Trabajo Práctico
Programación Concurrente
III

Ejercicio 1

El siguiente programa se encarga de notificar a los clientes de un supermercado de las promociones del día. Además notifica al departamento de marketing que la promoción del día ya fue difundida.

```
import java.time.Instant;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;

public class PromotionSender {

   public static void main(String[] args) {
       List<String> promotions = Arrays.asList("Descuento en Café: ",
               "Descuento en Refrescos: ",
               "Descuento en Congelados: ");
       notifyPromotions(promotions);
       System.out.println("Se realizaron todas las notificaciones de la
promoción.");
   }

   private static void notifyCustomers(String promotion) {
       try {
           TimeUnit.SECONDS.sleep(1);
           System.out.println("Cliente: " + promotion);
       } catch (InterruptedException e) {
           //
       }
   }

   private static void notifyMarketing(String promotion) {
       try {
           TimeUnit.SECONDS.sleep(1);
           System.out.println("Marketing: " + promotion);
       } catch (InterruptedException e) {
           //
       }
   }

   private static void notifyPromotions(List<String> promotions) {
       for (String promotion : promotions) {
           promotion = promotion + "30%";
```

```
            promotion = promotion + " Sólo por hoy";
            notifyCustomers(promotion);
        }
        notifyMarketing("Hoy se publicitó un descuento del 30%");
    }

}
```

La salida del programa es la siguiente:

```
Cliente: Descuento en Café: 30% Sólo por hoy
Cliente: Descuento en Refrescos: 30% Sólo por hoy
Cliente: Descuento en Congelados: 30% Sólo por hoy
Marketing: Hoy se publicitó un descuento del 30%
Se realizaron todas las notificaciones de la promoción.
```

a) Implementar el método `notifyPromotionsParallel` donde la notificación a los clientes sea concurrente usando **`ParallelStream`**.
b) Implementar el método `notifyPromotionsCompletable` donde la notificación a los clientes sea concurrente usando **`CompletableFuture`** de forma que la notificación al equipo de marketing sea independiente de la ejecución de la notificación a los clientes.

Ejercicio 2

Retomando el **Ejercicio 5 del TP Programación Concurrente Thread Safety** que contaba la cantidad de líneas de los archivos de un directorio, modificar la implementación para que:

- Se listen uno a uno los archivos del directorio en cuestión indicando
  - Nombre del archivo (Path)
  - Cantidad de líneas del archivo (Reutilizando el método del Ejercicio 5)
  - Tamaño del archivo (Puede usar Files.size(path))
- Implementarlo de forma de utilizar un `CompletableFuture` para contar las líneas de un archivo y otro `CompletableFuture` para obtener el tamaño de un archivo. Luego combinar ambos en un nuevo `CompletableFuture` que retorne la línea a imprimir con toda la información del archivo. ¿Puede la clase `FileLinesCounter` de la solución seguir siendo un `Callable`?
- Puede usar un `StringBuffer` para que, a medida que termine el `CompletableFuture` de cada archivo, se agregue una nueva línea al buffer con el resultado final a imprimir.