# Exam SYNCHRONIZATION

Date: 1 December 2023

Time: 09:00 – 10.30 h

Location: R10 4.11 + 4.12

Teacher: A. Postma

Resources: none

Grading: 100 points (+ 10 points bonus)

Important note:

|  |
| --- |
| **For all exercises, in which code is requested:**  **Explain the working of your code, and add your reasoning about the correctness! Without a decent explanation no points will be awarded!** |

1. Monitor (20 points)

Explain the concept of a monitor.

Imagine we solve the readers/writers problem using a monitor. Should the methods that access the shared data be inside the monitor class, outside the monitor class, or are both options possible? Explain your answer.

# Pocket money problem (30 points)

A father works in a factory where he gets a salary. He has several children which want to have pocket money regularly. This is modeled in the code below.

The code below shows the (incomplete) implementation of a multithreaded application with 1 father thread and 10 child threads. The money of the father is stored in a shared variable fatherMoney.

Due to irregular work, the father regularly receives a variable salary, which lies between 500 and 1000 euro, and which will be added to fatherMoney. Each of the children will regularly receive pocketMoney which will be subtracted from fatherMoney.

The children have to wait if the father does not have enough money to pay the pocket money (fatherMoney can never be negative).

The synchronization of the threads is not yet implemented. It is your task to implement it. Make sure to also add any required variables or synchronization primitives. If needed you may add additional methods as well.

Make your application as efficient as possible (avoid busy waiting, or unnecessary blocking or waking up of threads).

Hint: consider using one or more conditions in your solution (in addition to other synchronization primitives and variables).

**Explain the working of your code, and your reasoning on its correctness!**

|  |
| --- |
| # shared variables  fatherMoney = MyInt(0, "fatherMoney")  salary = MyInt(0, “salary”)  pocketMoney = MyInt(30, “pocketMoney”)  def father():  while True:  salary.v = random.randrange(500, 1000, 1)  # salary between 500 and 1000 euro  # start of critical section  fatherMoney.v += salary.v  # end of critical section  def child():  while True:  # start of critical section  fatherMoney.v -= pocketMoney.v  # end of critical section  def setup():  // start father thread  subscribe\_thread(father)  // Start 10 child threads  for i in range(10):  subscribe\_thread(child) |

# Tourists and bus drivers problem (30 points)

Consider the following code. It shows the (incomplete) implementation of a multithreaded application that simulates the transportation of two groups of 30 tourists each from the airport to two hotels, A and B. Transportation is done by two shuttle buses. One bus driver will drive to hotel A, and the other will drive to hotel B. There is room for 30 tourists in each bus. One group of 30 tourists goes to hotel A and the other group of 30 tourists goes to hotel B.

The actions of the tourists and the bus drivers are simulated in the methods tourist and busDriver, respectively

You may assume that the methods that are invoked in the tourist method and the busDriver method have already been implemented. The details of the implementations of these methods are not relevant for the current exercise.

The application below creates two threads running the busDriver method, and 60 threads running the tourist method. Half of the tourists have destination hotel A and the other half has destination B.

Bus drivers A and B will only drive to their destination hotel if the bus is full.

So each bus driver will wait until 30 passengers are ready to be transported to the destination hotel of the bus. The tourists will have to wait until the bus for their destination hotel is at the airport, and the bus is full. Only then, the methods rideFromAirportToHotel(dest) and driveFromAirportToHotel(dest) are executed.

Note that tourists for hotel A will take the bus with destination hotel A, and tourists for hotel B will take the bus for destination hotel B. Bus driver A will drive to hotel A, and bus driver B will drive to hotel B.

Make changes in the code below so as to realize the requested synchronization (you don’t have to code the actual transportation; only the synchronization is relevant).

Note: Synchronization is only needed before driving from the airport to the hotel; the method doOtherThingsAndDriveBackToAirport() does not need to be synchronized.

**Explain the working of your code, and add your reasoning about the correctness!**

|  |
| --- |
| # TO DO (add shared variables)  N = 30  def busdriver(dest):      while True:          doSomeThings()          # wait until all passengers are seated (TO DO)          driveFromAirportToHotel(dest)          doOtherThingsAndDriveBackToAirport()  def tourist(dest):      while True:          fly()          arriveAtAirport()          # wait until all other passengers and bus driver  # are present (TO DO)          rideFromAirportToHotel(dest)  def doSomeThings(): print("Do some things")  def driveFromAirportToHotel(dest):  print("Drive from airport to hotel " + str(dest))  def doOtherThingsAndDriveBackToAirport():  print("Do other things and drive back to airport")  def fly(): print("Fly")  def arriveAtAirport(): print("Arrive at airport")  def rideFromAirportToHotel(dest):  print("Ride from airport to hotel " + str(dest))    def setup():      for i in range(N):          subscribe\_thread(lambda: tourist("destA"))          subscribe\_thread(lambda: tourist("destB"))      subscribe\_thread(lambda: busDriver("destA"))      subscribe\_thread(lambda: busDriver("destB")) |

# Shop Animali and the guinea pigs (30 points)

Below, the code of a multithreaded application is given. This application simulates the activities of animal shop Animali with respect to buying and selling guinea pigs. The shop owner has a cage in which a maximum of 10 guinea pigs can be placed at any moment in time.

The suppliers will add guinea pigs to the cage, whereas the clients remove guinea pigs from the cage.

For simplicity, we assume that suppliers of guinea pigs will add guinea pigs to the cage one at a time. Clients will remove one guinea pig from the cage at a time if available.

In the application given below, we create and start a large number of supplier threads, and a large number of client threads.

The actions of the suppliers of guinea pigs and of the clients have been modeled in the methods supplier and client, as indicated below.

Synchronization (according the above synchronization rules) is not yet implemented, but it is your task to implement it in the following four methods:

- enterSupplier

- exitSupplier

- enterClient

- exitClient

Please implement the required synchronization in the above methods (on a separate piece of Fontys exam paper). Do not change the code of the rest of the application. Make sure to avoid unnecessary stopping and waking up of threads as much as possible. If needed, add shared variables to the code, and explain how they are used.

**Explain the working of your code, and add your reasoning about the correctness!**

|  |
| --- |
| # shared variables (TO DO)  def enterSupplier (): # TO DO  def exitSupplier (): # TO DO  def enterClient(): # TO DO  def exitClient(): # TO DO  def supplier():  while True:  growGuineaPig()  enterSupplier()  # start critical section  addGuineaPigToCage()  # end critical section  exitSupplier()  def client():  while True:  buyGuineaPig()    enterClient()  # start critical section  removeGuineaPigFromCage()  # end critical section  exitClient()  def growGuineaPig(): print("Grow guinea pig")  def addGuineaPigToCage(): print("Add guinea pig to cage")  def buyGuineaPig(): print("Buy guinea pig")  def removeGuineaPigFromCage(): print("Remove guinea pig from cage")  def setup():  for i in range(65): # a large number  subscribe\_thread(supplier)  for i in range(157): # a large number  subscribe\_thread(client) |