SYSTEMS ARCHITECTURE

LAST NAME: .....
NAME: .....

LABORATORY SESSION 11: Threads and locks

PROBLEM 1 – Introduction

In this exercise, we are going to learn about how to use threads and locks. In order to do that, follow the next steps:

1) Execute this example about father and son without using threads

```
class SonWithoutThread{
    private int ident;
    public SonWithoutThread (int id){
        this.ident =id;
    }
    public void run() {
        if (this.ident == 0)
            System.out.println("Hello Luke");
        else
            System.out.println("Skywalker");
    }
}

public class FatherWithoutThreads {

       public static void main(String[] args) {
             SonWithoutThread h1,h2;
         h1 = new SonWithoutThread(0);
         h2 = new SonWithoutThread(1);
         h2.run();
         h1.run();
         System.out.println("I am your father");

       }
}
```

2) Execute the same example using 3 threads and locks

```
/**JavaFile*************************************************************

  FileName    [Example with Locks]

  Synopsis [Example of Locks]

  Author      [Iria Estevez-Ayres <ayres@it.uc3m.es>]

  Copyright   [Copyright (c) 2020 Carlos III University of Madrid
  All rights reserved.

  Permission is hereby granted, without written agreement and without license
  or royalty fees, to use, copy, modify, and distribute this software and its
  documentation for any purpose, provided that the above copyright notice and
  the following two paragraphs appear in all copies of this software.

  IN NO EVENT SHALL THE CARLOS III UNIVERSITY OF MADRID BE LIABLE TO ANY PARTY
  FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING
  OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE CARLOS III
  UNIVERSITY OF MADRID HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

```
  THE CARLOS III UNIVERSITY OF MADRID SPECIFICALLY DISCLAIMS ANY WARRANTIES,
  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
  FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS ON AN
  "AS IS" BASIS, AND CARLOS III UNIVERSITY OF MADRID HAS NO OBLIGATION TO
  PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.]

*****************************************************************************/


/**
 * Main class of the Java program.
 */
class SonWithLocks extends Thread{
    private int ident;
    static int contador;
    public SonWithLocks (int id){
        this.ident =id;
    }
    public void run() {
        if (this.ident == 0){
            System.out.println("Hola");
            synchronized(SonWithLocks.class){
                contador++;
                SonWithLocks.class.notifyAll();
            }
        }else if (this.ident == 1){
            synchronized(SonWithLocks.class){
                while (contador!=1){
                    try{
                            SonWithLocks.class.wait();
                    }catch(Exception e){}
                }
            }
            System.out.println("Luke");
            synchronized(SonWithLocks.class){
                contador++;
                SonWithLocks.class.notifyAll();
            }


        }
        else{
            synchronized(SonWithLocks.class){
                while (contador!=2){
                    try{
                            SonWithLocks.class.wait();
                    }catch(Exception e){}
                }
            }
            System.out.println("Skywalker");
            synchronized(SonWithLocks.class){
                contador--;
            }
        }
    }
}
public class FatherWithLocks {
    public static void main(String[] arg) {
       SonWithLocks h1,h2,h3;
        h1 = new SonWithLocks(0);
```

```
        h2 = new SonWithLocks(1);
        h3 = new SonWithLocks(2);
        h2.start();
        h1.start();
        h3.start();
        try{
        h1.join();
        h2.join();
        h3.join();
        }catch(Exception e){}
        System.out.println("Soy tu padre");
    }
}
```

3) Why do you think the execution between the two executions are diferent?

PROBLEM 2 – Bank account

In this problem, we are going to analyze the code of a bank account Where Scrooge depois 100 euros, and Donald withdraws that, but if there are syncronization issues, the balance could be wrong. Follow the steps to solve this exercise:

1) Execute the code of BankInitial.java

```
/**JavaFile*********************************************************************

  FileName    [Program that simulates a BankAccount]

  Synopsis [Scrooge McDuck and Donald share an account. Scrooge deposits
  money, Donald withdraws money]

  Author      [Iria Estevez-Ayres <ayres@it.uc3m.es>]

  Copyright   [Copyright (c) 2019 Carlos III University of Madrid
  All rights reserved.

  Permission is hereby granted, without written agreement and without license
  or royalty fees, to use, copy, modify, and distribute this software and its
  documentation for any purpose, provided that the above copyright notice and
  the following two paragraphs appear in all copies of this software.

  IN NO EVENT SHALL THE CARLOS III UNIVERSITY OF MADRID BE LIABLE TO ANY PARTY
  FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING
  OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE CARLOS III
  UNIVERSITY OF MADRID HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  THE CARLOS III UNIVERSITY OF MADRID SPECIFICALLY DISCLAIMS ANY WARRANTIES,
  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
  FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS ON AN
  "AS IS" BASIS, AND CARLOS III UNIVERSITY OF MADRID HAS NO OBLIGATION TO
  PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.]

**********************************************************************************/
import java.lang.*;
import java.util.*;

/*
 * Arquitectura de Sistemas II.
 * a) Que hace el codigo
 * b) Haz que tanto Donald como Scrooge llamen a sus funciones 10 veces
 *   Que ocurre ahora?
```

```
 */
class AccountInitial{

    private float balance = 0;
    private Random rnd = new Random();

    public void depositMoney(float amount){
       float tmp =  balance;

       System.out.println("(Adding money): the initial balance is: " + tmp);

       try {
           Thread.sleep(rnd.nextInt(1000));
       } catch (Exception e) {}

       tmp +=amount;

       balance = tmp;

       System.out.println("(Adding money): the final balance is: " + balance);
    }

    public void withdrawMoney(float amount){

       float tmp =  balance;


       System.out.println("(Withdrawing money): the initial balance is: " +
tmp);

       try {
           Thread.sleep(rnd.nextInt(1000));
       } catch (Exception e) {}

       tmp -=amount;

       balance = tmp;

       System.out.println("(Withdrawing money): the final balance is: " +
balance);
    }

}

class ScroogeInitial extends Thread{

    private AccountInitial account;

    ScroogeInitial(AccountInitial account){
       this.account = account;
    }

    public void run(){
       account.depositMoney(100);
    }
}

class DonaldInitial extends Thread{

    private AccountInitial account;
```

```
    DonaldInitial(AccountInitial account){
       this.account = account;
    }

    public void run(){
       account.withdrawMoney(100);
    }
}

public class BankInitial{

    public static void main(String[] args){
       AccountInitial account = new AccountInitial();
       DonaldInitial donald = new DonaldInitial(account);
       ScroogeInitial tiogilito = new ScroogeInitial(account);
       donald.start();
       tiogilito.start();
    }
}
```

In the previous execution, you may have realized that the same value does not always remain in the account, because there is a race condition.

2) Analyze the previous code and explain what the code is doing and why there is a race condition
3) In order to amplify the problem, make Donald and Scrooge call the functions 10 times
4) Solve the synchronization problem using locks

PROBLEM 3

Luke and Darth Vader have a conversation and each of them has 3 turns in the conversation. The sentences in each turn are the following:

Luke:

- (Luke): He told me enough! He told me YOU killed him!
- (Luke): No. No. That's not true. That's impossible!
- (Luke): No! No!

Darth Vader.

- (DV): If you only knew the power of the Dark Side. Obi-Wan never told you what happened to your father.
- (DV): No. I am your father.
- (DV): Search your feelings, you KNOW it to be true!

You are given a code Where both Luke and Darth Vader are threads and they will print the three messages. However, if they are not syncrhonized, the order of the conversation could not be natural. Modify the code so that the threads are synchronized using locks and the order of the conversation is as follows:

- (DV): If you only knew the power of the Dark Side. Obi-Wan never told you what happened to your father.
- (Luke): He told me enough! He told me YOU killed him!
- (DV): No. I am your father.
- (Luke): No. No. That's not true. That's impossible!
- (DV): Search your feelings, you KNOW it to be true!
- (Luke): No! No!

The initial code is as follows:

```
/**JavaFile*************************************************************

  FileName    [Program that prints the dialogue between Luke Skywalker and
  Darth Vader]

  Synopsis [Contains three classes, that should be executed as threads ]

  Author      [Iria Estevez-Ayres <ayres@it.uc3m.es>]

  Copyright   [Copyright (c) 2019 Carlos III University of Madrid
  All rights reserved.

  Permission is hereby granted, without written agreement and without license
  or royalty fees, to use, copy, modify, and distribute this software and its
  documentation for any purpose, provided that the above copyright notice and
  the following two paragraphs appear in all copies of this software.

  IN NO EVENT SHALL THE CARLOS III UNIVERSITY OF MADRID BE LIABLE TO ANY PARTY
  FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING
  OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE CARLOS III
  UNIVERSITY OF MADRID HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

  THE CARLOS III UNIVERSITY OF MADRID SPECIFICALLY DISCLAIMS ANY WARRANTIES,
  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
  FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS ON AN
  "AS IS" BASIS, AND CARLOS III UNIVERSITY OF MADRID HAS NO OBLIGATION TO
  PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.]

**************************************************************************/
import java.lang.*;
import java.util.*;

/*
 * Arquitectura de Sistemas (2021-22)
 *
 * Ejercicio en clase:
 *
 * Synchronize Luke and Darth Vader so that the dialogue makes sense.
 *
 * The code to synchronize must be inside class Lock
 *
 */

class Lock{

    private int value = 0;

    public void espero(){
    }

    public void aviso(){
    }
}

class Luke extends Thread{

    private Lock lockLS, lockDV;

    Luke(Lock lockLS, Lock lockDV){
          this.lockLS = lockLS;
```

```
            this.lockDV = lockDV;
    }

    public void run(){
        Random rnd = new Random();
           System.out.println("(Luke): He told me enough! He told me YOU killed
him!");
           try {
               Thread.sleep(rnd.nextInt(2000));
           } catch (Exception e) {}

           System.out.println("(Luke): No. No. That's not true. That's
impossible!");
           try {
               Thread.sleep(rnd.nextInt(2000));
           } catch (Exception e) {}

           System.out.println("(Luke): No! No!");
    }
}

class DarthVader extends Thread{
    private Lock lockLS, lockDV;

    DarthVader(Lock lockLS, Lock lockDV){
           this.lockLS = lockLS;
           this.lockDV = lockDV;
    }

    public void run(){
        Random rnd = new Random();
           System.out.println("(DV): If you only knew the power of the Dark
Side. Obi-Wan never told you what happened to your father.");
           try {
               Thread.sleep(rnd.nextInt(2000));
           } catch (Exception e) {}
           System.out.println("(DV): No. I am your father.");
           try {
               Thread.sleep(rnd.nextInt(2000));
           } catch (Exception e) {}

           System.out.println("(DV): Search your feelings, you KNOW it to be
true!");
    }
}
public class StarWars{

    public static void main(String[] args){
      Lock lockLS = new Lock();
      Lock lockDV = new Lock();
      Luke luke = new Luke(lockLS, lockDV);
      DarthVader daddy = new DarthVader(lockLS, lockDV);
      daddy.start();
      luke.start();
    }
}
```

PROBLEM 4 – Extraordinary exam 16.06.2023

Given the following software solution for the critical section problem, indicate which of the three fundamental requirements it satisfies and provide a reasoned justification for each answer. Assume that "turn" is a shared variable that is initialized at the beginning, and the code is executed by two processes, P0 and P1, as follows:

Process P0 code

```
do {
        while (turn == 1);
        critical section
        turn = 1;
        reminder section
} while (true);
```

Process P1 code

```
do {
        while (turn == 0);
        critical section
        turn = 0;
        reminder section
} while (true);
```

PROBLEM 5 – Partial exam October 2019

Given the following synchronization problems, 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 5.1

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 5.2

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 5.3

At a given piece of code, only 5 threads executing at the same time.

PROBLEM 6 – 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 possess the lock** when performing `compartida++`. In order to obtain this, we want to use the shared variable `contador`.

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 without direct synchronization over the shared variable.