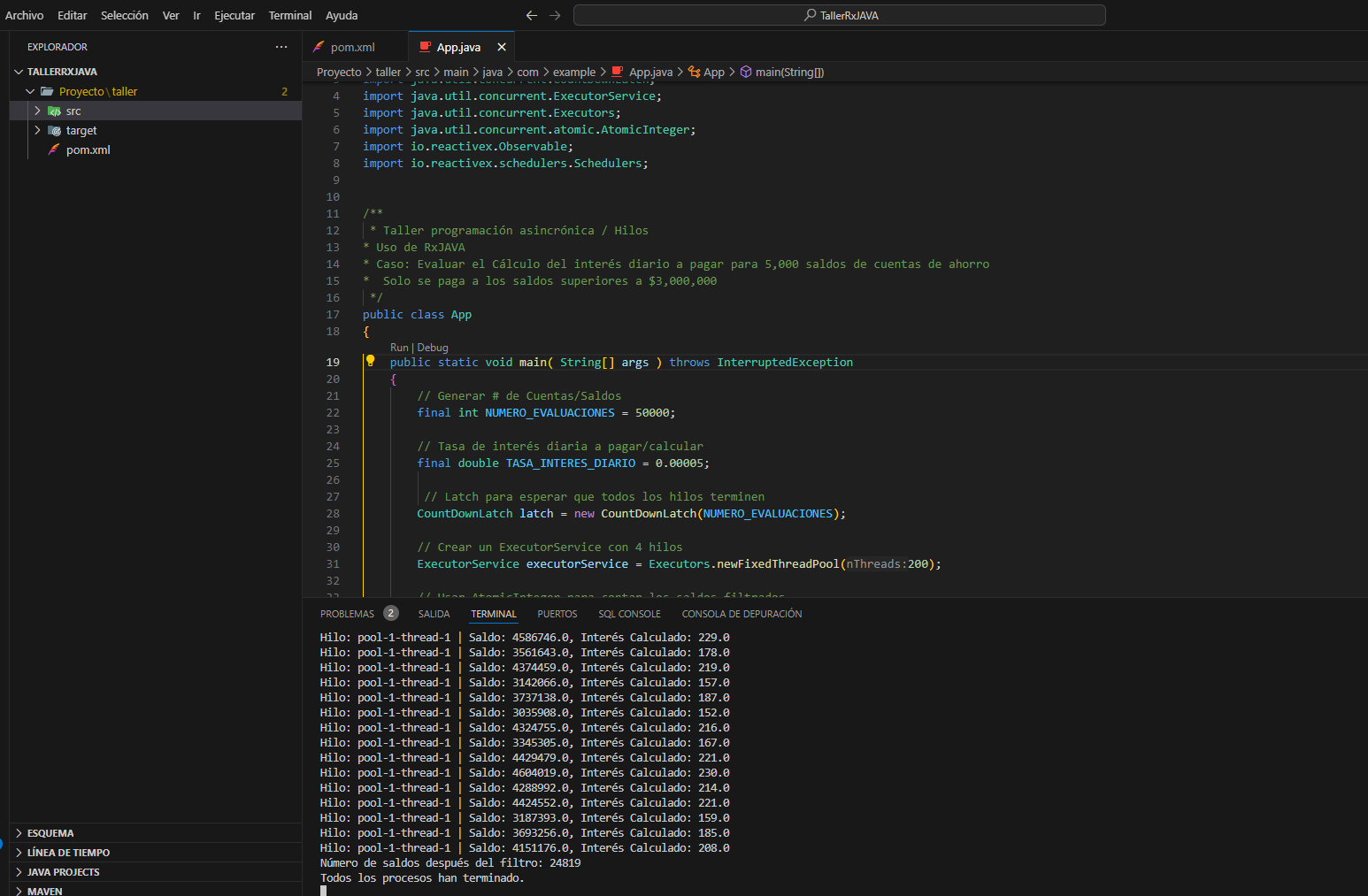
**TALLER RXJAVA**

Alumno: Rhonal Reyes Pardo

***Programa en ejecución:***



***Código implementado:***



***Detalle del código:***

package com.example;

import java.util.Random;

import java.util.concurrent.CountDownLatch;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.atomic.AtomicInteger;

import io.reactivex.Observable;

import io.reactivex.schedulers.Schedulers;

/\*\*

 \* Taller programación asincrónica / Hilos

\* Uso de RxJAVA

\* Caso: Evaluar el Cálculo del interés diario a pagar para 5,000 saldos de cuentas de ahorro

\*  Solo se paga a los saldos superiores a $3,000,000

 \*/

public class App

{

    public static void main( String[] args ) throws InterruptedException

    {

        // Generar # de Cuentas/Saldos

        final int NUMERO\_EVALUACIONES = 50000;

        // Tasa de interés diaria a pagar/calcular

        final double TASA\_INTERES\_DIARIO = 0.00005;

         // Latch para esperar que todos los hilos terminen

        CountDownLatch latch = new CountDownLatch(NUMERO\_EVALUACIONES);

        // Crear un ExecutorService con 4 hilos

        ExecutorService executorService = Executors.newFixedThreadPool(200);

        // Usar AtomicInteger para contar los saldos filtrados

        AtomicInteger countFiltered = new AtomicInteger(0);

        ;

        // Crear un observable que emite 10,000 saldos aleatorios

        Observable.range(0, NUMERO\_EVALUACIONES)

                .map(i -> generarSaldoAleatorio()) // Generar saldo aleatorio

                .filter(saldo -> saldo >= 3000000) // Filtrar saldos superiores a 3,000,000

                .doOnNext(saldo -> countFiltered.incrementAndGet()) // Incrementar el contador de saldos filtrados

                // Buscamos que el trabajo se distribuya entre los  hilos que definimos.

                .observeOn(Schedulers.from(executorService)) // Usar un ExecutorService con varios hilos

                .doOnNext(saldo -> {

                    double interes = calcularInteres(saldo, TASA\_INTERES\_DIARIO);

                    String hiloActual = Thread.currentThread().getName();  // Obtener el nombre del hilo actual

                    System.out.println("Hilo: " + hiloActual + " | Saldo: " + saldo + ", Interés Calculado: " + interes);

                })

                .doOnComplete(() -> {

                    latch.countDown(); // Decrementa el contador cuando se complete

                    System.out.println("Número de saldos después del filtro: " + countFiltered.get()); // Imprimir el conteo de saldos filtrados

                    System.out.println("Todos los procesos han terminado.");

                })

                .subscribe();

                //.subscribe(System.out::println);

        // Esperar a que todas las evaluaciones terminen

        latch.await();

        // Apagar el ExecutorService después de que termine el procesamiento

        executorService.shutdown();

    }

    // Genera un saldo aleatorio entre 1,000,000 y 5,000,000

    private static double generarSaldoAleatorio() {

        Random rand = new Random();

        return Math.round(1000000 + (rand.nextDouble() \* (5000000 - 1000000)));

    }

    // Calcula el interés usando la fórmula: saldo \* tasa de interés diaria

    private static double calcularInteres(double saldo, double tasaInteresDiaria) {

        return Math.round(saldo \* tasaInteresDiaria);

    }

}