates ten Woolies crossing the bridge with a maximum of three Woolies on the bridge at the same time. Each Woolie is listed with their time of arrival to the queue, and their time to cross the bridge. For example, Deb makes the request to cross the bridge at second number 4 and her crossing time is 3 seconds.Assume the following:
   (a) If a Woolie has not yet entered the queue, or has finished crossing the bridge, leave the cells blank.
   (b) Woolies can start crossing upon arrival if there is room on the bridge; otherwise they must wait.
   (c) Assume all Woolies are crossing in the same direction.

   Fill in the cells with these symbols, based on the status of the Woolie:
   (a) crossing bridge: `X`
   (b) waiting in queue: `O`

Name	Arrive	Cross	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
Alf	01	7																				
Bev	02	4																				
Cal	03	6																				
Deb	04	3																				
Eli	05	3																				
Fay	06	2																				
Gia	07	4																				
Hal	08	3																				
Ira	09	3																				
Kim	10	2																				

2. List the order in which the Woolies successfully finish crossing. The list has the time of completion and the Woolie(s) that finished at that time. If two Woolies finish at the same time, list them alphabetically.

3 Implementation

Implement and submit the following files:

1. RunWoolies.java is the provided main program test file that has the start of a suite of tests.
2. TrollsBridge.java is the **monitor** class. This monitor represents the bridge and its troll, who controls and coordinates access so that (1) no more than the correct, specified number of Woolies is on the bridge at the same time, and (2) Woolies get on the bridge in the order they arrive.
3. Woolie.java is the Thread subclass representing a Woolie citizen who is crossing the Zyxnine River Bridge.

3.1 Provided Files

- OutputExample.txt, a sample run of the provided tester.
- RunWoolies.java, main program test file that has the start of a suite of tests.

4 Program Operation

A Woolie's life consists of trying to get on the bridge, crossing the bridge at its own rate, and getting off the bridge on the other side. The `Thread.sleep(long)` method simulates the walking time across the bridge, and the `TrollsBridge` has two methods to support getting on and getting off: `TrollsBridge.enterBridgePlease` and `TrollsBridge.leave`.

4.1 Getting on the bridge

The troll at the bridge will need to maintain a queue of those waiting to get onto the bridge. Since each Woolie executes in a separate thread of execution, the troll's bridge needs to store the passed-in Woolie instance in the queue (See the java.util.concurrent package documentation here.).

Since the troll is mean-spirited and the Woolies are a wild bunch, the troll scowls and yells at each Woolie when they enter. (See the OutputExample.txt for specifics on this scowl message).

When there is enough space on the bridge, `TrollsBridge.enterBridgePlease` returns to the Woolie, the calling thread.

The troll's bridge must ensure that only the Woolie who is actually next in line is the one who gets on and starts crossing the bridge; all other Woolies must continue waiting. The troll must also ensure that there are no more than specified number of Woolies crossing at the same time.

4.2 Getting of the bridge

When a Woolie reaches the other side of the bridge, it must call the `TrollsBridge.leave` method to tell everyone that it has crossed the bridge and is getting off. In this way, the troll operating the bridge will allow the next Woolies who may be waiting to cross the bridge.

Remember: it is possible that multiple Woolie instances leave the bridge before the next waiting one starts crossing. Be sure to not allow a new Woolie to cross the bridge ahead of those already waiting to cross, you do not want to start a fight between the Woolies!!

4.3 Some details about Woolies

Every Woolie has the following information:

1. name
2. crossing time. Like most people, every Woolie walks at a different speed, this field will indicate how much time it takes to the Woolie to cross the bridge.

3. the name of the destination city (Merctran or Sicstine), and
4. a reference to the TrollsBridge that it will cross

All that information is required, you must validate that every Woolie has a value per each attribute. You must also validate that the value is valid. For example, there are only two possible values for destination city (Merctran or Sicstine). If any of the values does not conform with the prerequisites, the program should throw an appropriate exception and terminates.

The Woolie.run() method simulates crossing the bridge. The Woolie tries to get on the bridge right away. The first thing the Woolie must do is ask the troll permission to cross the bridge. Then, the Woolie crosses the bridge, you must use a loop and the Thread.sleep(long) to simulate the Woolie crossing at his/her speed. The Woolie.run() method also produces some output. Finally, when the Woolie is ready to get off the bridge, it must inform the troll that it is leaving the bridge. Study the provided Outputexample.txt file.

5 Testing

The provided RunWoolies.java file is the starting point for the main program test file. Each test case creates multiple Woolies with different crossing times and destinations. The provided test program has 2 simple test cases that show you how to create the TrollsBridge and start the Woolies. OutputExample.txt shows a sample run of the provided tester.

5.1 Add Test Cases

Add at least 2 new test cases to the test program. Be sure that the cases verify that the implemented design ensures the following properties:

1. Multiple Woolies can be on the bridge at the same time;
2. No more than 3 Woolies can be on the bridge at the same time;
3. Woolies can be on the bridge and going in different directions at the same time;
4. Each Woolie remains on the bridge for its specified length of time;
5. Each Woolie eventually gets off the bridge;
6. Each Woolie eventually gets on the bridge; and
7. The test program eventually terminates after all the Woolies have crossed.

Create a bunch of Woolies simultaneously, or stagger their execution by separating their start() calls with calls to Thread.sleep(long).

NOTE: The order of starting threads is not necessarily the order that they run. Insert short sleeps between each start() invocation in order to minimize the chances that the Woolies try to get on the bridge in an order other than the order intended.

6 Submission

You will need to submit all of your code and the answers for questions 1 and 2 to the MyCourses assignment before the due date. You must submit your `hw9` folder and a PDF

file with the answers for questions 1 and 2 as a ZIP archive named "`hw9.zip`" (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this homework). Moreover, the zip file must contain only the java files of your solution. Do not submit the .txt files provided in this homework nor compiled files.

7 Grading

The grade breakdown for this homework is as follows:

- Questions 10%
- Implementation 90%
  - Functionality 70%
    - TrollsBridge.java 40%
    - Woolies.java 30%
  - RunWoolies, the extended test file 10%
  - Code Style 10%