Langara
THE COLLEGE OF HIGHER LEARNING.

Department of Computing Science & Information Systems
CPSC 1181
Lab#11
November 19, 2019
Objectives:

Build a multi-threaded server that implements a supplied protocol.

Preparation:

Study your text book and related class notes

Due date:

Due Date: 11:00 PM on Monday November 25, 2019

Where to upload:

zip folder to yourStudentID.zip and upload it to Lab11 in D2L.

Groups: You can do this lab assignment in a group of maximum two, but first you need your instructor's permission, since your marker may consider it as a plagiarism case.

Background (optional) : Interface Map
You do not need to study Interface Map to finish this lab assignment. The related code and class is already developed for you. However, it is recommended to learn following topics. Also study section 15.4 Maps from your text book.
Also check following links:

HashMap

Set

Collection

Iterator

These links provide you with a new frontier of programming in Java that you should gradually become familiar with. Learning these features and tools enable you to develop efficient and more manageable code. Note that these tools are not limited to above mentioned links. These are just sample from Java rich library.

HashMap

There are many classes implemented the Java `Map` interface. The most common class that implements the `Map` interface is the Java HashMap. A `HashMap` is a hash table based implementation of the Map interface. It permits *null* keys and values. Also, this class does not maintain any order among its elements and especially, it does not guarantee that the order will remain constant over time. Finally, a `HashMap` contains two fundamental parameters: initial capacity and performance. The capacity is defined as the number of buckets in the hash table, while the load factor is a measure that indicates the maximum value the hash table can reach, before being automatically increased.

What to do:

Download class Inventory along with test class. The class uses a `HashMap` to control an inventory. Compile, run and study the code. The code provides four methods to manage an inventory.
public void addItem(String item_name, int n); *// Add n units of item specified by item_name to the inventory. The inventory with the item_name is created if the item it is not exist, otherwise it adds n units to the number of available units in the inventory.*

Example:

addItem("Monitor", 3); *// Creates item Monitor with initially 3 units in the inventory*

addItem("Monitor", 2); *// Adds two more units to the item Monitor*

Now there are 5 Monitors in the inventory.

public int checkInventory(String item_name); *// returns number of the available units for the item_name in the inventory. It the item is not available in the inventory then it returns -1; indicating that there is no such item in the inventory. Note that the method returns zero if there no unit left in the inventory.*

Example:

checkInventory("Monitor");

The method should return 5, since there are five monitors in the inventory.

checkInventory("Chicken"); *// returns -1, indicating there is no such item in the inventory*

public int takeItem(String item_name, int n); *// removes maximum n units specified by the item_name from the inventory if available, and returns number of units removed. Return -1 if there is no such item in the inventory.*

Example:

getItems("Monitor", 8); *// it returns 5 since there are only 5 monitors in the inventory, and sets number of available units of the monitors to zero.*

getItem("Monitor", 5); *// returns zero, since no units available anymore for item Monitor.*

public String getThresholds(int threshold); *// returns all items and their available units in the inventory that have equal or less than the threshold value available units with following format: “[item_name1 unitsAvailable] [item_name2 unitsAvailable] ...”*

Example:

addItem("Mouse", 2);

addItem("Keyboard", 10);

getThresholds(3); *// returns "[Monitor, 0] [Mouse, 2]" since number of monitors is zero in the inventory.*

Note: We do not use HashMap for inventory purpose. However, we used it here just for learning purpose. If you want to implement an inventory, then you need to use a Database Management System (DBMs) like mysql.

Par 1: Add locking mechanism to the Inventory class. Since it is going to be accessed by multiple threads, possibly simultaneously, locking is required to ensure the data remains valid.

Par 2:

Sockets are a crucial tool for Internet and client/server programming. A socket is one end of a bidirectional communication pipeline.

Server:

1. Create a server socket that listens on a given port number for a client connection.

2. The server should get commands and respond to them.

3. The protocol for communication from client sides is

    command parameters

command is a request, and it can be one of the ADD_ITEM, CHECK_ITEM, TAKE_ITEM, GET_THRESHOLD, and QUIT.

parameters are the strings or numbers sent to server depend on command type,

The protocol is summarized in the following table:

command	*parameters*	Server Respond
ADD_ITEM item n	String: item, int: n	SUCCEED or FAILED
CHECK_ITEM item	String: item	SUCCEED response or FAILED
TAKE_ITEM item n	String: item, int: n	SUCCEED response or FAILED
GET_THRESHOLD n	int: n	SUCCEED response or FAILED
QUIT		SUCCEED response then close connection with this client or FAILED

4. The protocol for communication from server side is

status answer

status SUCCEED , FAILED

answer is the result of the operation.

5. Server should respond to the client until it gets command QUIT.

6. The server should not crash for any reason. To prevent crashing, server should catch exceptions and sends back the status as FAILED.

Client:

1. Create Client class that runs as a thread. The client should send at least ten request to the server.

2. There should be random delay time intervals (100 – 500 ms) between each request that client sends. You should add this delay to simulate the real time client/server communication.

Server:

Threads are implemented by the JVM and may use OS resources. Multi-threading can simplify many complex problems, into a series of fairly independent parallel threads of execution.

1. The Server should spawn a thread to communicate with each client and responds to it.

2. Test the program with one server and 5 clients.

3. Avoid responding to clients one-by-one. Instead server should communicate with all clients simultaneously; otherwise your solution will not be marked.

What to upload:

1. zip your source files and UML diagram for both server and clinet programs into *yourStudentID.zip* file and upload it to lab11 in dropbox of D2L.

If you are doing the assignment in a group then :

2. The group should be maximum of two students. (more than 2 participant will not be marked) **and you need your instructor permission first. Otherwise it will be considered as plagiarism.**

3. Create file partner.txt. Write both partner's name and student_IDs on this file and add it to *yourStudentID.zip* file.
4. Add names and student_IDs of both participants in the top of your source files and UML diagram.
5. Upload only one copy to the drop box. If both partners upload the files then there will be 20% deduction from your lab assignment as a penalty.

[70 marks]