Project 1

Note: the projects must be done individually. Use of third-party code is not allowed. Taking from and also giving work to others is considered plagia-rism/cheating.
There are no exceptions!

Due Date: November 23 (no penalty until November 26)

Today is Election Day

Voters are anxious to vote. Once arrived at the designated voting place they will have to **wait** on line for their ID to be checked (each voter should block on a separate object). They will wait until one of the available ***ID_checkers*** **notifies** them. They will be notified in a FCFS (use a similar implementation to the one in rwcv.java) order.

Once done with the ID check, voters can enter the voting kiosk line. The next step is to fill out a ballot form with their information. There are *num_k* kiosks. The voter will pick the kiosk with the shortest line and **wait** until they get in front so they can enter their information. Due to COVID a lot of regulations are in place. A ***kiosk_helper*** will control all this movement. He will inform the next voter when a kiosk becomes available and that he/she can move on. When a voter is done completing his/her ballot (and usually it takes some time), he/she will let the helper know and the helper will **notify** the next voter on that specific line that he/she can move on.

Finally, the last step. Getting to the scanning machines. There are *num_sm* scanning machines. Before entering the scanning machine area, voters gather in groups of size *num_sm* (use one object for each group). There are *num_sh* **scanning-helpers**. Once all scanners are available, a helper will let the group know. Next the voters of the group will move on and each voter will take one scanner. ¾ of the voters will need help. Decide which voter needs help in a random way. If the voter needs help, he/she will **notify** one of the available scanning-helpers and **wait** to be helped. After getting help they will finally scan their ballot and leave the building with a very patriotic feeling that they exercised they right to vote and maybe tomorrow will be a better day. After all the voters leave the clerks will be able to leave as well.

Develop a monitor(s) that will synchronize the threads (voter, ID_checker, kiosk_helper, scanning_helper) in the context of the story described above. Closely follow the details of the story and the synchronization requirements. Still you have a lot of flexibility so having any two similar implementations will rise a red flag.

You can create any other methods/objects you may need. Make sure that any shared variable/structure is accessed in a mutual exclusion fashion. Besides synchronized blocks use at least one synchronized method.

Default values:
num_voters = 20
num_ID_checkers = 3
num_k = 3
num_sm = 4
num_sh = 2

Do NOT use busy waiting. If a thread needs to wait, it must wait on an object (class object or notification object).

Use only basic monitors as the ones discussed in class (**synchronized methods, synchronized blocks, wait(), notify and notifyAll**). **Use NO** other synchronization tools like locks, concurrent collections, locks…..or even volatile variables.

Use the **age( )** method and appropriate **println** statements to show how synchronization works. It is important to have print statements before, inside, and after critical events. State clearly what the current thread executing the print statement is doing. Also, be sure to include the name of the thread and a timestamp relative to the start time of the program.

Between major events make sure you insert a sleep of random time. Also make sure you give the information necessary to understand what that event is.

Choose the appropriate amount of time(s) that will agree with the content of the story. I haven't written the code for this project yet, but from the experience of grading previous semesters' projects, a project should take somewhere between 50 seconds and at most 2 minutes to run and complete. Set the time in such way that you don't create any exceptional situations.

Do not submit any code that does not compile and run. If there are parts of the code that contain bugs, comment it out and leave the code in. A program that does not compile nor run will not be graded.

Besides the synchronization details provided there are other synchronization aspects that need to be covered. You can use synchronized methods or additional synchronized blocks to make sure that mutual exclusion over shared variables is satisfied.

The Main class is run by the main thread. The other threads must be specified by either implementing the Runnable interface or extending the Thread class. Separate the classes into separate files. Do not leave all the classes in one file. Create a class for each type of thread.

Add the following lines to all the threads you make:

```
   public static long time = System.currentTimeMillis();
   public void msg(String m) {
      System.out.println("["+(System.currentTimeMillis()-time)+"] "+getName()+": "+m);
   }
```

There should be printout messages indicating the execution interleaving. Whenever you want to print something from that thread use: msg("some message here");

NAME YOUR THREADS or the above lines that were added would mean nothing.
Here's how the constructors could look like (you may use any variant of this as long as each thread is unique and distinguishable):

```
// Default constructor
public RandomThread(int id) {
    setName("RandomThread-" + id);
}
```

Design an OOP program. All thread-related tasks must be specified in their respective classes, no class body should be empty.

DO NOT USE System.exit(0); the threads are supposed to terminate naturally by running to the end of their run methods.

Javadoc is not required. Proper basic commenting explaining the program flow, self-explanatory variable names, correct whitespace and indentations are required.

Name your project as follows: LASTNAME_FIRSTNAME_CSXXX_PY
where LASTNAME is your last name, FIRSTNAME is your first name, XXX is your course, and Y is the current project number.
For example: Fluture_Simina_CS344_p1

Zip only the source files (no .rar) .Upload the project on BlackBoard.