Cmpe 436 Term Project Report

Name: Serkan Özel

Email: serkan.ozel@boun.edu.tr

Student No: 2015400123

Contents:

1. Abstract
2. Introduction
3. Approach/Methodology
4. Demonstration/Experiments
5. References
6. Appendix

Abstract:

Coupon monster is a multi user multithreaded application in which users compete with each other in order to earn coupons. This project involves binary semaphores to ensure mutual exclusion. In order to earn a coupon a user solves a randomly generated problem. Only one user at a time can open problem solving dialog. Every problem has a solve time associated with it. The harder the problems are, the more the solve time is. The difficulty affects the score a coupon gives. The user with highest score wins the game.

Introduction:

Mutual exclusion means no two threads enter to certain critical region at the same time. In order to satisfy this, binary semaphore, counting semaphore or monitors are widely used in java. I used binary semaphore which means it has a variable with two values, namely a boolean. Whenever a thread acquires the lock the boolean variable becomes false. When the thread releases the lock, the boolean variable becomes true again, so the boolean variable indicates availability of the lock.

Multithreaded servers are a method of building a server in which a thread is created for each connection. The adventage of this method is listener thread can welcome new connections faster than one threaded servers because the server thread’s only job is to listen for new connections with socket.accept() and upon getting one giving the connection socket to a communication thread. Communication thread is the thread that deals with the connection socket, basically all the events after a client connects are handled by this thread.

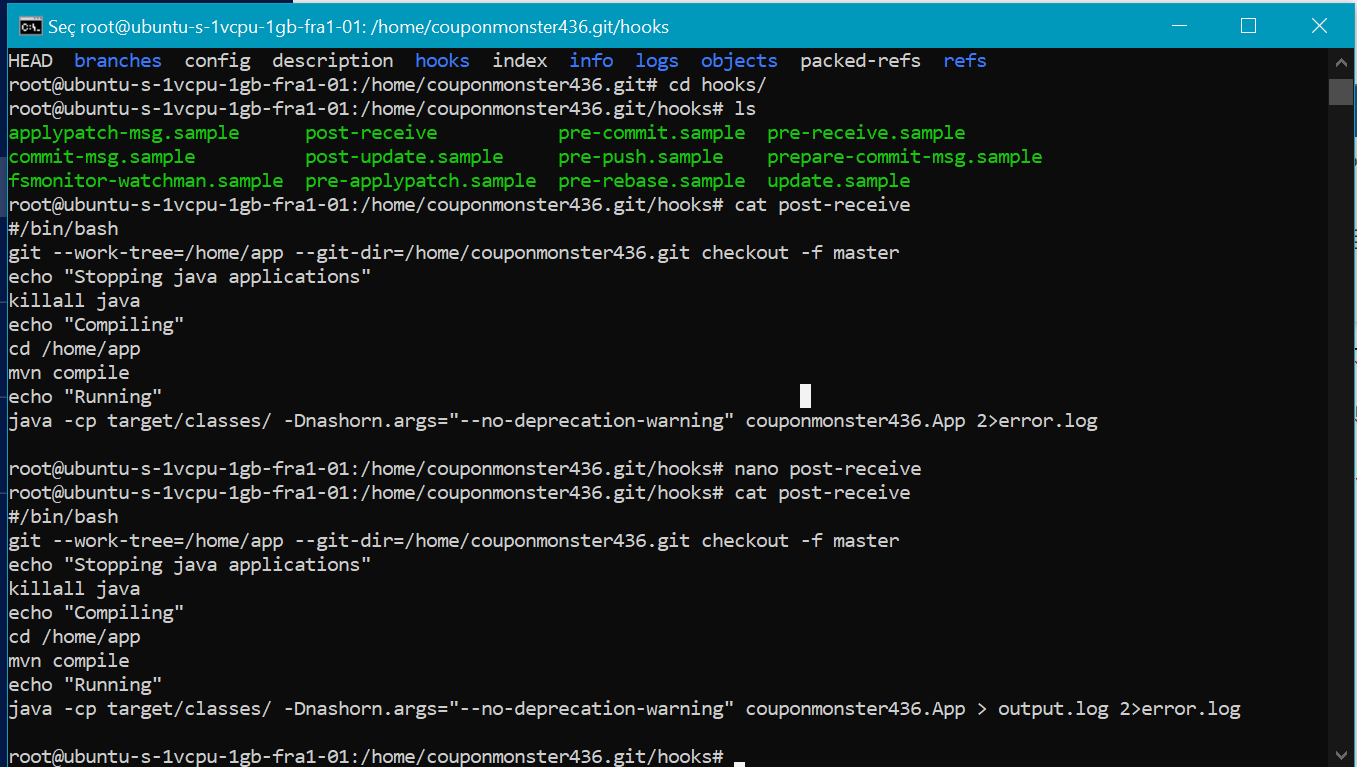
Sockets are like endpoints of two computers in terms of communication. In my project, no two clients can communicate with each other directly. Instead every client communicate with the server and server communicates with all the clients. Server sometimes announces a data to all clients, sometimes communicates with a single client. Basically, sockets send bytes. I send strings encoded as utf-8. Utf-8 is a encoding of strings in which not every characters takes same number of bytes. In encodes how many bytes are needed in the first bytes number of leading 1’s.

Random number generation is important in computer application especially an application like this. My coupon difficulties, coupon generation times, coupon questions are all generated randomly with the help of Math.random() function which generates a random real number between 0 and 1.

Due to the server’s central position it should somewhere that everyone can reach. Therefore, I used a DigitalOcean droplet which is a machine with a IP. In client side that ip is used to connect to the server. Server machine always run the server side code. Server side code has a listener thread which is for listening new user connections. It also has a producer thread which is a random coupon generator thread. Upon generation, producer puts the coupon to an array in memory.

Approach/Methodology

Server side development: In order to update server side code while developing I used git hooks. Basically I created bare repository in the server side which means it’s source code is somewhere else. Upon getting new code the hook code is executed. The hook’s name should be post-receive in the hooks folder and it should be executable bash code. Upon receiving the server builds the code. I used maven to build the server side code.

The hook code does the following actions:

1. Update the project files in /home/app upon receiving a push from local development machine.
2. Stop all running java applications to avoid running multiple
3. Go to /home/app and compile the app with mvn compile
4. Run the application with java -cp(class path) target/classes couponmonster436.App

Random coupon generation: I generated the coupons as follows: Producer thread always runs. In a while loop it checks if the coupon number currently is less than 20. If it is 20 it will stop producing to avoid creating a lot of coupons. Between creation of two coupons producer thread sleeps for a random number of seconds between 1-15.

First, a coupon’s random hash is created. It is a sequence of lowercase english letters that has 30 length. Then the difficulty of the problem is randomly created such that it is between 1 and 15. Using this difficulty the problem statement is generated as follows:

Firstly a random number from 1 up to difficulty is generated and becomes the left most number in the question. Then (difficulty/2) times the following is repeated:

1. Get a random operator that is +, - or \*. Choosing a multiply operator is dependent on the current number of multiply operators. If it is above a (difficulty/5), no more \* can be chosen. This is done to avoid extra hard problems created randomly.
2. If operator is multiply increment number of multiplies and generate a random number that is between 1 and (difficulty/2).
3. Otherwise generate a number that is between 1 and difficulty.
4. Append these operator and number to the problem.

Also, parantheses are added randomly to the problem.

Then, the answer of the problem is calculated by using JavaScript script engine that is built inside Java. It has a method called eval which takes a expression as string a returns result.

Communication: The communication is done via sending utf-8 encoded strings. There are several message types in the application. The first number in a message indicates the message type(from0-9). There is a vector of communication threads called Communications. It is used for broadcasting a message to every connected thread. The pulse message type is type 9. Its main purpose is to detect disconnected clients. I found out that only way of being sure the other side disconnected, is to try to write to the socket and if it gives an error.[1] Message types:

* 0 Get: Send to new users all the current coupons “;” seperated
* 1 To all communication threads, when a new coupon is generated
* 2 To all communication threads, when a coupon is won and no longer exist anymore 3 Get answer from user, check if it is correct return result
* 4 Get selection request from user, try to get lock, if busy return false, else return true
* 5 Get the information of a coupon’s question dialog is dismissed from a client.
* 6 Send user data, name, username, score to the requesting user
* 7 Send all users’ data to the requesting user
* 8 Get new user data from a user and return if it is successfully changed. Usernames must be unique. Names may not be.
* 9 Get empty message, for checking if the other side has died. If someone disconnects I delete the corresponding user from users array, also I remove the disconnecting thread from Communications vector so that no more broadcasts are sent to that thread.

Client Side:

App State: I used idea of application state.The AppState class is a singleton which means it has only a single instance. These are the fields in the AppState class:

**connected**: whether the application is connected to the server or not.

**coupons**: Vector of coupons

**gainedCoupons**: Vector of gained coupons

**onlinePeople**: Vector of online people

**listener and listenerThread**: Listener thread

**attempting**: Indicates whether or not there is an solving attempt in the client side. This variable is set when a coupon’s button is pressed, and cleared when question dialog is dismissed. I used this variable to prevent pressing question attempt button multiple times to open question dialog of multiple coupons.

View-Tabs: I have four tabs:

**Play**: You see the coupons and attempt to solve them.

**Online people:** Online people tab you see the scoreboard.

**Gained coupons**: In this tab you see the coupons you won.

Options: In this tab you can reconnect if your connection is lost, also you can change your username and name.

View- Dynamic data showing: Coupons are received from the server asyncronously from a communication socket. I used Recyclerview for gained coupons and main coupons fragments. For the online people fragment I used grid view and adapter. RecyclerView has its own special adapter. Adapter is a class that takes your data and transforms it into a view by data binding.

Client Side communication: There is a main listener thread in client side which is for receving coupons. I tried to handle coupon answer sending with this thread, but it was very slow since I now create a new thread for coupon answer submit, and username and name change requests. The main thread joins these threads. These threads has a result attribute in them. Main thread checks the result after the thread is exits. I implemented my async post data communication like this.

Demonstration/Experiment

To try my application, download it and then install. Then, when you open the application it will try to connect to the server in the background. If you are connected, you will see “Connected” text in the top right corner. First, you will get a random default username and name like Coupon Monster, couponmonster0. You can change it in the options if you don’t like it. Then, in order to try gaining a coupon press earn this coupon under a coupon. If there is not a coupon, simply wait for 1-15 seconds for other coupon to be generated on server. You gain 1 score for each 1 difficulty coupon. After gaining some, you can check your rank in online people tab.

I have uploaded my application to play store[1]. You can download my application from play store directly. This way your phone will trust my application. If I share a apk link directly, it will say it may be harmful.

References:

[1[https:/stackoverflow.com/a/10241044/9483495](https://stackoverflow.com/a/10241044/9483495)

[2]<https://play.google.com/store/apps/details?id=com.srknzl.couponmonster>

Appendix:

Some screenshots from the application:

