Examen Final – Febrero 2017

Hora final de entrega de examen: 13:30

En la web de la asignatura (Entregables y evaluación | Ficheros examen 201702) tienes los ficheros de trabajo. *(Nota: los ficheros fuente están codificados en UTF-8. Deberían compilar porque las tildes y ñs están solo en los comentarios, pero si las ves de forma incorrecta puedes corregirlo con botón derecho en el paquete | Properties | Text File Encoding - Other | UTF-8)*

- Sólo se permite la conexión a internet hasta las 9:30. Aprovecha ese tiempo para hacer todas las búsquedas que necesites. A partir de ese momento **debes desconectar wifi/red**.
- Para entregar el examen **comprime** todo **el directorio src** (que la carpeta src esté incluida) y cambia el nombre de ese fichero con tu **nombre y apellido**. Vuelve a conectar la red, y envía el examen por alud (Entregables y evaluación | Entrega examen 201702) y adjunto por email (andoni.eguiluz@deusto.es).
- Es obligatorio además tras entregar contestar al **formulario de entrega** (ALUD: Formulario 201702).

1. Planteamiento

Siguiendo con el desarrollo de las farmacias de guardia de Bizkaia que se realizó en la convocatoria ordinaria, nos piden ampliaciones sobre la gestión ya existente. La base del sistema son estas dos clases:

- **FarmaciaGuardia**. Permite definir farmacias de guardia. Cada uno de estos objetos representará a una farmacia indicándose su localidad, hora de apertura y cierre, zona(barrio), dirección postal, y teléfono.
- **MapaFarmacias**. Permite cargar las farmacias de guardia en un mapa de memoria, donde cada localidad (clave) contendrá una lista (ArrayList) de las farmacias de guardia en esa localidad.

La clase **Main** es la principal del sistema, y lanza una ventana ya funcional (**VentanaFarmacias**) que se tendrá que completar de acuerdo a las siguientes tareas.
Todas las tareas son independientes y puedes realizarlas en cualquier orden. No hace falta realizar todas, observa la puntuación (sobre 10) de cada una y decide cuáles quieres desarrollar y en qué orden.
No hay que añadir ningún fichero, **todas las tareas se deben realizar en los ficheros existentes, en los lugares marcados al efecto con los comentarios // TAREA N**

2. Tareas

Tarea 1 [1,5]. **JUnit** (*FarmaciaGuardiaTest.java – unas 25 líneas de código*). Añade a la prueba unitaria ya existente de la clase *FarmaciaGuardia* un test que compruebe el funcionamiento correcto del Mapa ordenado de farmacias (Clase MapaFarmaciasOrdenadas). Que haga al menos las siguientes cosas:

- Crea un mapa de farmacias normal con una farmacia de lekeitio y cuatro de Bilbao. Crea un mapa de farmacias ordenadas con ese mapa.
- Comprueba que en el mapa primero está Bilbao y luego Lekeitio
- Comprueba que en el conjunto de farmacias de Bilbao, el orden es por horario, zona y dirección (para ello prueba que las farmacias de Bilbao tengan al menos dos horarios distintos, al menos dos zonas distintas, y al menos un par de farmacias con el mismo horario y zona pero distinta dirección).

Tarea 2 [2]. **BDs** (*VentanaFarmacias.java – unas 20 líneas,* clase nueva *BD.java*). Queremos hacer una analítica de cómo cada usuario utiliza nuestro programa. Para ello queremos tener actualizado a través de las distintas ejecuciones el número de veces que pulsa el botón de Listado, el número de veces que pulsa el de Capicúa, y el número de veces que hace click en el reloj (da igual horas que minutos, izquierdo que derecho).
Para ello programa una base de datos nueva que tenga una tabla **analitica** con dos columnas: código (String) para el tipo de interacción ("Listado", "Capicua", "Click") y contador (entero) para el número de veces que se hace cada una de ellas. Utiliza las operaciones básicas de BD de la forma más eficaz (insert, update, select).
Programa las operaciones básicas de BD en una clase aparte BD.java, y utilízalas desde *VentanaFarmacias*. Deberás localizar los mejores puntos del código donde implementar cada cosa, acordándote de usar o crear la BD al principio y cerrarla al final. Marca los puntos donde incluyas código con // TAREA 2.
Para comprobar que todo va bien, cada vez que haya una nueva interacción visualiza al final del área de mensajes el número total de interacciones de ese tipo que van ocurriendo.

Tarea 3 [2]. **Hilo** (*VentanaFarmacias.java – unas 30 líneas de código*). Queremos hacer un poco más colorida la tabla de farmacias. Para ello vamos a definir un nuevo botón que al ser pulsado, haga que el color de las cabeceras de la tabla vaya cambiando entre verde y cyan. Para ello:

- Crea un nuevo botón "Color" en la botonera superior.
- Haz que al ser pulsado por primera vez, cree un hilo que cada 5 milésimas de segundo modifique el color de dibujado de cabeceras (atributo ya existente *colorCabecera*), del siguiente modo: subiendo 1 el componente azul hasta que llegue a 255, bajando 1 ese componente hasta que llegue a 0. Y así sucesivamente.
- Haz que al ser pulsado por segunda vez, detenga el hilo para que el color quede fijado en su valor actual (utiliza el método predefinido *interrupt()*).
- Si se vuelve a pulsar, se comporta de la misma manera (vuelve a activar el coloreado, lo para; etc.)

Recuerda que la interrupción del hilo puede ocurrir mientras el hilo está esperando (el *sleep*) con lo que deberás considerar esa excepción para parar el hilo si se interrumpe la espera. No utilices el método desaconsejado **stop()**.

. ----------

Tarea 4 [2+1]. **JTree + Recursividad** (*VentanaFarmacias.java – 25/35 líneas de código*). Añade a la botonera un botón "Árbol" que al ser pulsado sustituya la tabla de farmacias por un árbol de farmacias (JTree) donde la raíz es un nodo "Farmacias", el segundo nivel es un nodo por localidad (ordenadas por orden alfabético), y el tercer nivel la lista de las farmacias de cada localidad, en orden de horario, zona, y dirección. Puedes utilizar el mapa ordenado para este trabajo.

Para un punto extra, haz el proceso con las farmacias en orden **inverso** de horario, zona y dirección (las localidades igual), utilizando un método recursivo que se encargue de crear los árboles en sentido inverso [la inversión de orden debe estar estrictamente realizada aprovechando las ejecuciones recursivas].

. ----------

Tarea 5 [1,5]. Evento de swing. (*VentanaFarmacias.java – unas 10 líneas de código*). Al redimensionar la ventana la tabla se redimensiona y todas las columnas se expanden por igual. Modifica el código del constructor de la ventana para que al redimensionarla, la tercera columna (el teléfono) tenga una anchura mínima y máxima de 80 píxels, o si es mayor que 80, la sexta parte del ancho actual de la tabla.

. ----------

Tarea 6 [2]. Estructuras de datos. (*PruebaEficienciaMapaFarmacias.java – unas 60 líneas de código*). Verás que hay una clase adicional, *PruebaEficienciaMapaFarmacias*, que tiene el banco de pruebas visto en la práctica 5/6 preparado ya con el mapa de farmacias. Está programado para llenar un mapa con "n" farmacias ficticias de Bilbao y Barakaldo, y medir el tiempo que tarda un proceso en buscar otras "n" farmacias ficticias de Bilbao (no existentes) en ese mapa.

Se trata de añadir a este banco de pruebas otros dos tests que utilicen las mismas farmacias de prueba y las mismas búsquedas para el test:

- Utizando el mapa ordenado ya suministrado (*MapaFarmaciasOrdenadas*). El tiempo debería ser sustancialmente menor.
- Utilizando un arraylist con todas las farmacias (entremezclando las de Bilbao y Barakaldo). El tiempo debería ser algo mayor.

. . ----------

Tarea 7 [1,5]. **Recursividad** (*FarmaciaGuardia.java – unas 15 líneas de código*). Añade el botón "Codif" para que se codifiquen las direcciones de todas las farmacias de Bilbao y aparezcan en el área de texto de mensajes.

Para codificar cada dirección debes crear un método recursivo que recibe la dirección original y la devuelve codificada del siguiente modo: primero se toma el carácter que esté exactamente en la mitad (si son pares, el string vacío) y a partir de ahí se toma un carácter hacia la derecha y otro hacia la izquierda, uno a la derecha y otro a la izquierda, y así hasta llegar a los extremos. Por ejemplo las siguientes direcciones quedarían así:

```
Autonomía, 9        queda codificada como:     "moínao,t u9A"
Autonomía, 41       queda codificada como:     "míoan,o t4u1A"
```