1) a high-level analysis;

2) general design of the system and the synchronization tools used;

3) the main classes that intervene with its description (attributes and methods);

4) a diagram of classes that show how they are related;

and 5) the source code, as an annex.

HIGH-LEVEL ANALYSIS

This program simulates the behavior of a hospital whose only function is to vaccinate people.

This hospital has different rooms to accomplish that:

* Reception: In this room, there is an auxiliary worker working as receptionist. The patients will enter, and, after checking with the receptionist whether they have an appointment or not:
  + If they have an appointment, they will wait in the restroom for the auxiliary to tell them an available desk so they can get the vaccine
  + If they don’t have an appointment, they will leave the hospital
* Vaccination Room: This room has 10 desks for people to be vaccinated. There are 10 workers that are in charge of vaccinating the patients, and they will take from 3 to 5 seconds to be vaccinated, as long as there are available vaccines. After that they will go to the first observation desk that is available
* Observation Room: This room has 20 desks for people. The vaccinated patients will wait for 10 seconds to confirm that the vaccine doesn’t have any side effect on them
  + If they don’t have any reaction, they will leave the hospital
  + If they have a reaction, a worker will come to their desk to check that side effect, and after helping with it (taking from 2 to 5 seconds) the patient will leave

As mentioned, there are 3 groups of people with different functionalities here:

* Patients: They have 2 possible special features
  + Having an appointment: 99% of them have an appointment
  + Having a reaction to the vaccine: 5% of them have reactions to the vaccine
* Healthcare workers: They are the ones in charge of vaccinating and checking up the patients. After vaccinating 15 people they have a break of 5-8 seconds until they have to come back. There are some special cases:
  + Helping patients with reaction to the vaccine is their priority, so, if they are working but there’s no one on their desk in the vaccination room, or they are resting, they will move to the observation room to help
* Auxiliary workers: There are 2 really different types:
  + Receptionist: They are in charge of checking if the patients have an appointment or not, and telling them which desk in the vaccination room they should go in. They take breaks every 10 patients
  + Vaccines preparer: They prepare the vaccines doses for the healthcare workers to apply. They take breaks every 20 vaccines

GENERAL DESIGN OF THE SYSTEM AND SYNCHRONIZATION TOOLS USED

After this overview of the program, we can focus on a more programming-oriented view of the problem:

\_\_\_

Synchronization Tools Used:

Semaphores

* Binary
* Non-binary

Locks

Monitors

MAIN CLASSES

Hospital:

ATTRIBUTES

* Reception reception
* VaccRoom vaccRoom
* ObservationRoom obsRoom
* Atomic Integer capacity
* HashMap<Integer, Patient> patients
* HashMap<Integer, HcareWorker> hcareWorkers
* ArrayList<HcareWorker> restroom
* Semaphore semEnterVacc
* Semaphore semEnterObs
* Semaphore semPatients
* Semaphore semException
* MainWindow window;
* CustomLogger clogger

METHODS

* enterHospital(Patient patient)
* enterReception(Patient patient, AuxWorker aWorker)
* enterVaccRoom(Patient patient, int iDDesk)
* enterObservationRoom(Patient patient, int iDDesk)
* getters
  + getVacRoom()
  + getReception()
  + getObsRoom()
  + getClogger()
  + getRestRoom()
  + getPatient
  + getHcareWorkers()
* setters
  + setVacRoom()
  + setReception()
  + setObsRoom()
  + setClogger()
  + setRestRoom()
  + setPatient
  + setHcareWorkers()
* addPatient()
* removePatient()
* addWorker()
* restRoomToString()
* startWindow()

RECEPTION

ATTRIBUTES

* final ArrayList<Patient> waitingQ
* AuxWorker auxWorker1
* ArrayList<Patient> enteringQ
* Semaphore waitingSemaphore
* Semaphore enteringSemaphore
* Hospital hospital
* Lock wLock
* Lock eLock

METHODS

* enterWaitingQueue()
* exitWaitingQueue()
* enterEnteringQueue()
* exitEnteringQueue()
* getters
  + getNextPatient()
  + getEnteringQueue()
  + getWaitingQueue()
  + getAuxWorker()
* setAuxWorker()

VACCINATION ROOM

ATTRIBUTES

* AtomicInteger vaccines
* ArrayList<Desk> desks
* AuxWorker aWorker
* Lock desksLock
* Condition availableDesk
* Lock vaccLock
* Condition vaccinating
* Semaphore semDesks

METHODS

* sitPatient()
* exitPatient()
* getAvailableDesk()
* vaccinate()
* notifyVaccine()
* createVaccine()
* takeVaccine()
* getters
  + getDesks()
  + getAuxWorker()
  + getVaccines()
* setDesks()
* setAuxWorker()

OBSERVATION ROOM

ATTRIBUTES

* ArrayList<Desk> desks
* Lock desksLock
* Condition availableDesk

METHODS

* sitPatient()
* exitPatient()
* getAvailableDesk()
* checkComplications()
* getDesks()

PATIENT

ATTRIBUTES

* int pid
* int randomChance
* Hospital hospital
* int timeToGetDesk
* int timeToVaccine
* int timeWithComplications
* boolean waitToGetVaccinated

METHODS

* run()
* setTimeToGetDesk()
* setTimeToVaccine()
* setTimeWithComplications()
* getPid()
* isInfected()
* hasAppointment()

HEALTHCARE WORKER

ATTRIBUTES

* int hid
* int pVaccinated
* int iDDeskVacc
* int iDDeskObs
* int timeToVaccine
* int timeToRest
* int counter
* int maximum
* boolean beenAwaken
* boolean working
* Lock lock
* Condition noWorkToDo
* Hospital hospital
* boolean isVaccinating

METHODS

* run()
* signalNoWorkToDo()
* isVaccinating()
* isWorking()
* setTimeToVaccine()
* takeBreak()
* getHId()
* getpVaccinated()
* setpVaccinated()

AUXILIARY WORKER

ATTRIBUTES

* int aid
* int counter
* int totalCount
* Hospital hospital
* boolean isResting
* Semaphore semCounter

METHODS

* run()
* availableDesk()
* getAid()
* checkArrivingPatient()
* isResting()
* addToCounter()
* resetCounter()

STRING MANAGER

ATTRIBUTES

* Hospital hospital
* MainWindow window

METHODS

* run()
* textSetter()

CUSTOM LOGGER

ATTRIBUTES

String path

BufferedWriter bw

METHODS

* write()

DESK

ATTRIBUTES

* int iDDesk;
* int iDPatient;
* int iDWorker

METHODS

* leavePatient()
* leaveWorker()
* setPatient()
* getPatient()
* setWorker()
* getWorker()

{}Me pongo con esto mañana prontito{}

CLASS DIAGRAM

{}Esto se hace solo{}