Contenido

[**Sección 25: Threads: Hilos y Concurrencia** 1](#_Toc140046165)

[257. Introducción a los Threads 1](#_Toc140046166)

[258. Creando hilos heredando de la clase Thread 2](#_Toc140046167)

[259. Creando hilos implementando la interface Runnable 5](#_Toc140046168)

[260. Implementando hilos con expresiones lambda y Runnable 5](#_Toc140046169)

[261. El método join vs sleep 5](#_Toc140046170)

[262. Usando modificador synchronized para la sincronización de hilos 5](#_Toc140046171)

[263. Métodos wait() y notify() sincronización Productor Consumidor 5](#_Toc140046172)

[264. Métodos wait() y notify() sincronización Productor Consumidor - parte 2 5](#_Toc140046173)

[265. Ejemplo de sincronización Productor Consumidor con Java 8 Lambda 5](#_Toc140046174)

[266. Usando Timer y TimerTask para agendar tareas programadas 5](#_Toc140046175)

[267. Agendando tareas programadas que se repiten cada cierto periodo de tiempo 5](#_Toc140046176)

[268. Executor Framework - Single Thread Executor 5](#_Toc140046177)

[269. Executor Framework - Future 5](#_Toc140046178)

[270. Executor Framework - Fixed Thread Pool 5](#_Toc140046179)

[271. Executor Framework – Personalizando parámetros del pool con ThreadPoolExecutor 5](#_Toc140046180)

[272. Executor Framework – Tareas programadas con ScheduledExecutorService 5](#_Toc140046181)

[273. Executor Framework – Tareas programadas que se repiten cada cierto… 5](#_Toc140046182)

[274. Descargar Código Fuente 5](#_Toc140046183)

[Tarea 28: Hilo Alfanumérico 6](#_Toc140046184)

# **Sección 25: Threads: Hilos y Concurrencia**

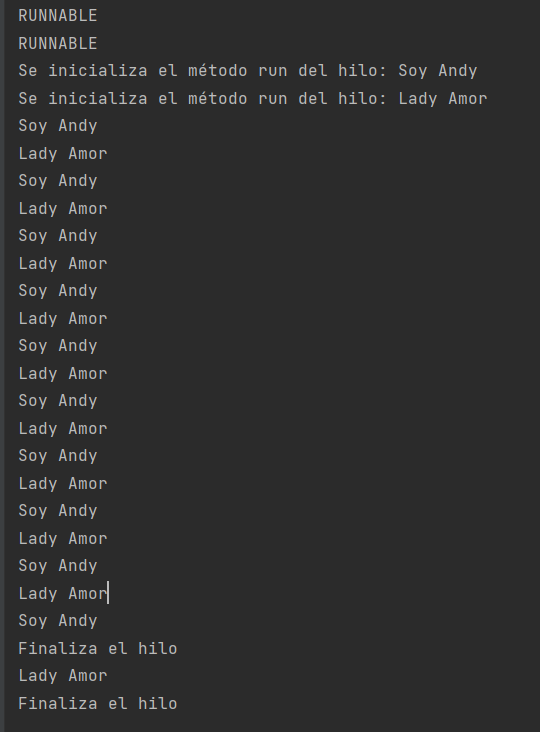
# 257. Introducción a los Threads

| Método | Descripción |
| --- | --- |
| start() | Inicia la ejecución del hilo, llamando al método run(). |
| run() | Define el código que se ejecutará en el hilo. |
| sleep(long ms) | Hace que el hilo actual se suspenda durante el número especificado de milisegundos. |
| join() | Hace que el hilo actual espere hasta que el hilo en el que se invoca haya terminado su ejecución. |
| interrupt() | Interrumpe el hilo actual, lanzando una excepción InterruptedException. |
| isAlive() | Verifica si el hilo está en ejecución. Devuelve true si el hilo aún está en ejecución; de lo contrario, false. |
| yield() | Hace que el hilo actual ceda el turno al siguiente hilo en la cola de planificación. |
| setPriority(int priority) | Establece la prioridad del hilo, un número entre Thread.MIN\_PRIORITY (1) y Thread.MAX\_PRIORITY (10). |
| getName() | Obtiene el nombre del hilo. |
| setName(String name) | Establece el nombre del hilo. |

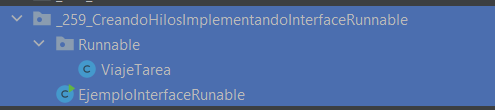
# 258. Creando hilos heredando de la clase Thread

package \_258\_CreandoHilosHeredandoClaseThreads.Threads;  
  
public class NombreThread extends Thread {  
 public NombreThread(String name) {  
 // Llamar al constructor de la clase Thread para asignar el nombre al hilo  
 super(name);  
 }  
  
 // Sobrescribir el método run que contiene el código a ejecutar en el hilo  
 @Override  
 public void run() {  
 System.*out*.println("Se inicializa el método run del hilo: " + getName());  
  
 // Ejecutar un bucle 10 veces  
 for (int i = 0; i < 10; i++) {  
 try {  
 // Pausar el hilo durante 10 milisegundos  
 Thread.*sleep*(10);  
 } catch (InterruptedException e) {  
 // Capturar la excepción InterruptedException y lanzar una nueva excepción RuntimeException  
 // con la excepción original como argumento para propagarla  
 throw new RuntimeException(e);  
 }  
  
 // Imprimir el nombre del hilo actual  
 System.*out*.println(this.getName());  
 }  
  
 System.*out*.println("Finaliza el hilo");  
 }  
}

package \_258\_CreandoHilosHeredandoClaseThreads;  
  
import \_258\_CreandoHilosHeredandoClaseThreads.Threads.NombreThread;  
  
public class EjemploExtenderThread {  
 public static void main(String[] args) throws InterruptedException {  
  
 // Crear una instancia de la clase NombreThread pasando el nombre del hilo como argumento  
 Thread hilo = new NombreThread("Lady Amor");  
  
 // Iniciar el hilo  
 hilo.start();  
 System.*out*.println(hilo.getState());  
  
 // Crear otra instancia de la clase NombreThread pasando el nombre del hilo como argumento  
 Thread hilo2 = new NombreThread("Soy Andy");  
 // Iniciar el segundo hilo  
 hilo2.start();  
 System.*out*.println(hilo.getState());  
 }  
}

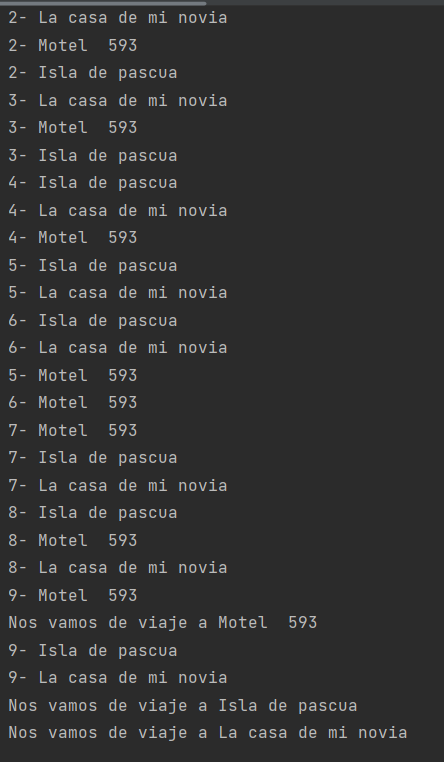


# 259. Creando hilos implementando la interface Runnable



package \_259\_CreandoHilosImplementandoInterfaceRunnable.Runnable;  
  
import java.util.Random;  
  
// Clase que implementa la interfaz Runnable para representar una tarea de viaje  
public class ViajeTarea implements Runnable {  
  
 private String nombre;  
  
 // Constructor para inicializar la tarea con un nombre  
 public ViajeTarea(String nombre) {  
 this.nombre = nombre;  
 }  
  
 // Método getter para obtener el nombre de la tarea de viaje  
 public String getNombre() {  
 return nombre;  
 }  
  
 @Override  
 public void run() {  
 // Método run() es obligatorio para implementar Runnable y se ejecutará en el hilo  
 for (int i = 0; i < 10; i++) {  
 System.*out*.println(i + "- " + nombre);  
 try {  
 // Simula una pausa aleatoria entre 0 y 1000 milisegundos  
 Thread.*sleep*((long) (Math.*random*() \* 1000));  
 } catch (InterruptedException e) {  
 // Manejo de excepción si se produce una interrupción durante la pausa  
 throw new RuntimeException(e);  
 }  
 }  
 System.*out*.println("Nos vamos de viaje a " + nombre);  
 }  
}

package \_259\_CreandoHilosImplementandoInterfaceRunnable;  
  
import \_259\_CreandoHilosImplementandoInterfaceRunnable.Runnable.ViajeTarea;  
  
// Clase principal que contiene el método main para iniciar los hilos de viaje  
public class EjemploInterfaceRunable {  
 public static void main(String[] args) throws InterruptedException {  
 // Creamos instancias de la clase ViajeTarea y las pasamos a nuevos hilos  
 new Thread(new ViajeTarea("Isla de Pascua")).start();  
 new Thread(new ViajeTarea("La casa de mi novia")).start();  
 new Thread(new ViajeTarea("Motel 593")).start();  
 }  
}



# 260. Implementando hilos con expresiones lambda y Runnable

# 261. El método join vs sleep

# 262. Usando modificador synchronized para la sincronización de hilos

# 263. Métodos wait() y notify() sincronización Productor Consumidor

# 264. Métodos wait() y notify() sincronización Productor Consumidor - parte 2

# 265. Ejemplo de sincronización Productor Consumidor con Java 8 Lambda

# 266. Usando Timer y TimerTask para agendar tareas programadas

# 267. Agendando tareas programadas que se repiten cada cierto periodo de tiempo

# 268. Executor Framework - Single Thread Executor

# 269. Executor Framework - Future

# 270. Executor Framework - Fixed Thread Pool

# 271. Executor Framework – Personalizando parámetros del pool con ThreadPoolExecutor

# 272. Executor Framework – Tareas programadas con ScheduledExecutorService

# 273. Executor Framework – Tareas programadas que se repiten cada cierto…

# 274. Descargar Código Fuente

# Tarea 28: Hilo Alfanumérico