**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**

**La clase Actividad1:**public class Actividad1 {

public static void main(String[] args) {

Hilo hilo1 = new Hilo("Primero");

Hilo hilo2 = new Hilo("Segundo");

hilo1.start();

hilo2.start();

try {

hilo1.join();

hilo2.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

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

}

}

**La clase Hilo:**

class Hilo extends Thread {

private String mensaje;

/\*\*

\* metodo para instanciar los dos hilos

\* **@param** mensaje

\*/

public Hilo(String mensaje) {

this.mensaje = mensaje;

}

public void run() {

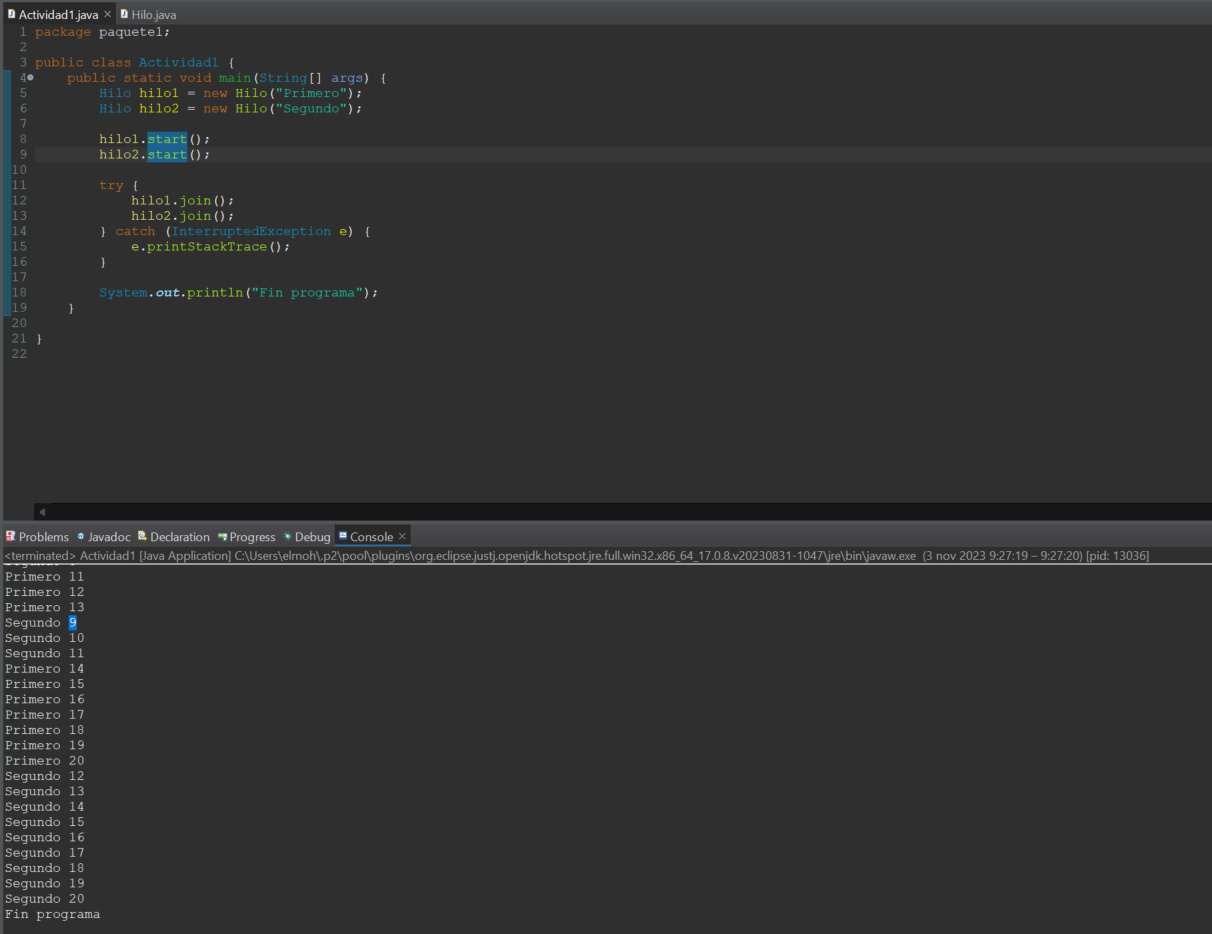
for (int i = 1; i <= 20; i++) {

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

}

}

}

****

**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**

**La clase Actividad2:**

package p1;

public class Actividad2 {

public static void main(String[] args) {

if (args.length != 1) {

System.***out***.println("Uso: java Actividad2 <número de hilos>");

return;

}

int n = Integer.*parseInt*(args[0]);

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

Thread hilo = new Hilo(i);

hilo.start();

}

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

}

}

**La clase Hilo:**

package p1;

class Hilo extends Thread {

private int numeroHilo;

public Hilo(int numeroHilo) {

this.numeroHilo = numeroHilo;

}

public void run() {

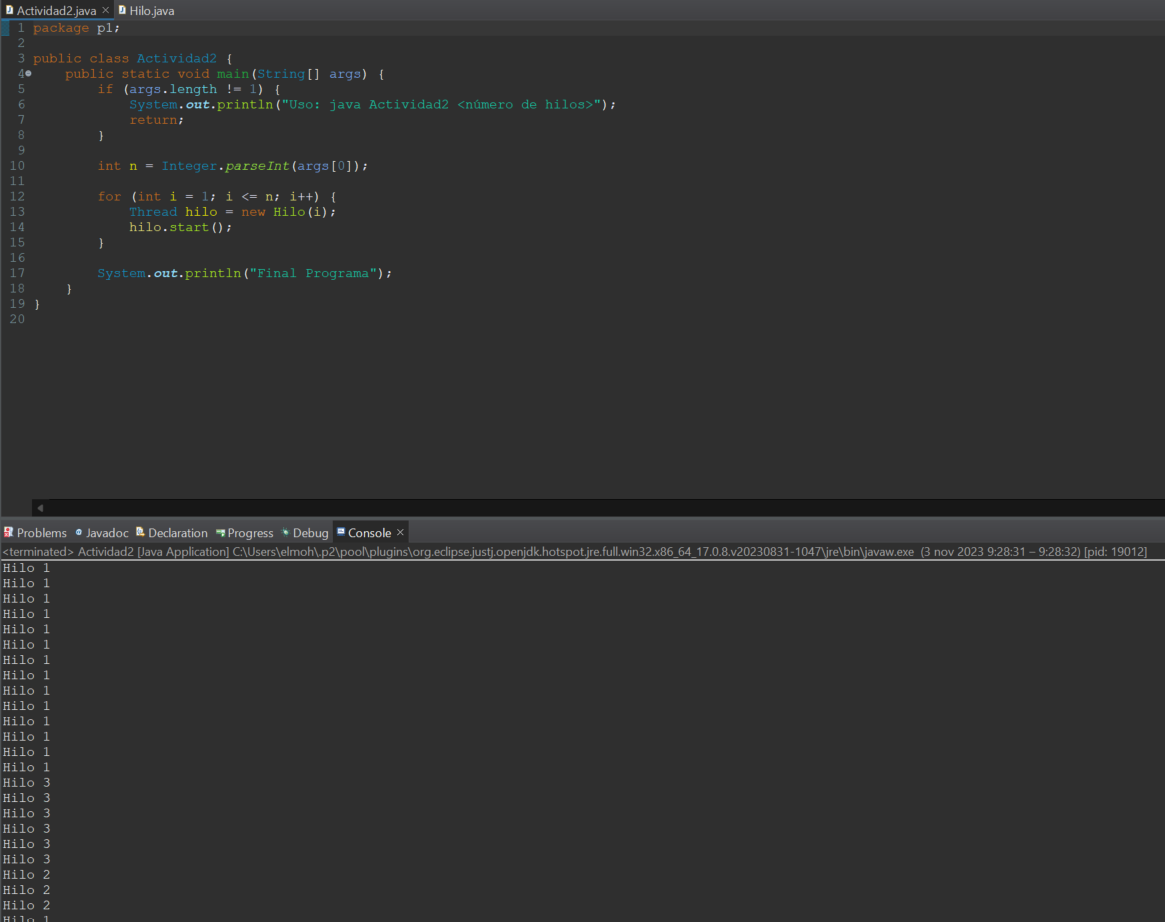
for (int i = 1; i <= 20; i++) {

System.***out***.println("Hilo " + numeroHilo);

}

}

}

****

**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**

package p1;

public class Actividad3 {

public static void main(String[] args) {

Thread hilo = Thread.*currentThread*(); // Obtener el hilo actual

// Mostrar los valores iniciales

System.***out***.println("El nombre del hilo es " + hilo.getName() + " y tiene la prioridad " + hilo.getPriority());

// Cambiar el nombre y la prioridad

hilo.setName("SUPER-HILO-DM2");

hilo.setPriority(6);

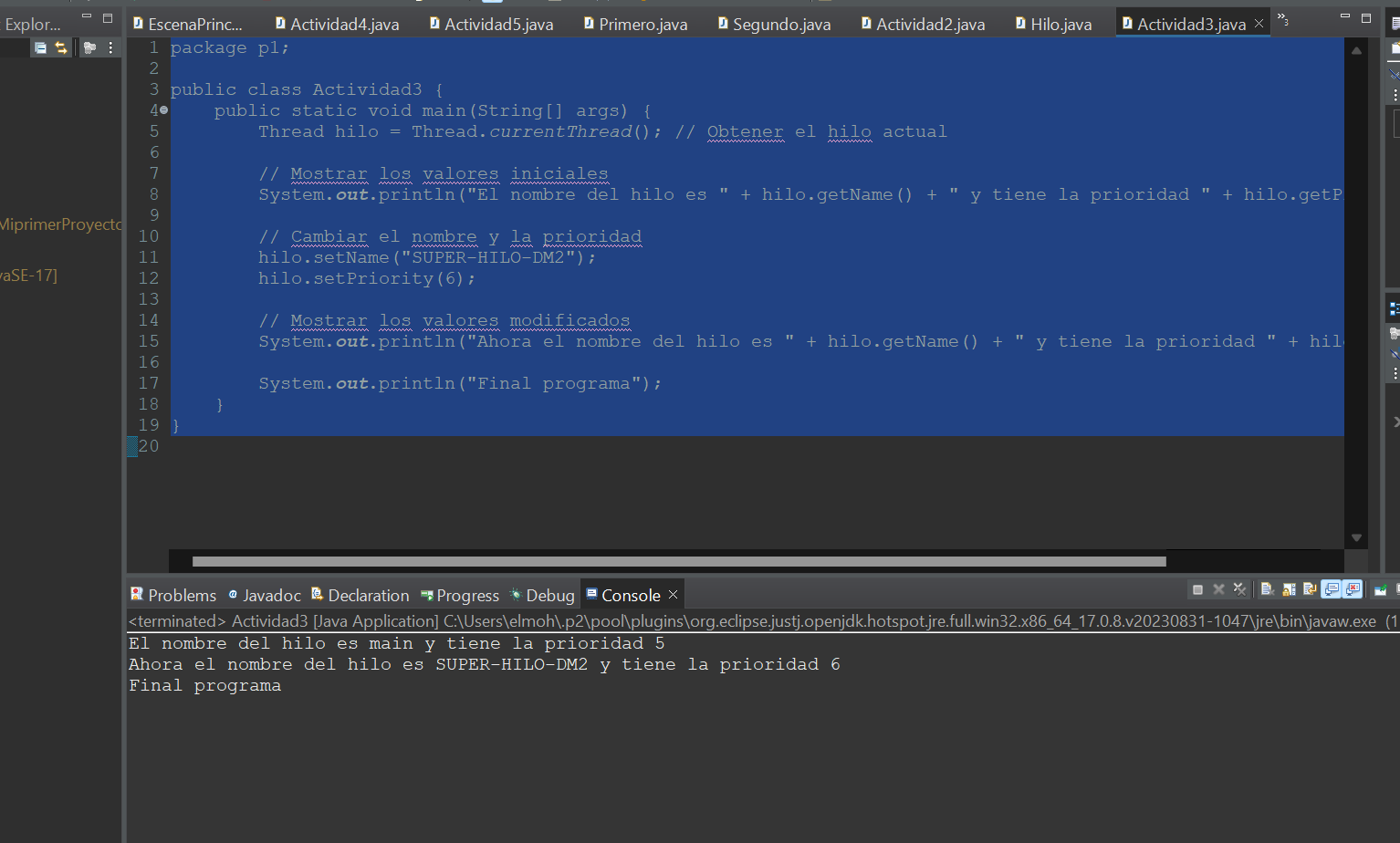
// Mostrar los valores modificados

System.***out***.println("Ahora el nombre del hilo es " + hilo.getName() + " y tiene la prioridad " + hilo.getPriority());

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%).