	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación
Salas A y B

Profesor: Ing. José Antonio Ayala Barbosa

Asignatura: Programación Orientada a Objetos

Grupo: 1

No de Práctica(s): Práctica 12 Hilos

Integrante(s): José Luis Arroyo Chavarría
Francisco Moisés Barrera Guardia
Juan Manuel Peralta Rodríguez
Rodrigo Daniel Reséndiz Cruz

No. de Lista o Brigada: Equipo I

Semestre: 3

Fecha de entrega: 21/1/2021

Observaciones:

CALIFICACIÓN: _____

I. Previo

La palabra reservada synchronized:

Se usa para indicar que ciertas partes del código, (habitualmente, una función miembro) están sincronizadas, es decir, que solamente un subproceso puede acceder a dicho método a la vez.

Cada método sincronizado posee una especie de llave que puede cerrar o abrir la puerta de acceso. Cuando un subproceso intenta acceder al método sincronizado mirará a ver si la llave está echada, en cuyo caso no podrá accederlo. Si método no tiene puesta la llave entonces el subproceso puede acceder a dicho código sincronizado.

II. Objetivo

Implementar el concepto de multitarea utilizando hilos en un lenguaje orientado a objetos.

III. Introducción

Al momento de realizar un proyecto nuevo en Java se busca ser lo más eficaz y ordenado posible, a la hora en la que se habla sobre hilos en Java nos referimos a otra forma de crear la posibilidad de concurrencia de actividades, es decir, los hilos comparten el código el acceso a algunos datos de forma similar a como un objeto tiene acceso a otros objetos.

En Java un hilo es un objeto con capacidad de correr en forma concurrente el método run(), en cierta manera es como tener dos “program counters” para un mismo código. Una diferencia con los procesos es que carece de sentido y no es posible en este enfoque hacer mutar un proceso con algo similar a exec().

En esta práctica se verá que son los hilos y como sirven los mismos, además, del tiempo en que estos mismos van respondiendo y cuál va a correr primero y así consecutivamente

IV. Desarrollo

En esta práctica se hicieron actividades que tiene que ver con el manejo de hilos en java, a continuación, se enlistan las actividades.

1. Creación de un hilo.

```java
package pool2;

public class POO12 {
    public static void main(String[] args) {
        System.out.println("1++++++++++");
        HiloTred hilo = new HiloTred("PRIMER HILO");
        hilo.start();
        new HiloTred ("Segundo Hilo").start();
        System.out.println("Termina el hilo principal");
```

Ilustración 1.-Còdigo Actividad 1 (main)

```java
package pool2;

public class HiloTred extends Thread{

    public HiloTred(String nombre) {
        super(nombre);
    }

    @Override
    public void run(){
        for (int i = 0; i < 10; i++) {
            System.out.println("Iteración "+(i)+" de "+get
        }
        System.out.println("Termina el hilo "+getName());
    }

}
```

Ilustración 2.-Código Actividad 2 (clase Hilothread)

En la primera actividad tenemos la elaboración de un hilo el cual se crea mediante una clase que extiende de thread, en dicha clase se utiliza el método run() para crear un hilo en ejecución. Una vez dentro de dicho método se imprimen las iteraciones de cada hilo existente.

En el main se van a crear dos objetos de tipo hilotred, los cuales van a ser impresos en consola para ver sus iteraciones.

```
1++++++++++
Termina el hilo principal
Iteración 0 de PRIMER HILO
Iteración 1 de PRIMER HILO
Iteración 2 de PRIMER HILO
Iteración 0 de Segundo Hilo
Iteración 1 de Segundo Hilo
Iteración 2 de Segundo Hilo
Iteración 3 de Segundo Hilo
Iteración 4 de Segundo Hilo
Iteración 5 de Segundo Hilo
Iteración 3 de PRIMER HILO
Iteración 6 de Segundo Hilo


Termina el hilo Segundo Hilo
Iteración 4 de PRIMER HILO
Iteración 5 de PRIMER HILO
Iteración 6 de PRIMER HILO
Iteración 7 de PRIMER HILO
Iteración 8 de PRIMER HILO
Iteración 9 de PRIMER HILO
Termina el hilo PRIMER HILO
```

Ilustración 3.- Ejecuciòn Actividad 1

Como se puede ver en la ejecución, los hilos comienzan a ejecutarse mediante iteraciones, pero no se ejecutan de forma secuencial, sino que lo hacen de forma paralela, por lo que se disputan una “carrera” para ver quien termina de concluir todas sus iteraciones primero.

2. ***Creación de un hilo con Runnable:***

```java
package pool2;

public class hiloRonabol implements Runnable{

    @Override
    public void run(){
        for (int i = 0; i < 10; i++) {
            System.out.println("Iteración "+i+" de"+Thread.currentThread().getName());
        }
        System.out.println("Termina el hilo "+Thread.currentThread().getName());
    }
}
```

Ilustración 4.- Código Actividad 2 (clase hiloRunnable)

```
System.out.println("2+++++++++++++++");
new Thread(new hiloRonabol(), "Primer Hilo").start();
new Thread(new hiloRonabol(), "Segundo Hilo").start();
System.out.println("Termina hilo principal");
```

Ilustración 5.-Código Actividad 2 (main)

En la segunda actividad se hizo algo parecido a la primara actividad, la única diferencia es que ahora implementamos la interface Runnable la cual se encarga de producir hilos funcionales para otras clases. Dentro de la clase hiloRunnable se implementa dicha interface proporciona un método run para que sea ejecutado por un objeto de tipo Thread creado, en el main se hacen dos hilos y se inician como en la anterior actividad.

```
2+++++++++++++++
Termina hilo principal
Iteración 0 dePrimer Hilo
Iteración 0 deSegundo Hilo
Iteración 1 dePrimer Hilo
Iteración 1 deSegundo Hilo
Iteración 2 dePrimer Hilo
Iteración 2 deSegundo Hilo
Iteración 3 dePrimer Hilo
Iteración 3 deSegundo Hilo
Iteración 4 dePrimer Hilo
Iteración 5 dePrimer Hilo
Iteración 6 dePrimer Hilo
Iteración 4 deSegundo Hilo

Iteración 7 dePrimer Hilo
Iteración 5 deSegundo Hilo
Iteración 8 dePrimer Hilo
Iteración 6 deSegundo Hilo
Iteración 9 dePrimer Hilo
Iteración 7 deSegundo Hilo
Termina el hilo Primer Hilo
Iteración 8 deSegundo Hilo
Iteración 9 deSegundo Hilo
Termina el hilo Segundo Hilo
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ilustración 6.- Ejecución Actividad 2

Como se puede visualizar en la ejecución, el programa hecho hace lo mismo que en la anterior actividad, pero con el detalle de utilizar la interface Runnable.

3. ***Implementación de un grupo de hilos con la clase GroupThread.***

```
package GrupoHilos;

public class ClaseMain {
    public static void main(String[] args) {

        ThreadGroup grupoHilos = new ThreadGroup("Grupo con prioridad maxima");

        Thread hilo1 = new Ghilos(grupoHilos, "Hilo 1 con prioridad máxima ");
        Thread hilo2 = new Ghilos(grupoHilos, "Hilo 2 con prioridad normal ");
        Thread hilo3 = new Ghilos(grupoHilos, "Hilo 3 con prioridad normal ");
        Thread hilo4 = new Ghilos(grupoHilos, "Hilo 4 con prioridad normal ");
        Thread hilo5 = new Ghilos(grupoHilos, "Hilo 5 con prioridad normal ");

        hilo1.setPriority(Thread.MAX_PRIORITY);
        grupoHilos.setMaxPriority(Thread.NORM_PRIORITY);
        System.out.println("Maxima prioridad del grupo "+ grupoHilos.getMaxPriority())

        System.out.println("Prioridad hilo 1 = "+ hilo1.getPriority());
        System.out.println("Prioridad hilo 2 = "+ hilo2.getPriority());
        System.out.println("Prioridad hilo 3 = "+ hilo3.getPriority());
        System.out.println("Prioridad hilo 4 = "+ hilo4.getPriority());
        System.out.println("Prioridad hilo 5 = "+ hilo5.getPriority());

        hilo1.start();
        hilo2.start();
        hilo3.start();
        hilo4.start();
        hilo5.start();

        listaHilos(grupoHilos);
    }
```

```
public static void listaHilos(ThreadGroup grupoActual){
    int numHilos;
    Thread[] listaDeHilos;
    numHilos = grupoActual.activeCount();
    listaDeHilos = new Thread[numHilos];
    grupoActual.enumerate(listaDeHilos);
    System.out.println("Numero de hilos activos"+numHilos);
    for (int i = 0; i < numHilos; i++) {
        System.out.println("Hilo Activo"+i+"="+listaDeHilos[i].getName());
    }
}
```

Ilustración 7.-Código Actividad 3 (main)

```
6   package GrupoHilos;
7   public class Ghilos extends Thread{
8
9       public Ghilos(ThreadGroup g, String n ) {
10          super(g, n);
11      }
12
14      public void run(){
15          for (int i = 0; i < 10; i++) {
16              System.out.println(getName()+"iteracion "+i);
            }
```

Ilustración 8.-Código Actividad 3 (Clase grupoHilos)

En la tercera actividad hicimos un grupo de 5 hilos cuyo primer hilo tendría una prioridad mayor a los demás hilos, primero hicimos la clase grupoHilos que extiende de Thread y se utiliza un constructor cuyos valores de entrada son el grupo de hilos y el nombre asignado a cada uno de ellos, luego el proceso es el mismo que los anterior haciendo uso del método run().

En la clase principal se crean todos los hilos y se establece una prioridad máxima con valor de 10 para el primer hilo, y los demás con prioridades normales cuyo valor es 5, dichas prioridades se imprimen para validar lo anterior y se ponen a correr todos los hilos y finalmente mandamos a llamar al método ListaHilos para contar el número de hilos activos.

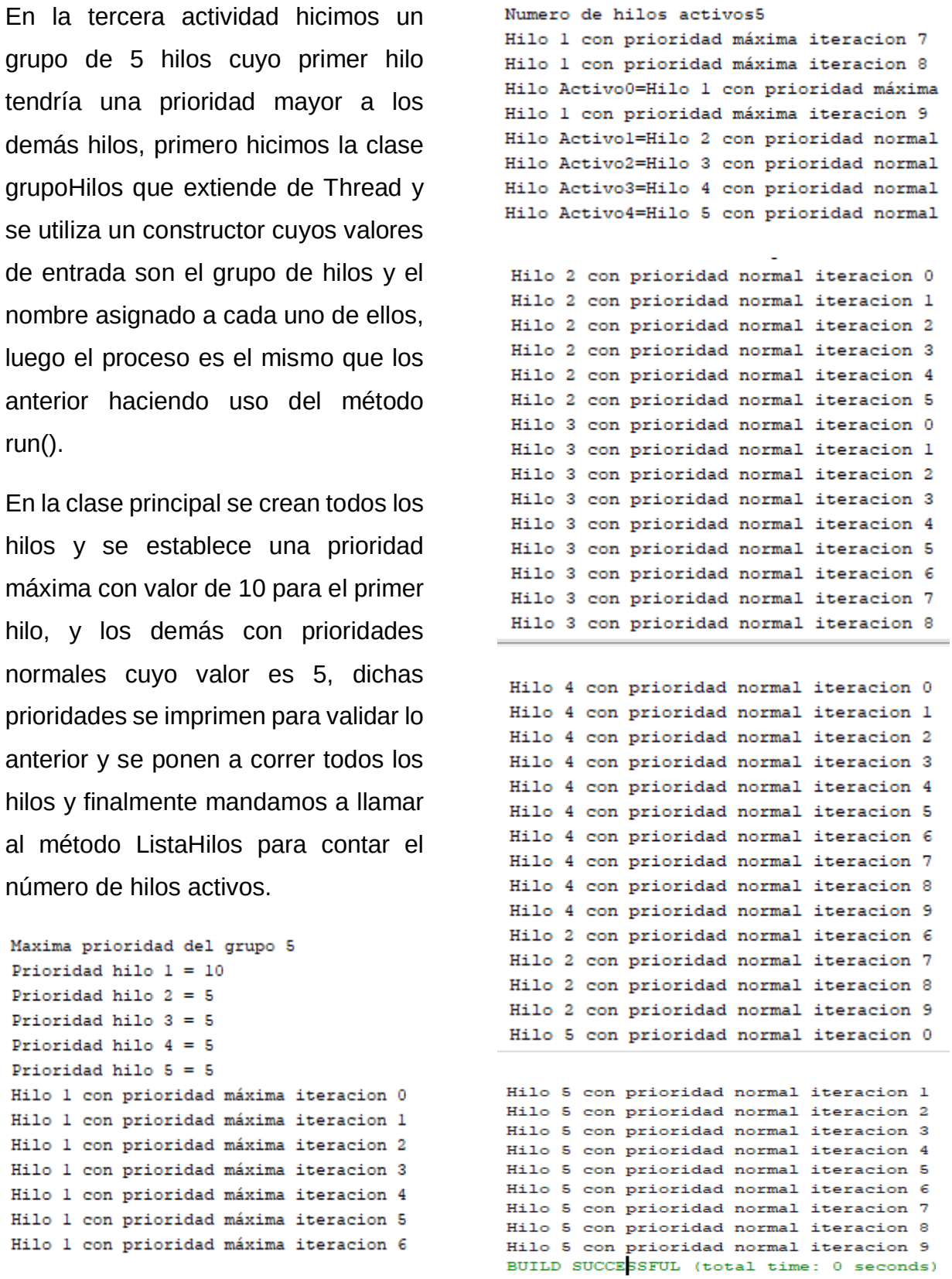


Ilustración 9.- Ejecución Actividad 3

En la ejecución se ve que el hilo 1 no se ejecuta a la par de los demás hilos, esto es porque, al darle mayor prioridad, nos aseguramos que dicho hilo se ejecutará primero. En el caso de los demás hilos, como tienen la misma prioridad (media) se ejecutan de forma paralela.

Estas fueron todas las actividades abarcadas en la práctica de Hilos.

V. Código fuente

❖ *Hilos 1*

```
package poop12;
public class Hilos {
    public static void main(String[] args)
{
        HiloTheread hilo = new
HiloTheread("Primer Hilo");
        hilo.start();
        new HiloTheread("Segundo
hilo").start();
        System.out.println("Termina el
hilo principal");
    }
}
```

❖ *HiloTheread*

```
package poop12;
public class HiloTheread extends
Thread{
    public HiloTheread(String nombre) {
        super(nombre);
    }
    @Override
    public void run(){
        for(int i=0; i<10; i++){
            System.out.println("Interación
"+(i)+" de " + getName() );
        }
        System.out.println("Termina el
hilo " + getName());
    }
}
```

```
package poop12;

public class Hilos {
    public static void main(String[] args) {

        HiloTheread hilo = new HiloTheread("Primer Hilo");
        hilo.start();
        new HiloTheread("Segundo hilo").start();
        System.out.println("Termina el hilo principal");
    }
}
```

```
package poop12;

public class HiloTheread extends Thread{
    public HiloTheread(String nombre) {
        super(nombre);
    }

    @Override
    public void run(){
        for(int i=0; i<10; i++){
            System.out.println("Interación "+(i)+" de " + getName() );
        }
        System.out.println("Termina el hilo " + getName());
    }
}
```

```
run:
Termina el hilo principal
Interación 0 de Primer Hilo
Interación 1 de Primer Hilo
Interación 2 de Primer Hilo
Interación 3 de Primer Hilo
Interación 4 de Primer Hilo
Interación 5 de Primer Hilo
Interación 6 de Primer Hilo
Interación 7 de Primer Hilo
Interación 8 de Primer Hilo
Interación 9 de Primer Hilo
Termina el hilo Primer Hilo
Interación 0 de Segundo hilo
Interación 1 de Segundo hilo
Interación 2 de Segundo hilo
Interación 3 de Segundo hilo
Interación 4 de Segundo hilo
Interación 5 de Segundo hilo
Interación 6 de Segundo hilo
Interación 7 de Segundo hilo
Interación 8 de Segundo hilo
Interación 9 de Segundo hilo
Termina el hilo Segundo hilo
BUILD SUCCESSFUL (total time: 0 seconds)
```

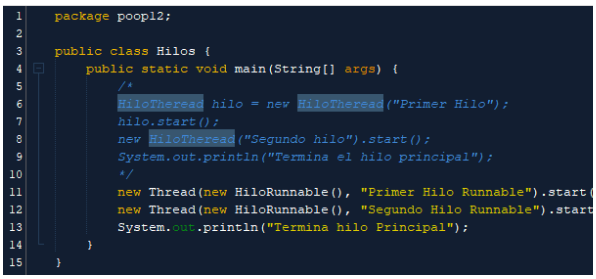
❖ *Hilos 2*

```
package poop12;
public class Hilos {
```

```
    public static void main(String[] args)
{
        /*
        HiloTheread hilo = new
HiloTheread("Primer Hilo");
        hilo.start();
        new HiloTheread("Segundo
hilo").start();
        System.out.println("Termina el
hilo principal");
        */
        new Thread(new HiloRunnable(),
"Primer Hilo Runnable").start();
        new Thread(new HiloRunnable(),
"Segundo Hilo Runnable").start();
        System.out.println("Termina hilo
Principal");
    }
}
```

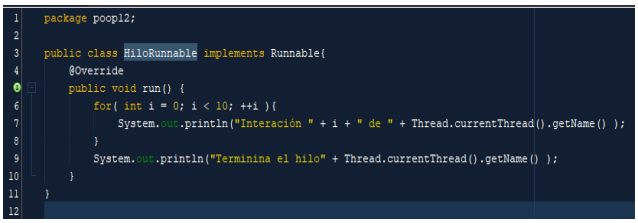
❖ *HiloRunnable*

```
package poop12;
public class HiloRunnable implements
Runnable{
    @Override
    public void run() {
        for( int i = 0; i < 10; ++i ){
            System.out.println("Interación
" + i + " de " +
Thread.currentThread().getName() );
        }
        System.out.println("Terminina el
hilo" +
Thread.currentThread().getName() );
    }
}
```



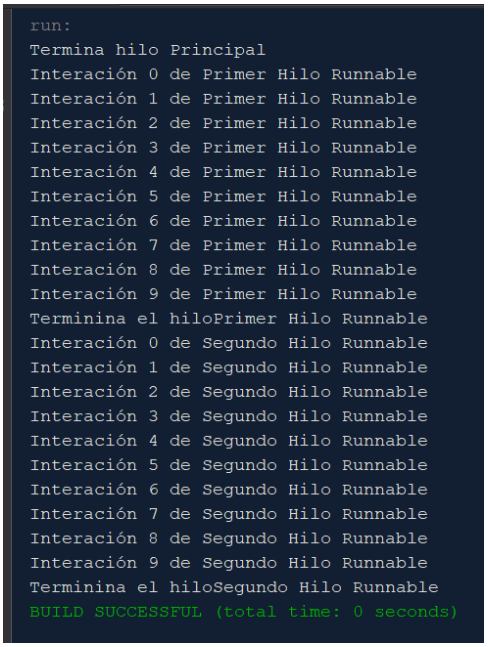
```
package poop12;

public class Hilos {
    public static void main(String[] args) {
        /*
        HiloTheread hilo = new HiloTheread("Primer Hilo");
        hilo.start();
        new HiloTheread("Segundo hilo").start();
        System.out.println("Termina el hilo principal");
        */
        new Thread(new HiloRunnable(), "Primer Hilo Runnable").start(
        new Thread(new HiloRunnable(), "Segundo Hilo Runnable").start
        System.out.println("Termina hilo Principal");
    }
}
```



```
package poop12;

public class HiloRunnable implements Runnable{
    @Override
    public void run() {
        for( int i = 0; i < 10; ++i ){
            System.out.println("Interación " + i + " de " + Thread.currentThread().getName() );
        }
        System.out.println("Terminina el hilo" + Thread.currentThread().getName() );
    }
}
```



```
run:
Termina hilo Principal
Interación 0 de Primer Hilo Runnable
Interación 1 de Primer Hilo Runnable
Interación 2 de Primer Hilo Runnable
Interación 3 de Primer Hilo Runnable
Interación 4 de Primer Hilo Runnable
Interación 5 de Primer Hilo Runnable
Interación 6 de Primer Hilo Runnable
Interación 7 de Primer Hilo Runnable
Interación 8 de Primer Hilo Runnable
Interación 9 de Primer Hilo Runnable
Terminina el hiloPrimer Hilo Runnable
Interación 0 de Segundo Hilo Runnable
Interación 1 de Segundo Hilo Runnable
Interación 2 de Segundo Hilo Runnable
Interación 3 de Segundo Hilo Runnable
Interación 4 de Segundo Hilo Runnable
Interación 5 de Segundo Hilo Runnable
Interación 6 de Segundo Hilo Runnable
Interación 7 de Segundo Hilo Runnable
Interación 8 de Segundo Hilo Runnable
Interación 9 de Segundo Hilo Runnable
Terminina el hiloSegundo Hilo Runnable
BUILD SUCCESSFUL (total time: 0 seconds)
```

❖ *grupoHilos*

```
package GrupoHilos;
public class grupoHilos extends
Thread{
    public grupoHilos(ThreadGroup g,
String n){
        super(g,n);
    }
    @Override
    public void run(){
        for( int i = 0; i < 10; ++i ){
            System.out.println( getName()
+ " iteración " + i );
        }
    }
}
```

❖ *ClaseMain*

```
package GrupoHilos;
public class ClaseMain {
```

```
    public static void main(String[] args)
{
        ThreadGroup grupohilos = new
ThreadGroup("Grupo con prioridad
normal");
        Thread hilo1 = new grupoHilos(
grupohilos, "Hilo 1 con prioridad
máxima");
        Thread hilo2 = new grupoHilos(
grupohilos, "Hilo 2 con prioridad
normal");
        Thread hilo3 = new grupoHilos(
grupohilos, "Hilo 3 con prioridad
normal");
        Thread hilo4 = new grupoHilos(
grupohilos, "Hilo 4 con prioridad
normal");
        Thread hilo5 = new grupoHilos(
grupohilos, "Hilo 5 con prioridad
normal");
hilo1.setPriority(Thread.MAX_PRIORI
TY);
grupohilos.setMaxPriority(Thread.NO
RM_PRIORITY);
        System.out.println("Maxima
prioridad del grupo " +
grupohilos.getMaxPriority() );
        System.out.println("Prioridad Hilo
1 = " + hilo1.getPriority());
        System.out.println("Prioridad Hilo
2 = " + hilo2.getPriority());
        System.out.println("Prioridad Hilo
3 = " + hilo3.getPriority());
        System.out.println("Prioridad Hilo
4 = " + hilo4.getPriority());
        System.out.println("Prioridad Hilo
5 = " + hilo5.getPriority());
        System.out.println("\n\n\n");
        hilo1.start();
        hilo2.start();
        hilo3.start();
        hilo4.start();
        hilo5.start();
        listaHilos(grupohilos);
    }
    public static void
listaHilos(ThreadGroup grupoActual){
        int numHilos;
        Thread[] listaDeHilos;
        numHilos =
grupoActual.activeCount();
        listaDeHilos = new
Thread[numHilos];
grupoActual.enumerate(listaDeHilos);
        System.out.println("Num Hilos
activos: " + numHilos);
        for(int i = 0; i < numHilos; ++i){
            System.out.println("Hilo activo
" +i+ " = "+ listaDeHilos[i].getName());
        }
    }
}
```

```
package GrupoHilos;

public class grupoHilos extends Thread{
    public grupoHilos(ThreadGroup g, String n){
        super(g,n);
    }

    @Override
    public void run(){
        for( int i = 0; i < 10; ++i ){
            System.out.println( getName() + " iteración " + i );
        }
    }
}
```

```
package GrupoHilos;

public class ClaseMain {

    public static void main(String[] args) {
        ThreadGroup grupohilos = new ThreadGroup("Grupo con prioridad normal");
        Thread hilo1 = new grupoHilos( grupohilos, "Hilo 1 con prioridad máxima");
        Thread hilo2 = new grupoHilos( grupohilos, "Hilo 2 con prioridad normal");
        Thread hilo3 = new grupoHilos( grupohilos, "Hilo 3 con prioridad normal");
        Thread hilo4 = new grupoHilos( grupohilos, "Hilo 4 con prioridad normal");
        Thread hilo5 = new grupoHilos( grupohilos, "Hilo 5 con prioridad normal");

        hilo1.setPriority(Thread.MAX_PRIORITY);
        grupohilos.setMaxPriority(Thread.NORM_PRIORITY);
        System.out.println("Maxima prioridad del grupo " + grupohilos.getMaxPriority() );

        System.out.println("Prioridad Hilo 1 = " + hilo1.getPriority());
        System.out.println("Prioridad Hilo 2 = " + hilo2.getPriority());
        System.out.println("Prioridad Hilo 3 = " + hilo3.getPriority());
        System.out.println("Prioridad Hilo 4 = " + hilo4.getPriority());
        System.out.println("Prioridad Hilo 5 = " + hilo5.getPriority());
        System.out.println("\n\n\n");

        hilo1.start();
        hilo2.start();
        hilo3.start();
        hilo4.start();
        hilo5.start();

        listaHilos(grupohilos);
    }

    public static void listaHilos(ThreadGroup grupoActual){
        int numHilos;
        Thread[] listaDeHilos;
        numHilos = grupoActual.activeCount();
        listaDeHilos = new Thread[numHilos];
        grupoActual.enumerate(listaDeHilos);
        System.out.println("Num Hilos activos: " + numHilos);
        for(int i = 0; i < numHilos; ++i){
            System.out.println("Hilo activo " +i+ " = "+ listaDeHilos[i].getName());
        }
    }
}
```

```
run:
Maxima prioridad del grupo 5
Prioridad Hilo 1 = 10
Prioridad Hilo 2 = 5
Prioridad Hilo 3 = 5
Prioridad Hilo 4 = 5
Prioridad Hilo 5 = 5
```

```
Num Hilos activos: 5
Hilo activo 0 = Hilo 1 con prioridad máxima
Hilo activo 1 = Hilo 2 con prioridad normal
Hilo 1 con prioridad máxima iteración 0
Hilo 1 con prioridad máxima iteración 1
Hilo 1 con prioridad máxima iteración 2
Hilo 1 con prioridad máxima iteración 3
Hilo 1 con prioridad máxima iteración 4
Hilo 1 con prioridad máxima iteración 5
Hilo 1 con prioridad máxima iteración 6
Hilo 1 con prioridad máxima iteración 7
Hilo 1 con prioridad máxima iteración 8
Hilo 1 con prioridad máxima iteración 9
Hilo activo 2 = Hilo 3 con prioridad normal
Hilo activo 3 = Hilo 4 con prioridad normal
Hilo activo 4 = Hilo 5 con prioridad normal
Hilo 4 con prioridad normal iteración 0
Hilo 4 con prioridad normal iteración 1
Hilo 4 con prioridad normal iteración 2
Hilo 4 con prioridad normal iteración 3
Hilo 4 con prioridad normal iteración 4
```

```
Hilo 4 con prioridad normal iteración 5
Hilo 4 con prioridad normal iteración 6
Hilo 4 con prioridad normal iteración 7
Hilo 4 con prioridad normal iteración 8
Hilo 4 con prioridad normal iteración 9
Hilo 5 con prioridad normal iteración 0
Hilo 5 con prioridad normal iteración 1
Hilo 5 con prioridad normal iteración 2
Hilo 5 con prioridad normal iteración 3
Hilo 5 con prioridad normal iteración 4
Hilo 5 con prioridad normal iteración 5
Hilo 5 con prioridad normal iteración 6
Hilo 5 con prioridad normal iteración 7
Hilo 5 con prioridad normal iteración 8
Hilo 5 con prioridad normal iteración 9
Hilo 2 con prioridad normal iteración 0
Hilo 2 con prioridad normal iteración 1
Hilo 2 con prioridad normal iteración 2
Hilo 2 con prioridad normal iteración 3
Hilo 2 con prioridad normal iteración 4
Hilo 2 con prioridad normal iteración 5
Hilo 2 con prioridad normal iteración 6
Hilo 2 con prioridad normal iteración 7
Hilo 2 con prioridad normal iteración 8
Hilo 2 con prioridad normal iteración 9
Hilo 3 con prioridad normal iteración 0
Hilo 3 con prioridad normal iteración 1
Hilo 3 con prioridad normal iteración 2
Hilo 3 con prioridad normal iteración 3
Hilo 3 con prioridad normal iteración 4
Hilo 3 con prioridad normal iteración 5
Hilo 3 con prioridad normal iteración 6
```

```
Hilo 3 con prioridad normal iteración 7
Hilo 3 con prioridad normal iteración 8
Hilo 3 con prioridad normal iteración 9
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Cuenta (Trabajo extra)

```
package Cuenta;
public class cuenta extends Thread{
    private static long saldo = 0;
    public cuenta (String nombre){
        super(nombre);
    }
    @Override
    public void run(){
        if (getName().equals("Deposito 1
") ||
                getName().equals
("Deposito 2 ")){
            this.depositarDinero(100);
        }else{
            this.extraerDinero(50);
        }
        System.out.println("Termina el " +
getName());
    }
    public synchronized void
depositarDinero(int cantidad){
        saldo += cantidad;
        System.out.println ("Se
depositaron " + cantidad + " pesos ");
            notifyAll();
    }
    public synchronized void
extraerDinero(int cantidad) {
        try {
            if (saldo <= 0){
System.out.println(getName()    +    "
espera deposito "
                    + "\nSaldo = " + saldo);
                sleep(5000);
            }
        } catch(InterruptedException e){
            System.out.println(e);
        }
        saldo -= cantidad;
```

```
        System.out.println(getName() + "
extrajo " + cantidad +
                " pesos.\nSaldo restante =
"+ saldo);
                notifyAll();
    }
}
```

❖ ClaseMainCuenta

```
package Cuenta;
public class ClaseMainCuenta {
   public static void main (String [] args)
{
        new cuenta("Acceso 1: ").start();
        new cuenta("Acceso 2: ").start();
        new        cuenta("Deposito        1:
").start();
        new        cuenta("Deposito        2:
").start();
        System.out.println ("Termina el
hilo principal");
    }
}
```

```
package Cuenta;

public class cuenta extends Thread{
    private static long saldo = 0;
    public cuenta (String nombre){
        super(nombre);
    }
    @Override
    public void run(){
        if (getName().equals("Deposito 1 ") ||
                getName().equals ("Deposito 2 ")){
            this.depositarDinero(100);
        }else{
            this.extraerDinero(50);
        }
        System.out.println("Termina el " + getName());
    }
    public synchronized void depositarDinero(int cantidad){
        saldo += cantidad;
        System.out.println ("Se depositaron " + cantidad + " pesos ");
            notifyAll();
    }
    public synchronized void extraerDinero(int cantidad) {
        try {
            if (saldo <= 0){
                System.out.println(getName() + " espera deposito "
                    + "\nSaldo = " + saldo);
                sleep(5000);
            }
        } catch(InterruptedException e){
            System.out.println(e);
        }
        saldo -= cantidad;
        System.out.println(getName() + " extrajo " + cantidad +
                " pesos.\nSaldo restante = "+ saldo);
                notifyAll();
    }
}
```

```
package Cuenta;

public class ClaseMainCuenta {
    public static void main (String [] args) {
        new cuenta("Acceso 1: ").start();
        new cuenta("Acceso 2: ").start();
        new cuenta("Deposito 1: ").start();
        new cuenta("Deposito 2: ").start();
        System.out.println ("Termina el hilo principal");
    }
}
```

```
run:
Termina el hilo principal
Acceso 1:  espera deposito
Saldo = 0
Deposito 2:  espera deposito
Saldo = 0
Deposito 1:  espera deposito
Saldo = 0
Acceso 2:  espera deposito
Saldo = 0
Deposito 2:  extrajo 50 pesos.
Saldo restante = -150
Termina el Deposito 2:
Acceso 1:  extrajo 50 pesos.
Saldo restante = -150
Acceso 2:  extrajo 50 pesos.
Saldo restante = -150
Termina el Acceso 2:
Deposito 1:  extrajo 50 pesos.
Saldo restante = -150
Termina el Deposito 1:
Termina el Acceso 1:
BUILD SUCCESSFUL (total time: 5 seconds)
```

VI. Conclusión

- José Luis Arroyo Chavarría:

Durante esta práctica al realizar los ejercicios me pareció un poco extraño la manera de cómo se ejecutaban los hilos dependiendo de los diversos usos a los que usuario desea. Siento que al profundizar esta práctica veremos varias posibilidades de su uso.

- Francisco Moisés Barrera Guardia:

Esta práctica, se pudo apreciar el cómo funcionan los hilos, además de que se aprendió el cómo se programan los mismos en java, junto a todos los temas que se fueron viendo a lo largo del código y se pudieron reafirmar

- Juan Manuel Peralta Rodríguez:

Gracias al desarrollo de esta práctica se pudo revisar a profundidad los diferentes usos de los hilos en el lenguaje de programación Java, así como su aplicación para la resolución de diferentes problemas cotidianos, de igual forma se pudieron cumplir los objetivos previamente propuestos dúrate el desarrollo de las actividades.

- Rodrigo Daniel Reséndiz Cruz:

En esta práctica pudimos ver cómo manejar hilos por lo que nos ayudará a poder formar nuestro sistema de aparatos de ambiente de nuestro proyecto final. Además, de esta práctica surge la curiosidad de ver más aplicaciones del contenido revisado.

VII. Referencias

- http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/threads/sincronizacion.htm#:~:text=La%20palabra%20reservada%20synchronized%20se,dicho%20m%C3%A9todo%20a%20la%20vez.&text=Si%20m%C3%A9todo%20no%20tiene%20puesta,acceder%20a%20dicho%20c%C3%B3digo%20sincronizado.
- http://profesores.elo.utfsm.cl/~agv/elo330/2s10/lectures/Java/threads/JavaThreads.html#:~:text=En%20Java%20un%20hilo%20es,algo%20similar%20a%20exec().