95-702 Distributed Systems

Project 6

Due: Friday, December 6, 5:00 PM

NOTE: This is **not** due at midnight, but 5pm!

Project Topics: Messaging and the Chandy-Lamport Snapshot Algorithm

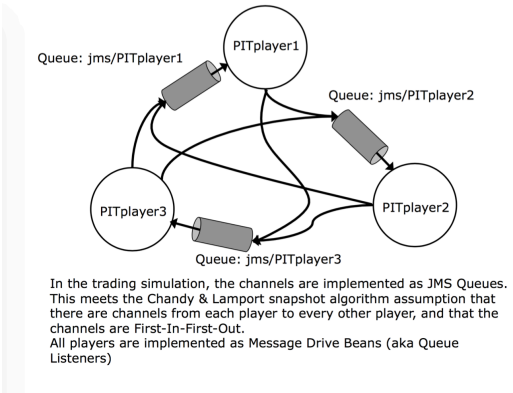
This project has three tasks:

- Add monopoly-seeking behavior to the players in a simulation.
- Add the Chandy-Lamport Snapshot Algorithm into a distributed system.
- Use the Chandy-Lamport Snapshot to understand the behavior of a system.

The distributed system is loosely based on the card game *Pit.*
Wikipedia: https://en.wikipedia.org/wiki/Pit_(game)
Amazon: https://amazon.com/Winning-Moves-Games-Deluxe-Pit/dp/B00000DMBD

Our simulation of the game has six players. Each player has a set of commodities that it trades with the other players. A player initiates a trade by making an offer to another randomly chosen player. The other player can accept or reject the trade. If they accept it, they pay with a commodity of their own. If they reject it, they send back the commodity. The trading action therefore is a fast series of offers, acceptances, rejections, and more offers.

Each of the 6 players is modeled as a Message Driven Bean. The code for each is nearly identical, except for its class name, the Queue it listens to, and an instance variable named myPlayerNumber. Each of the players instantiates a PITPlayerModel which does all the business (game) logic for the simulation.



In the trading simulation, the channels are implemented as JMS Queues. This meets the Chandy & Lamport snapshot algorithm assumption that there are channels from each player to every other player, and that the channels are First-In-First-Out.
All players are implemented as Message Drive Beans (aka Queue Listeners)

All communication between the players is done by JMS Message Queues. Each player has its own Queue that it listens to. Other players can communicate with the player by sending a message to its Queue. (See the picture on left.)

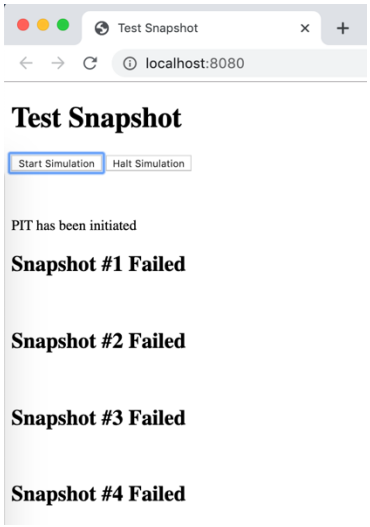
A servlet and a Test Snapshot web page allow the system to be tested. Clicking on the Start Simulation button will start the simulation running. The servlet will send a series of messages to each Player's Queue.

- First it sends a Reset.HALT message to each Player and awaits its acknowledgement response. This ensures that the players stop trading if they had been actively doing so.
- Next it sends a Reset.CLEAR message to each Player to have them reset their data structures and awaits their responses.
- Once all six Players have been reset, it sends a NewHand message to each with a set of commodities. In this way, each Player is assigned its own initial set of commodities. These commodities are also known as *cards*. As soon as each Player receives its NewHand, it begins trading.

Trading continues until the maxTrades threshold is hit. This can be adjusted in the PITPlayerModel so the trading does not go on forever. The trading can also be stopped by clicking the Halt Simulation button on the Test Snapshot page.

A new round of trading can then be started by using the PITsnapshot servlet again.

Initially, the Test Snapshot page will show a list of Snapshot Failed messages (see picture on right). This is normal because the snapshot has not yet been implemented. You will be implementing it.



Setting up the ConnectionFactory and Queues

Unlike the JMS Lab, you will need to set up the ConnectionFactory and Queues outside the program itself. This can be done by editing the tomee.xml file. The tomee.xml file is within the *conf* directory of the *apache-tomee-plus-version* directory. For example: apache-tomee-plus-8.0.0-M3/conf/tomee.xml

The following gist is a tomee.xml file with the necessary resources for Project 6 clearly indicated:

https://gist.github.com/joemertz/b7d0a88cb65022a02fbbc4a9d46bea7d

Insert into your tomee.xml file the gist code shown between the

```
<!-- BEGIN: Add for Project 6 -->
```

and

```
<!--  END: for Project 6 -->
```

Installing the system

Download Fall2019Project6.zip from the course schedule and unzip it.

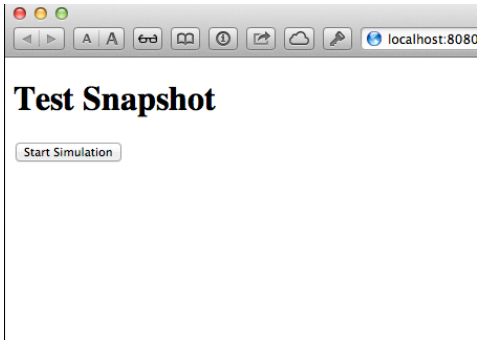
In IntelliJ, “Open” the project folder.

Edit the Run/Debug Configurations and confirm:

- In the Server Tab:
  - ☐ VM options set to: -Dorg.apache.activemq.SERIALIZABLE_PACKAGES=*
  - ☐ The “Before launch, Build...” window lists “Build 2 artifacts”.
- In the Deployment Tab, the following are listed to deploy
  - ☐ PITsimulation:ejb exploded
  - ☐ PITdashboard:war exploded

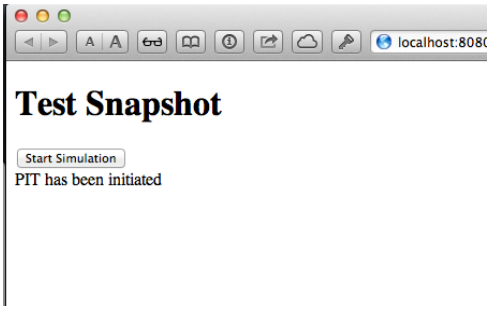
Run TomEE and the PITdashboard (a web application) and the PITsimulation (a set of MDBs) should be deployed.

Testing



Open a web browser and browse to the URL:

http://localhost:8080/



Click on the button to start the simulation and after a short while you will see the response: **"PIT has been initiated"**

Go to the Server Log in IntelliJ and review the output that is produced by the system. It should look something like the following.
(Note: there will be many more lines, and I deleted the frequent and numerous informational lines of the form:`18-Nov-2019 22:05:55.649 INFO [MyJmsResourceAdapter-worker- - 1]`* …)

```
Servlet sending Reset HALT to PITplayer0
PITplayer0 received Reset HALT
Servlet Reset HALT from PITplayer0 ACKNOWLEDGED
Servlet sending Reset HALT to PITplayer1
PITplayer1 received Reset HALT
Servlet Reset HALT from PITplayer1 ACKNOWLEDGED
Servlet sending Reset HALT to PITplayer2
PITplayer2 received Reset HALT
```

* The first person to post a simple way to turn off the production of these INFO messages to Piazza will receive a bonus point.

```
Servlet Reset HALT from PITplayer2 ACKNOWLEDGED
Servlet sending Reset HALT to PITplayer3
PITplayer3 received Reset HALT
Servlet Reset HALT from PITplayer3 ACKNOWLEDGED
Servlet sending Reset HALT to PITplayer4
PITplayer4 received Reset HALT
Servlet Reset HALT from PITplayer4 ACKNOWLEDGED
Servlet sending Reset HALT to PITplayer5
PITplayer5 received Reset HALT
Servlet Reset HALT from PITplayer5 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer0
PITplayer0 received Reset RESET
Servlet Reset CLEAR from PITplayer0 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer1
PITplayer1 received Reset RESET
Servlet Reset CLEAR from PITplayer1 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer2
PITplayer2 received Reset RESET
Servlet Reset CLEAR from PITplayer2 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer3
PITplayer3 received Reset RESET
Servlet Reset CLEAR from PITplayer3 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer4
PITplayer4 received Reset RESET
Servlet Reset CLEAR from PITplayer4 ACKNOWLEDGED
Servlet sending Reset CLEAR to PITplayer5
PITplayer5 received Reset RESET
Servlet Reset CLEAR from PITplayer5 ACKNOWLEDGED
Servlet sending newhand to 0
Servlet sending newhand to 1
PITplayer0 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer1 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer1 numTrades: 0
PITplayer0 numTrades: 0
Servlet sending newhand to 2
Servlet sending newhand to 3
PITplayer2 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer2 numTrades: 0
PITplayer0 offered: Wheat to player: 3
PITplayer2 offered: Wheat to player: 3
PITplayer1 offered: Wheat to player: 4
PITplayer3 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer3 numTrades: 0
PITplayer3 offered: Wheat to player: 2
Servlet sending newhand to 4
PITplayer4 received offer of: Wheat from player: 1
PITplayer4 numTrades: 0
Servlet sending newhand to 5
PITplayer4 accepting offer and paying with: Wheat to player: 1
PITplayer4 hand: size: 0
PITplayer5 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer5 numTrades: 0
PITplayer5 offered: Wheat to player: 1
PITplayer3 received offer of: Wheat from player: 0
PITplayer3 accepting offer and paying with: Corn to player: 0
PITplayer3 hand: size: 11 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat
PITplayer2 received offer of: Wheat from player: 3
PITplayer2 accepting offer and paying with: Corn to player: 3
PITplayer2 hand: size: 11 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat
PITplayer4 new hand: size: 12 Wheat Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley
PITplayer4 offered: Wheat to player: 1
PITplayer3 received offer of: Wheat from player: 2
PITplayer3 rejecting offer of: Wheat from player: 2
PITplayer3 hand: size: 11 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat
PITplayer0 received: Corn as payment from player: 3
PITplayer0 hand: size: 12 Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Corn
PITplayer0 offered: Corn to player: 2
PITplayer1 received offer of: Wheat from player: 5
PITplayer1 accepting offer and paying with: Corn to player: 5
PITplayer1 hand: size: 11 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat
PITplayer1 received: Wheat as payment from player: 4
PITplayer1 hand: size: 12 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat Wheat
PITplayer1 offered: Coffee to player: 4
PITplayer2 received offer of: Corn from player: 0
PITplayer2 rejecting offer of: Corn from player: 0
PITplayer2 hand: size: 11 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat
PITplayer5 received: Corn as payment from player: 1
PITplayer5 hand: size: 12 Corn Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Corn
PITplayer5 offered: Corn to player: 1
PITplayer3 received: Corn as payment from player: 2
PITplayer3 hand: size: 12 Coffee Soybeans Oats Barley Wheat Corn Coffee Soybeans Oats Barley Wheat Corn
PITplayer3 offered: Coffee to player: 4
PITplayer1 received offer of: Wheat from player: 4
```

This is a global history of the actions being taken by the 6 players. It will eventually stop when each Player hits 20000 trades or you click Halt Simulation.

Near the end of the global history will be lines similar to:

```
Servlet Initiating Snapshot via PITplayer4
PITplayer4 received unknown Message type
Servlet: Timeout number 1 without a player reporting.
```

The first message is from the Servlet indicating that it is about to send a marker message into the queue of one of the PITplayers. PITplayer4 then reports that it got a message of unknown type (because it is of type Marker and it doesn't know how to handle them (yet)). The final line is from the Servlet again reporting that it has not received snapshot messages back from all of the players. At this point these console messages make sense because you have not implemented the snapshot algorithm yet.

Back in the browser, test results from 10 snapshots will be added to the window. It will look like the screenshot on page 2.

Again, the snapshots are failing because the snapshot code has not yet been implemented. That is your task; implement the snapshot code.

The Test Snapshot web page is reusable without re-loading. (It uses AJAX.) So, at any time you can just click on Start Snapshot to start the next snapshot.

Do not use Internet Explorer to test with the Test Snapshot page. IE erroneously caches the AJAX requests and you will get invalid results. Use Chrome or Safari instead.

If you get simulation to hit 20000 trades, you have completed the Commodity Trading Simulation lab. Show a TA for credit, and you are now ready to start the project.

Task – Implement the Chandy Lamport Snapshot Algorithm

In class, we discuss the Chandy Lamport Snapshot Algorithm. Implement this algorithm in the system so that you can check the state of players and commodities, such as what player is holding what commodities. Since there are 12 of each commodity given out by PITsnapshot, and none or consumed or added, there should always be a steady state of 12 of each commodity shared between the 6 Players. (Each Player, however, may have more or less than 12 commodities at any given time.)

Some pieces have been provided to you for this task:

The Marker class is defined for passing as the Marker in the snapshot algorithm.

The servlet PITsnapshot will initiate the snapshot by sending a Marker to some player. (All 6 Players run the same PITPlayeModel code, so the PITsnapshot should be able to initiate the snapshot by sending to any of the 6 Players.)

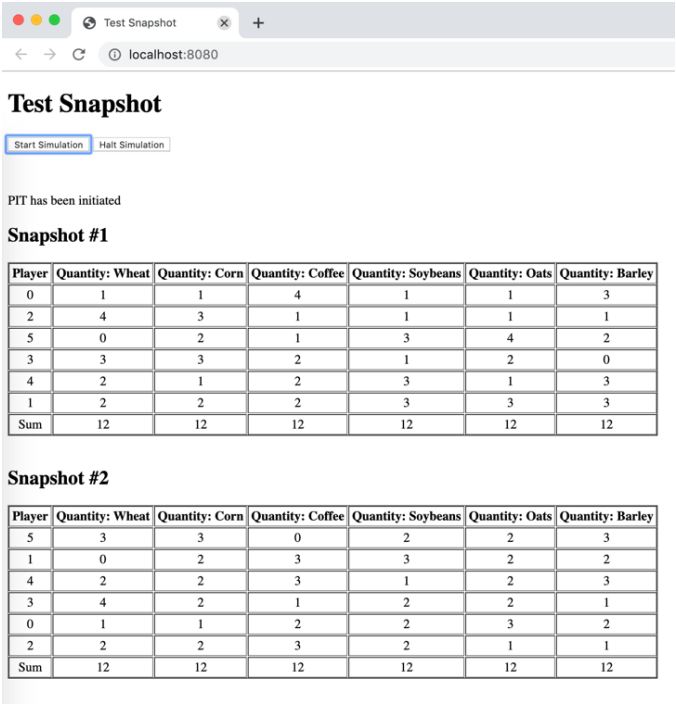
PITsnapshot will then wait and read from the PITsnapshot Queue. Each Player should send a message back to PITsnapshot via that Queue. The content of that message should be an ObjectMessage, and the Object should be a HashMap of commodities and counts (see the code for details). Add to the HashMap the identify of who the snapshot is coming from in the format: state.put("Player", myPlayerNumber);

Finally, the PITsnapshot servlet will report the sums of each commodity back to the browser.

The picture to the right shows only the first two snapshots. In total 10 will be attempted.

The snapshot is successful if the number of each commodity is 12. Until your code is correct, you will probably see cases where there is undercounting (commodities < 12) and overcounting (> 12). Your snapshot code should repeatedly pass all 10 tests.

Therefore, the core of this task is to modify ONLY the code in PITPlayerModel.java. **(No other file should be edited.)** Modify the player model so that it implements the snapshot algorithm for the PITplayers and pass the results to the PITsnapshot servlet.



Task – Add monopoly-seeking behavior to the players in a simulation

The goal of the original *Pit* card game is to corner a commodity market by having all 12 cards of that single commodity; for example, all 12 cards of Corn.

The simulation currently randomly originates trades. Add logic to the simulation so that each player will ***attempt*** to collect a monopoly of one commodity.

In order to do so:

- You can **only** modify PITPlayerModel.java.
- The logic must be generalizable and work for:
  - Any number of players
  - Any number of commodities
  - Any commodity names
  - Any number of commodity cards per player

- Your logic can refer to the information in a TenderOffer, including TenderOffer.sourcePlayer, to decide how to respond to an offer.
- Your logic can decide what playerNumber to tender a new offer to (currently new offers are tendered to a random other player). This playerNumber cannot be hard coded, however (e.g. always offer trades to hardcoded player 4).

The intention is not that your solution **must** achieve a monopoly. Rather, you should have a hypothesis, implement it, and discuss the outcomes you see in the snapshots (see the final task).

Also note that players do not know how many of each commodity there is (and you should not assume any number). Therefore, there is no way for a player to know that they have collected all of one commodity and therefore gained a monopoly. The regular flow of receiving and making offers, accepts, and rejects should continue until reaching 20000 trades.

Task – Use snapshots to understand the behavior of a system

Create a pdf document named MonopolyStrategyAnalysis.pdf with the following content:

A. Name and Andrew ID

B. Describe the strategy you implemented for having each player work toward a monopoly. Write this in prose so that a non-technical person would understand the approach you took for trading toward a monopoly. (200 words maximum)

C. Looking at your sequences of 10 snapshots, perhaps run several or many times, describe what you find interesting about any patterns of players moving toward achieving monopolies. How well did these patterns match or not match what you expected of your strategy? How well did your strategy result in one more players achieving monopolies? (400 words maximum)

Grace days

You may use your remaining grace days on this project, even though these days go into finals week.

What to turn in

1. Create a directory named with your Andrew ID (and only your Andrew id).
2. Take screen shots of a successful snapshot (because of its length, it will probably take more than one) and put it into this new directory.
3. Copy PITPlayerModel.java ***(only!)*** into the directory. This should have been the only file you modified.
    - You should **not** include your whole project, only PITPlayerModel.java.
4. Put the file MonopolyStrategyAnalysis.pdf in the directory.
5. Zip the directory containing these files
   Therefore your zip file, named YourAndrewID.zip, should contain ONLY:
   - A few screenshots
   - PITPlayerModel.java
   - MonopolyStrategyAnalysis.pdf
6. Submit the zipped file to Blackboard.

(Note: You must have used the correctly named Connection Factory and Queues to get full credit.)