Synchronization Problems.
Systems Architecture

Second Course, First Semester.
Bachelor in Telecommunication Technologies Engineering
Iria Estévez Ayres
Universidad Carlos III de Madrid.
2021–2022

1. Questions

Question 1: Partial Oct 2019
Given the following synchronization problemas, solve them **using exclusively locks in Java**. You should include the code of your solution.
You should reason your response. Answers with no explanation (only code), or answers with no code (only explanation), will not be considered valid.
Section 1.1
Exclusive access to a primitive-type class attribute (e.g. an `int`, shared only by threads of the same class). There are not and cannot be defined more attributes in that class. The class constructor has no input parameter. This class cannot access to variables external to it.
Section 1.2
Thread A must wait until threads C1, C2 and C3 have executed a certain no code portion. The order of C1, C2 and C3 cannot be determined in advance.
Section 1.3
At a given piece of code, only 5 threads executing at the same time.

Question 2: Group of Clients. January 2017
Using only semaphores that **should be** initialized to 1 or 0 (no other init value is admitted as solution), we have just implemented the synchornization of 5 client threads and a server thread (the `try-catch` blocks are not included for the sake of simplicity).

```
class Cliente extends Thread{
    private Semaphore foo,bar,qux,flob;
    private int num;
    public Cliente(Semaphore foo, Semaphore bar,
                   Semaphore qux,Semaphore flob){
        this.foo = foo;
        this.bar = bar;
        this.qux = qux;
        this.flob = flob;
    }
    public void run(){
        bar.acquire();
        foo.acquire();
        num++;
        foo.release();
        if (num == 5)
            qux.release(); //notifies to the server
        else
            bar.release();
        flob.acquire(); //waits to the server
        foo.acquire();
        num--;
        foo.release();
    }
}
```

Section 2.1
The solution is wrong. Identify all the errors. Fix the code of the client.
Section 2.2
Implement the server (at least its run method) to ensure that the systems works as intended.
Section 2.3
Implement the `main` of a class that should create server and clients. You should also create the sempahores, indicating their values (remember, the value should be 0 or 1).

Question 3: Thread order. January 2017
Given the following piece of code:

```
class Shared{
    int condi[];
    int num;
    public Shared(int num){
        this.num = num;
        this.condi = new int[num];
        //missing initialization code
    }
    public void method1(int i){
        synchronized(Shared.class){
            if (condi[i] == 0)
            try{
                this.wait();
            }catch(Exception e){}
        }
    }
    public void synchronized method2(int i){
            condi[(i+1) % num] = 1;
    }
}
```

We will use this class to order the execution of different sets of threads (possibly from different classes). Each set uses a different object from this class to synchronize the threads within the set. Each set is independent from the others, and has a different size (number of threads). Within a set, each thread has an unique identifier, starting with 0. The order of execution within each set is given by the identifiers: $0, 1, 2, \ldots (num-1), 0, 1, \ldots, (num-1), 0, \ldots$.

Section 3.1
Our code is incorrect. Identify all the problems that you find

Section 3.2
Fix the class and solve all the synchronization problems using only locks. Include the missing initialization code.

Section 3.3
Propose a solution for ordering the execution of a set of N threads of the same class using only semaphores.

Question 4: Readers-Writers. Partial Oct. 2018

```
class Shared{
  // shared variables
  //to complete

  public Shared(/*to complete*/){
   //to complete
  }
  public void read(){
    mutexRead.acquire();
    read_count++;
    if (read_count == 1)
      res.acquire();
    mutexRead.release();

     //reading

    mutexRead.acquire();
    read_count--;
    if (read_count == 0)
      res.release();
    mutexRead.release();
  }
```

```
public void write(){
  mutexWrit.acquire();
  writ_count++;
  if (writ_count == 1)
    res.acquire();
  mutexWrit.release();

   //writing

  mutexWrit.acquire();
  writ_count--;
  if (writ_count == 0)
    res.release();
  mutexWrit.release();
}
```

We want to use the previous piece of code to solve the readers-writers problem **giving priority to the writers** and using **only semaphores**. We have multiple groups of readers-writers threads within our system, and we want that each group shares an instance of this class.

Section 4.1

Is our code correct? Identify all the problems that you find (explaining its reason).

Section 4.2

Fix (and complete) the class using **only semaphores**.

Section 4.3

Implement the class Reader and the class Writer.

Section 4.4

Implement the code of a main that creates 2 groups of 20 readers and 20 writers each. Each group share an instance of class Shared. You should indicate the initial value of all variables involved.

Question 5: Readers-Writers Partial Oct 2017

The following piece of code attempts to implement a reader's code of the classic reader-writer problem by giving priority to readers (try-catches are not included for space reasons).

```
class Reader{
    private Semaphore foo,bar,qux,flob;
    private int num;
    public Reader(Semaphore foo, Semaphore bar){
        this.foo = foo;
        this.bar = bar;
    }
    public void run(){
        foo.acquire();
        num++;
        foo.release();
        while (num == 1)
           bar.acquire();
        // leo
        foo.acquire();
        num--;
        foo.release();
        while (num == 1)
           bar.acquire();
    }
}
```

Section 5.1

The previous solution is incorrect. Mark the errors and explain how to solve them. Fix the code of the reader.

Section 5.2

Implement the writer of the classic reader-writer problem giving priority to the readers.

Section 5.3

Implement the main method of a class that creates 10 readers and 10 writers. You should also implement the semaphores, stating their initial values.

Section 5.4

Implement this problem using only locks (synchronizing regions).

Question 6: Sharing a variable outside synch. January 2016
Given the following piece of code:

```
class Hilo extends Thread{
  int contador=0;
  static int compartida;
  public void run(){
    synchronized(this){
     if (contador!=0)
      try{
         wait();
        }catch (Exception e){}
     contador++;
    }
    compartida++;
    notifyAll();
  }
  public static void main(String[] s){
    Hilo h1=new Hilo();
    Hilo h2=new Hilo();
    Hilo h3=new Hilo();
    h1.run();
    h2.run();
    h3.run();
  }
}
```

We want to protect the shared variable `compartida` without using direct synchronized over it, that is,the threads **should not posses the lock** when performing `compartida++` In order to obtain this, we want to use the shared variable `counter`

Section 6.1
Is our code correct? Identify all the synchronization problems that you find

Section 6.2
Solve the synchronization problems implementing a solution using locks but without direct synchronization over the shared variable.

Section 6.3
Solve the synchronization problems implementing a solution using only semaphores. You should state their initial value.

Question 7: Philosophers. June 2015
We want to implement the philosophers problem with a monitor:

```
class Philosopher{
    int state[4];
    final int WAITING=0, EATING=1, THINKING=2;
    public static synchronized void tryToEat(int i){
        state[i] = WAITING;
        if ((state[(i+1) % 4] == EATING) ||
           (state[ (4 + i -1 ) % 4 ] == EATING))
            this.wait();
        state[i] = EATING;
    }
    public synchronized void stopEating(int i){
        state[i] = THINKING;
        notify();
    }
}
```

Is the code correct?

If not, explain the errors and propose an alternative implementation. You can assume that there are only 4 philosopher threads.

2. Problems

Problem 1: Dining Hall. June 2021
There is an unwritten rule in the Dining Hall of a Student Residence: no one is ever sitting at the table alone.
This internal rule gives rise to various situations:

- If you arrive and want to eat, but no one is eating, you must wait for someone to eat.
- If you finish eating and there are more than two people eating, you may leave.
- If you finish eating and there is only one other person eating, you cannot leave inmediately. You must wait for either your colleague to finish, or for another person to arrive.

Implement the whole system **using only semaphores**. Students are threads. You must implement the code of every class, the code of the main program that creates the students and specify the initial value of every variable.

No deadlocks, starvation or lack of mutual exclusion are allowed.

Problem 2: The Paella restaurant A and B models June 2020

In a restaurant they only serve paella as the menu of the day:

- There are X chefs, Y helpers and Z waiters.
- The bar can hold N individual paellas and M pieces of bread.
- Each chef places the individual paellas one by one if there is room at the bar. If there is not enough room, the chef will wait until there is.
- Each helper is cutting bread and placing one by one the pieces of bread at the bar, if there is room. If not, the helper will wait until there is room.
- Each menu consists of 1 paella and 2 pieces of bread.
- Each waiter may serve at least 1 menu and a maximum of 3 menus at the same time. The number of menus served each time is random.
- Each waiter waits to have all the items of his command available (not to wait loaded in the bar) before picking them and taking them to the table that is serving.
- Once the system is started, each chef serves 10 paellas, each helper serves 20 pieces of bread and each waiter completes 10 orders (each order of a random number of menus).

Model A: implement this system **exclusively with semaphores**.
Model B: implement this system **exclusively with monitors**.

The threads must be the chefs, the helpers and the waiters. You must implement a main program that creates everything. Your solution should work for any value of X, Y, Z, N and M. Solutions with deadlock, starvation or lack of mutual exclusion will be considered incorrect.

Problem 3: T.I.A. Factory. Models C and D. June 2020

New robot costume models are being assembled in a top secret factory property of the T.I.A. (Técnicos de Investigación Aeroterráquea):

- Each robot costume consists of one torso and two arms (this is the T.I.A., not Westworld).
- Each torso is generated by the robotic units A.
- Each arm is generated by the robotic units B.
- There are 2 robotic units A and 5 robotic units B.
- At the collection point there is room for 10 torsos and 5 arms.
- When a A or B unit generates an item it leaves the item in the collection point, if there is room for it. If not, the unit waits for room.
- In the collection point 10 C robotic units awaits. Each of them is numbered, from 0 to 9.
- Each C robotic unit has assigned an individual assembly point.
- **Strictly following his turn**, each C robotic unit pick up from the collection point the necessary elements for creating a costume. Then, it will carry the elements to their assembly point, where the costume will be assembled. Finally, the robotic unit will return to the collection point and will wait for its turn.
- The turn will strictly follow the assigned numbering. This assignment is circular: the 0 will collect pieces before the unit 1; the unit 1 before the 2; ...; the 9 before the 0; ...
- Once the system has started, each A unit generates 50 torsos, each B unit generates 40 arms and each C unit tries to assemble 10 costumes.
- The generation time of each torso and each arm is a random number between 100 and 200 seconds. The assembly time is a random number between 200 and 300 seconds.

Model C: implement this system **exclusively with semaphores**.
Model D: implement this system **exclusively with semaphores**.

The threads must be the A, B and C units. You must implement a main program that creates everything.
Solutions with deadlock, starvation or lack of mutual exclusion will be considered incorrect.

Problem 4: Peinador. Partial Oct 2019
Using only semaphores, implement this synchronization problem.

In Peinador, Vigo's airport, there are two runways (runway A and runway B) that can be used for take-off and landing.

When a plane arrives at the airport, it must follow the following steps:

1. Request a runway for landing.
2. If possible, a runway will be assigned to it (always the one with the lowest number of assigned aircraft). At this point, the aircraft takes up part of the airspace of Vigo airport. The plane will remain in space until it can land (if it cannot be assigned a runway, the plane will be will wait until it can be assigned).
3. At that time, it will no longer be part of the airspace to occupy the assigned runway, which he will leave at the end of the manoeuvre.

On the other hand, when a plane wants to take off from the airport:

1. It will request a take-off runway.
2. It will be assigned a runway to take off. The aircraft will remain in the hangar (infinite size) until it can take off, at which point it will occupy the runway assigned to it.
3. It will occupy this runway until the end of the take-off manoeuvre, and then it will be This is the time when it will take up part of Vigo's airspace.
4. It shall remain in that airspace until it leaves it.

The airspace has a capacity of N aircrafts (aircrafts that want to land or aircrafts that are leaving) and each runway may be assigned to M. You must differentiate between the two runways.
You should implement the classes you need, and a main program that create 100 planes that want to land, stay for a random time in the hangar and then take off from the airport.
The solution should work for any M and N value greater than 0. solution assumes that M and N are constants defined elsewhere in the program. Solutions with deadlocks, starvation or absence of mutual exclusion will not be considered valid.

Problem 5: Neville or the wooden bridge (June 2019)
These are difficult times. The one who should not be named has besieged Hogwarts and Neville is guarding the entrance to the wooden bridge, facing the scavengers and werewolves that are trying to get in.

Only a couple of hours ago, Seamus placed charges on the bridge to blow it up if it is necessary. However, this made the bridge much less robust: it can barely bear the weight of 5 magicians at a time, and magicians can only pass in one direction (towards Neville or towards Hogwarts).

In order to maintain the protective spell, 100 magicians must travel the bridge from Hogwarts, get to Neville, cast their protective spells during a random time (different for each magician) and return to Hogwarts for recharge their energy (for a random time). This should be done in a loop for each magician.

The entrance to the bridge where Neville is located is not considered part of the bridge and admits 10 magicians (not counting Neville himself).

Implement the whole system using a monitor. The monitor will model the bridge. The magicians are threads. You must implement the code of all classes involved, the code of the main program that generates the magicians and specify the initial value of all variables.

Solutions with deadlock, starvation, or no mutual exclusion will not be considered valid.

Problem 6: The playground (Partial Oct. 2018)

Using only semaphores, implement this synchronization problem.

In a school there are children of two different ages (each child is a thread), of kindegarden and primary.
To go to recess each child behaves as follows:

- Arrives at the meeting point.
  - If it is a primary school kid: if it is the first one, go to the next step (it is selected). If not, you must wait until you are selected.
  - If it is a child of kindergarden: if it is the first or the second, go to the next step (is selected). If not, you must wait to be selected.
- Check if there are already 3 children.
  - If not, you must wait until they are 3.
  - If there are already 3 children, it will warn the other 2 that they are waiting.
- The 3 check hands (the child prints it out on the screen).
- She goes to the playground. When the last child leaves the meeting point, the following three can meet.

You should implement all the classes that you need, as well as the main program that is executed according to the attached execution example.

Solutions with deadlocks, starvation or absence of mutual exclusion will not be considered valid.

Example of implementation for 3 primary and 6 kindegarden children (the identifiers start at 0):

```
$ java Playground
PRI1 at the meeting point
PRI1 selected
PRI1 We are not 3.
INF1 at the meeting point
INF1 selected
INF1 We are not 3.
INF2 at the meeting point
PRI0 at the meeting point
INF0 at the meeting point
PRI2 at the meeting point
INF2 selected
```

```
INF2 There are 3 of us!
INF3 at the meeting point
INF2 we shake hands
INF4 at the meeting point
PRI1 we shake hands
INF5 at the meeting point
INF1 we shake hands
INF1 The 3 of us are already in the playground. The others can follow
INF0 selected
INF0 We are not 3.
INF3 selected
INF3 We are not 3.
PRI0 selected
PRI0 There are 3 of us!
PRI0 we shake hands
INF0 we shake hands
INF3 we shake hands
INF3 The 3 of us are already in the playground. The others can follow
INF4 selected
INF4 We are not 3.
INF5 selected
INF5 We are not 3.
PRI2 selected
PRI2 There are 3 of us!
PRI2 we shake hands
INF4 we shake hands
INF5 we shake hands
INF5 The 3 of us are already in the playground. I'll leave the others to follow
```

Problem 7: Student residence (Partial Oct. 2017)

Students in a residence have a certain desire to throw parties in the residence rooms. Aware of this, the director of the residence has decided to end this activities by making tours around the rooms.
Both the director and the students behave as follows:

- There is room for an infinite number of students in each room.
- Once inside a room, students can leave whenever they want (whether or not the director of the residence is in the room).
- The director of the residence can only enter a room if there is no students inside or if there are more than the maximum allowed (15).
- Once inside a room, the director can only leave when everyone the students inside have already left.
- If the director of the residence is inside a room, the students should wait for him to come out before going inside again.

Implement the entire system using a monitor. The monitor will model a single room. The students and the director of the residence are the threads. You must implement the code for all classes involved (students, director of the residence, monitor), the code of the main program that generates the students and the director of the residence, and specify the initial value of all variables. You can assume that there is only one director of the residence.

Solutions with deathlocks, starvation, or absence of mutual exclusion will not be considered valid.

Problem 8: Meereen. June 2017

Soon the rumours of the Wildfire virtues as a quick and effective way to secularize a reign. Tyrion Lannister has heard these rumours and, although he considers that his sister is the reincarnation of Aerys IV, he is really partial to obtain as much as Wildfire as he can. Just in case, you know. Moreover, after the disgrace at King's Landing, a group of 300 alchemists (you know, from the alchemist guild, the people that have the secret of creating Wildfire) and they are willing to create Wilfire for one of the Cersei's *Valonqar* (maybe Tyrion will fulfill Maggy's prophecy, you never know).

Tyrion didn't believe in his own luck when the alchemists told him that you can increase the Wildfire power by exposing it to the closeness of alive dragons, and if you leave the vessels of Wildfire at the Dragons' Lair, they will be be even more powerful. Thus, a mechanism to allow the alchemist go into the Dragons' Lair was established:

- Before the actual Lair, there is an antechamber with capacity for 5 alchemists. As the alchemists are really scared, they prefer to not go alone into the lair. Thus, they always wait until a group of 3 alchemists is formed and ready (or not) to enter.
- After the antechamber, there is an *unidirectional entry* with **capacity for only 9 people**.
- Within the lair, **with no dragons present**, there is room for 200 alchemists.
- There is another entry exclusively for dragons.
- Lately, Drogon (the black dragon) has developed a certain taste for the human flesh, thus Tyrion's team decided that when Drogon is at the lair, no alchemist is allowed to be near him.
- If only Rhaegal (the green dragon) or only Viserion (the white dragon) are inside the lair, 100 alchemists are allowed to be inside. If the both are at the lair, 50 alchemists is the maximum.
- Tyrion has already talked with Daenerys, and she has talked really seriously with her draconian sons. She told them that if there are more alchemists than the allowed when they want to enter the lair, the dragons should wait until the number of alchemist diminishes until the allowed maximum. The dragons are not really happy with this, but you know... Mum rules. Thus, in order to enter into their lair they should:

  1. Try to enter, using the dragons' entry (if there are more alchemists than the maximum, they should wait).
  2. They can be at the lair as long as they please.

3. They will exit (using the dragons' entry).

- When an alchemist want to leave a vessel:
  1. S/he will check if there is space for her/him in the antechamber (capacity 5 alchemists). If not, s/he should wait.
  2. Once at the antechamber, s/he will check if there is already a formed group of 3 alchemists to enter. If not, s/he will wait until there is a formed group.
  3. Once a group is formed, s/he will try to enter in the **unidireccional** corridor) (capacity 9 alchemists). They can walk the corridor on their own, without waiting for the rest of he group.
  4. S/he will go through the corridor (time approx. 1 minute).
  5. S/he will enter in the lair.
  6. S/he will leave her/his vessel (approx. 10 minutes, as they should sing while they leave the vessels).
  7. And s/he will try to exit the lair.
  8. S/he will go through the corridor (time approx. 1 minute, capacity 9 alchemists).
  9. S/he will go through the antechamber (capacity 5 alchemists).
  10. S/he will go out.

Implement the **whole** system using **monitors** in Java. Los dragons y los alchemists son los hilos. Deberás implementar el código de todas las clases involucradas, el código del programa principal que genera a los dragons y a los alchemists, y especificar el valor inicial de todas las variables.

Solutions with deadlock, starvation or no mutual exclusion will be considered invalid (and they will be graded with a 0).

Problem 9: Workers implemented with semaphores. June 2015
In a factory, there is 1 worker of type A and 10 workers of type B:

1. The type A worker is sleeping until a type B worker notifies him. When s/he is awake, s/he puts in a platform 10 components and notifies each type B worker. Then, s/he returns to be asleep.
2. All type B workers are working. When s/he finishes with her/his component, s/he checks if all the other type B workers have finished as well. If not, s/he falls asleep. If s/he is the last one, s/he notifies the type A worker (and falls asleep). There is no established order of finishing with the components. When s/he is notified by the type A worker, s/he returns to the work.

Program the synchronization of this system using **only semaphores**. You can assume each worker is a thread.

You should program the run method of both classes and you should also identify which are the shared variables and their initialization values.

You can assume that at the beginning, the A type worker is sleeping and the B type workers are working.

Problem 10: The baboons (June 2014)

Provide the pseudocode of a program that, using semaphores and indicating the intialization value for each, solves the following problem.

Baboons, as any zoo aficionado would know, are voracious yet disciplined animals: they eat a piece of fruit and right afterwards drink some water, each of them taking a random amount of time to either eat the fruit or drink the water. Then, they rest and have another piece of fruit again.

Two baboon groups live at the shore of a river: western baboons and eastern baboons. Western baboons have access to potable water and west baboons have plenty of bananas. The river is full of caimans and crocodriles with the only hobby of watching the rope that communicates the river shores, waiting for some monkey to fall down. The rope is strong enough to support an indefinite amount of baboons crossing over the river, but this allowed only in one way at each time. In other words, if a monkey wants to cross from east to west and there are other baboons crossing it in the same direction, the baboon can cross as well, but if the baboons on the rope are crossing from west to east it must wait until the rope is free to use.

Solutions with deadlocks or innanition will not be considered valid.

Problem 11: The corridor (Partial Dec 2013)

Implements exclusively with semaphores a solution for the following problem.

There is a two-way corridor (there can be people going from left to right and people going from right to left at the same time, provided there is room), with two entrances (the West entrance and the East entrance).
The corridor has different sections (from left to right):

- West Entrance: capacity 1 person
- Section A: capacity 3 people.
- Section A-B: capacity 1 person.
- Section B: capacity 5 people.
- East Entrance: capacity 1 person.

The people must be threads and have a direction (East-West or West-East). Each person takes a random time to cross each section.

The system must work (no starvation, no deadlock and ensure mutual exclusion) for any number of people.

You should implement the code of all classes involved, the code of the main program that generates the people (generates, for example, 100) and specify the initial value of all variables.