**ACTIVIDAD 1**

**Enunciado**

Realiza el siguiente ejercicio utilizando una única clase Java (Hilo) que sustituya a las clases Primero y Segundo.

El nombre de la clase principal será Actividad1 y el número de veces que se sacará por la consola las cadenas de caracteres “Primero” y “Segundo” será 20.

**//Hilos1.java**

**public** **class** Hilos1 {

**public** **static** **void** main(String arg[]) {

Primero p=**new** Primero();

Segundo s=**new** Segundo();

p.start();

s.start();

System.***out***.print("Fin programa");

}

}

**//Primero.java**

**public class** Primero **extends** Thread {

**public** **void** run(){

**for** (**int** i=1;i<100;i++)

System.***out***.println("Primero " + i);

}

}

**//Segundo.java**

**public** **class** Segundo **extends** Thread {

**public** **void** run(){

**for** (**int** i=1;i<100;i++)

System.***out***.println("Segundo " + i);

}

}

**Solución**

**clase Hilo.java:**

Cass Hilo extends Thread {

// Mensaje que se imprimirá en la consola, puede ser "Primero" o "Segundo"

private final String mensaje;

// Constructor que inicializa el mensaje

public Hilo(String mensaje) {

this.mensaje = mensaje;

}

// Método que se ejecuta cuando se inicia el hilo

@Override

public void run() {

// Bucle que imprime el mensaje 20 veces

for (int i = 1; i <= 20; i++) {

System.out.println(mensaje + " " + i);

}

}

}

**Actividad1.java:**

public class Actividad1 {

public static void main(String[] args) {

// Creación de dos objetos de la clase Hilo, uno para imprimir "Primero" y otro para "Segundo"

Hilo hilo1 = new Hilo("Primero");

Hilo hilo2 = new Hilo("Segundo");

// Inicio de los hilos

hilo1.start();

hilo2.start();

try {

// El hilo principal espera a que hilo1 termine su ejecución

hilo1.join();

// El hilo principal espera a que hilo2 termine su ejecución

hilo2.join();

} catch (InterruptedException e) {

// Imprime el error en caso de una excepción

e.printStackTrace();

}

// Imprime "Fin programa" en la consola una vez que ambos hilos han terminado

System.out.print("Fin programa");

}

}

**Resolución de problemas**

**ACTIVIDAD 2**

**Enunciado**

Realiza un programa que cree n hilos, donde n se pasará como argumento.

Por cada hilo creado, se sacará por consola 20 veces un mensaje del tipo: “Hilo n”, donde n será el número de hilo que se está ejecutando.

Después de haber creado los n hilos, sacar por consola un mensaje que diga “Final Programa”.

El nombre de la clase principal será Actividad2.

**Solución**

**clase HiloAct2.java:**

//Clase Hilo que se encarga de imprimir el mensaje

class Hilo extends Thread {

private final int numeroHilo; // Número del hilo

// Constructor

public Hilo(int numeroHilo) {

this.numeroHilo = numeroHilo;

}

// Método que se ejecuta cuando el hilo empieza

@Override

public void run() {

for (int i = 0; i < 20; i++) {

System.out.println("Hilo " + numeroHilo); // Imprimir el mensaje

}

}

}

**Actividad2.java:**

import java.util.Scanner; // Importar la clase Scanner para leer la entrada del usuario

// Clase principal que crea y gestiona los hilos

public class Actividad2 {

public static void main(String[] args) {

// Crear un objeto Scanner para leer la entrada del usuario

Scanner scanner = new Scanner(System.in);

// Pedir al usuario que introduzca el número de hilos

System.out.print("Introduce el número de hilos: ");

int n = scanner.nextInt(); // Leer el número de hilos

// Array para almacenar los hilos

Thread[] hilos = new Thread[n];

// Crear y empezar los hilos

for (int i = 0; i < n; i++) {

hilos[i] = new Hilo(i + 1); // Crear un nuevo hilo

hilos[i].start(); // Empezar el hilo

}

// Esperar a que todos los hilos terminen

for (int i = 0; i < n; i++) {

try {

hilos[i].join(); // Esperar a que el hilo termine

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// Imprimir el mensaje final

System.out.println("Final Programa");

// Cerrar el objeto Scanner para liberar los recursos asociados

scanner.close();

}

}

**Resolución de problemas**

**ACTIVIDAD 3**

**Enunciado**

Realiza un programa que use los métodos setName() y setPriority() de la clase Thread para cambiar el nombre y prioridad iniciales del hilo. Además, será necesario utilizar los métodos getName() y getPriority() de la clase Thread para visualizar en consola, tanto los valores iniciales del nombre y prioridad del hilo como los valores posteriormente modificados.

El nombre de la clase principal será Actividad3.

El resultado esperado será el siguiente:

El nombre del hilo es Thread-0 y tiene la prioridad 5

Ahora el nombre del hilo es SUPER-HILO-DM2 y tiene la prioridad 6

Final programa

**Solución**

**Actividad3.java:**

package ejer3;

// Clase principal que crea y gestiona los hilos

public class Actividad3 {

public static void main(String[] args) {

// Crear un hilo

Thread hilo = new Thread(() -> {

// Imprimir el nombre y la prioridad inicial del hilo

System.out.println("El nombre del hilo es " + Thread.currentThread().getName() +

" y tiene la prioridad " + Thread.currentThread().getPriority());

// Cambiar el nombre y la prioridad del hilo

Thread.currentThread().setName("SUPER-HILO-DM2");

Thread.currentThread().setPriority(6);

// Imprimir el nombre y la prioridad modificados del hilo

System.out.println("Ahora el nombre del hilo es " + Thread.currentThread().getName() +

" y tiene la prioridad " + Thread.currentThread().getPriority());

});

// Iniciar el hilo

hilo.start();

try {

// Esperar a que el hilo termine

hilo.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

// Imprimir el mensaje final

System.out.println("Final programa");

}

}

**Resolución de problemas**

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Correcto funcionamiento de los programas desarrollados en Java (35%).
3. Ajuste a los resultados esperados (35%).
4. Claridad del código Java (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida). Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios. La primera línea de comentarios sólo incluirá lo siguiente: // Resolución de problemas.
6. Variables no usadas (-15%).
7. Librerías no usadas (-15%).