Proyecto de la la unidad formativa 1 del módulo Programación de servicios y procesos.

1. Instrucciones.

2. Requisitos mínimos para obtener 5.

3. Requisitos para obtener más de 5.

4. Entrega.

1. Instrucciones.

1. Tomar como punto de partida una copia del proyecto desarrollado en la actividad 4.
2. Cada alumno deberá elegir un tema diferente y diseñar dos clases que actúen de manera similar a *Empleado* y *Jornada*.
    a. Las clases elegidas deberán tener algún atributo numérico sobre el que se realizarán los cálculos.
    b. Las clases elegidas deben tener algún atributo adicional como por ejemplo *nombre* en *Empleado* o *fecha* en *Jornada*. Sobre estos atributos no habrá que realizar cálculos en el programa, pero mejoran la presentación de los listados.
3. Será obligatorio usar el patrón observer del API Java.
4. Las actualizaciones de elementos de ventana se realizarán desde el EDT.
5. El enunciado está descrito en términos de *Empleado* y *Jornada*. Cada alumno deberá de trasladar los requisitos a su tema y clases elegidas.

2. Requisitos mínimos para obtener 5.

6. Volatile.
    a. Al iniciar el proceso de Productores-Consumidores deberá aparecer una ventana de diálogo preguntando si se desea cambiar las unidades de medida en la cual los productores generan los datos.
    b. La medición de tiempo del proceso deberá de contar también el tiempo que el usuario dedica a contestar a la ventana de diálogo.
    c. Hacer uso de un guarded block como el que se vio en la práctica 2.4 que impida que se inicie el proceso de Productores-Consumidores mientras que el usuario no conteste la ventana de diálogo.
    d. Será necesario ejecutar el programa usando la variable volatile para comprobar que el proceso funciona normalmente.
    e. Será necesario ejecutar el programa quitando la definición de volatile de dicha variable para comprobar que el programa no funciona normalmente. En concreto, no debería de iniciarse nunca el proceso de

Productores-Consumidores aunque el usuario conteste a la ventana de diálogo.

3. Requisitos para obtener más de 5.

7. Lista en la que los consumidores acumulan los empleados.
    a. Inicialmente la lista de empleados estará vacía.
    b. Cada consumidor añadirá a la lista al empleado solamente en caso de que no exista. En el momento de creación del empleado, el atributo *horasTotales* tomará como valor el número de horas de la jornada que se está procesando.
    c. Si el empleado ya existía, actualiza el valor del atributo *horasTotales*.
    d. Para comprobar si existe el empleado en la lista será necesario implementar un algoritmo de búsqueda.
8. Cálculo de medias.
    a. Implementar un cálculo de media de horas de los elementos en alguna de las dos listas, jornadas o empleados.
    b. La media ha de ser calculada con varios threads.
    c. Deberá mostrarse el tiempo que ha llevado calcular la media.
    d. Deberá indicarse cuántos threads se han usado para el cálculo de la media.
    e. Deberá indicarse cuántos elementos tiene la lista sobre la que se ha calculado la media.
9. Productores con listados independientes.
    a. Cada productor tiene su propio *ArrayList<Jornada>* en vez de ir generando jornadas de una lista común.
    b. Esto podría representar una situación en la que cada productor genera jornadas procedentes de diferentes fuentes de información.
10. Informe.
    a. Añadir un nuevo botón para generar un informe.
    b. Dicho botón estará inactivo hasta que no finalice el proceso de Productores-Consumidores.
    c. El proceso consistirá en generar un informe del siguiente tipo:

```
Empleado 0, Puesto: Caja, duración: 5 horas, duración total: 11
Empleado 2, Puesto: Mantenimiento, duración: 8, duración total: 28
Empleado 0, Puesto: Almacén, duración: 6 horas, duración total: 11
...
```

    d. Al iniciarse el proceso se generan dos threads que accederán a *ArrayList<Jornada>* y a *ArrayList<Empleado>*.
    e. Cada thread deberá de tener acceso exclusivo a las dos listas (no solamente a una) para poder generar una línea del informe.
    f. Cada thread tardará 100ms en generar una línea de informe.
    g. Será necesario indicar la duración del proceso.
11. Deadlock.
    a. Generar un abrazo mortal haciendo uso de synchronized sentences.
    b. Resolver el abrazo mortal.
    c. Cada thread notificará la línea de informe a la ventana haciendo uso del patrón Observer.

4. Entrega.

12. Entregar dos archivos separados:
    a. Archivo PDF con las capturas de ejecución que se piden.
    b. Archivo ZIP que contenga la carpeta del proyecto Netbeans.
13. El archivo PDF contendrá:
    a. Las capturas representativas de la ejecución del programa.
    b. Elementos implementados de la lista de requisitos para obtener **más** de 5.
    c. En caso de que los haya, elementos implementados no incluidos en los requisitos del proyecto.