Práctica 3: Monitores

1- Dadas las siguientes clases

```
public class MiMonitor {

    public int i = 0;

    public synchronized void add()
    {
       for(i=0; i < 100000; ){i++;}
    System.out.println(i);

    }

    public synchronized int get(){
      return i;
    }

    public static void main(String args[])
    {
       try{
           MiMonitor s =  new MiMonitor();
           MiThread hola = new MiThread(s);
           MiThread hola2 = new MiThread(s);
           hola.start();
           hola2.start();
           System.out.println(s.get());
            }catch(Exception e){System.out.println(1);}

    }
}


public class MiThread extends Thread {
   public MiMonitor m;

   public MiThread(MiMonitor m){
     this.m = m;
   }

   public void run()
   {
     m.add();
   }
}
```

a) Indicar cuáles son los posibles resultados de las ejecuciones de main

b) Indicar cuáles son los posibles resultados de las ejecuciones de main si se cambia la definición de

```
public synchronized int get()
```

por

```
public int get(){
```

2- Definir en Java utilizando métodos synchronized una clase Contador que sea segura concurrentemente.
3- Definir en Java una clase Semáforo que implemente las operaciones de un semáforo utilizando métodos *synchronized*.
4- Redefinir la clase Molinete vista en la primera clase de modo de utilizar la clase Semaforo definida en el punto anterior en lugar de la clase Semaphore de Java.
5- Dar una implementación en Java utilizando métodos *synchronized* un buffer de números enteros cuya dimensión está prefijada al momento de la creación.
    a. Implementar una clase productor que permita agregar a un buffer dado al momento de creación números naturales consecutivos
    b. Implementar una clase consumidor que tome los elementos de un buffer dado al momento de creación y los muestre por pantalla.
    c. Dar un programa que cree un buffer de dimensión 2 y active concurrentemente un consumidor y un productor.
6- Modificar la solución anterior de modo tal de utilizar dos variables de condición (una indicando que el buffer contiene elementos y otra que el buffer tiene posiciones libres).
    a. Comparar esta solución con la dada en el punto anterior
7- Implementar en Java una solución para el problema de los lectores/escritores utilizando métodos *synchronized* (y variable de condición implícita).
8- Dar una implementación para el problema de los lectores/escritores utilizando locks y variables de condición.
9- Considerar el siguiente problema: Se desea modelar el hecho de un bote que utilizan las personas para cruzar un rio. En todo momento el bote se encuentra en una de las dos costas del río (norte o sur). Las personas que llegan a una de las costas pueden abordar el bote si el mismo se encuentra en esa orilla y aún tiene lugar disponible (el bote tiene una capacidad fija que se define al momento de construcción). El bote puede atravesare el río sólo cuando tiene una autorización y cuando está completo. Cuando llega a la orilla, descarga a todas las personas y carga las que están esperando. Cuando se llene y vuelva a tener la autorización, vuelve a cruzar el río en dirección opuesta.
    Notar que la autorización puede llegar antes o después de que esté completo.
    a) Modelar el problema utilizando monitores
    b) Implementar su modelo en Java.
    c) Su modelo contempla el hecho de que las personas que esperan deberían ingresar en el orden en que arribaron? Cómo modificaría la solución para considerar además esta restricción.