Examen Programación III – Enero 2019

- En la web de la asignatura ("Exámenes y Evaluación") tienes los ficheros con los que se va a trabajar en un archivo comprimido. Crea un proyecto en eclipse y descomprime ese fichero en la carpeta src de ese proyecto (¡respeta los paquetes ya creados!).
- Al final del examen, comprime esa carpeta (.zip o .rar) y comprueba que tiene los ficheros que has modificado. Entrega en la tarea correspondiente de ALUD.

1. Planteamiento

Tienes a tu disposición un sistema de seguimiento de sesiones en centros escolares y un editor de animaciones de sprites sobre los que se pedirán las tareas. Respeta los lugares indicados para cada tarea e indica con un comentario `// Tn` si modificas código en cualquier otro punto.

2. Tareas

Se pide que realices las siguientes tareas (su puntuación sobre 10 entre corchetes):

1. JUnit [2,5] [`TestAnimacion.java`, 30 líneas de código] Crea una clase de JUnit nueva **`TestAnimacion`**, para probar que la animación de los sprites funciona. Para ello debes crear una ventana de sprites al inicio y liberarla al final, y en el test realizar el siguiente proceso (observa los corchetes que indican los tres chequeos que deben ser ciertos):

- Añadir un fichero gráfico cualquiera a la lista (para que haya algo que mover)
- Visualizar la ventana
- Asignar 100 al cuadro de texto de la velocidad y 25 al del ángulo
- Simular un click en el botón *bSecuenciaAnim* (esto hará que la imagen se vaya moviendo)
- Realizar un bucle de espera que haga pausas sucesivas de 100 milisegundos, **comprobando** [1] para la prueba que el gráfico (*lAnim*) se va moviendo a la derecha (la X se ha incrementado entre antes y después de cada pausa). Debes acabar el bucle cuando no se produzca movimiento del gráfico.
- La animación se para cuando el gráfico se sale del panel, así que tras el bucle **comprueba** [2] que la coordenada *x* o la *y* de *lAnim* están en los límites del panel gráfico (*pArena*).
- Comprueba por último que el tiempo total que tarda en hacerse y detenerse la animación con esta configuración de velocidad y ángulo **es inferior a 10 segundos** [3].
- Puedes tener que realizar algunas pequeñas pausas para dar tiempo a que Swing realice su trabajo.

2. Estructuras de datos [3] [`VentanaDatos.java`, 45-50 líneas de código]. Queremos hacer un listado de avisos a las mentoras para las que se cumpla alguna de las dos condiciones:

- Han pasado más de tres semanas entre dos de sus sesiones
- Hay diferencia de asistencia significativa entre sesiones (más de dos personas de diferencia entre su sesión en la que tiene más personas y la sesión en la que tiene menos).

Para realizar esta tarea, tienes ya definida la clase de utilidad `Datos.Sesion` en la clase de Datos. También puedes usar el formateador de fechas `sdf` que verás allí.

Resuelve este proceso en el método *clickT2()* que se lanza con el botón "T2" correspondiente. Debes recorrer todas las sesiones partiendo del fichero de datos Mentoring2018.csv, y crear **obligatoriamente la siguiente estructura de datos**: un mapa con **clave email de mentora** y valor un ***TreeSet* de *Datos.Sesion***. Ya que el treeset permite orden, en el mapa vete introduciendo por cada mentora el conjunto de sus sesiones, que **se deben ir ordenando por fecha**.

Tras cargar el mapa completo, podrás recorrerlo para buscar distancias largas entre fechas o diferencias de varias personas entre la sesión con más y con menos asistentes.

Tienes un fichero `datos.txt` en el que verás con el set de datos suministrado qué mentoras deberían aparecer con este aviso (usa ese mismo formato para mostrarlas en consola).

3. Base de Datos [2] [`VentanaDatos.java`, 30 líneas de código] Si pulsas una vez el botón "Carga feedback" y luego el "Guardar en BD" se te generará la base de datos sqlite de centros y sesiones (con una tabla para cada centro y otra para cada sesión, definidas en BD.java, líneas 36-46).

Programa el método *clickT3()* asociado al botón "T3" para que **partiendo estrictamente de la información de la base de datos**, visualice en consola los 10 centros con menor satisfacción media de sus estudiantes (ordenados por satisfacción media). Además para cada centro mostrar las sesiones que ha habido (no sacar las de 0 estudiantes) en orden de sesión (ver formato de salida en el fichero `datos.txt`).

Si necesitas más métodos que los existentes en la clase BD, programa directamente las sentencias de base de datos, o añade los métodos que necesites.

4. Recursividad [2,5] [`ControladorVentanaSprites.java`, 30 líneas] Observa que el botón "Buscar" localiza los ficheros png que haya en un directorio dado, utilizando el método *cargaFicherosGraficosOrdenados*. Modifica ese método para que se busquen ficheros también en sus posibles subcarpetas, respetando las siguientes condiciones:

- Para evitar que tarde demasiado tiempo, como máximo se procesan tres niveles anidados de carpetas (contando como primer nivel la carpeta inicial). Es decir, no se procesan las subcarpetas de nivel de anidamiento cuatro o superior.
- Para evitar que los ficheros sean demasiados, el proceso debe dejar de buscar si ya se han encontrado **50 ficheros png**.
- Deben cargarse a la lista de la ventana de ficheros los pngs de cada carpeta, **antes** de procesar recursivamente sus subcarpetas.
- El orden alfabético debe mantenerse solo dentro de cada carpeta (si se encuentran ficheros en varias carpetas, en la lista quedan ordenados entre sí solo los de la misma carpeta).
- Visualiza en consola el número total de ficheros recorridos en la búsqueda (sean o no pngs).
- Puedes utilizar los parámetros o valores de retorno que veas oportunos y debes añadir algún método. Es imprescindible que el algoritmo de procesado **sea recursivo**.

Tienes una carpeta con ficheros para probar esta tarea en ALUD (*sprites-pruebaEx.zip*). Y una descripción del resultado de aplicarla en ese directorio (en el fichero correspondiente `datos.txt`)

5. Swing [3] [`VentanaDatos.java`, 65 líneas de código]. Realiza las siguientes tareas en los puntos del fichero fuente que creas más oportunos. Recuerda comentar `// T5` los lugares donde modifiques código.

- Que la tabla visualice los números enteros centrados en sus celdas.
- Que la tabla visualice los valores cero de sesiones o estudiantes en gris (y el resto en negro).
- Que la tabla visualice los valores medios de satisfacción como progressbars (extrapolando el valor doble de 0.0 a 5.0 a un progreso de 0 a 500).
- Que al pasar el ratón por encima de la tabla, si es en uno de esos valores medios de satisfacción aparezca el valor real correspondiente completo en el label superior *lMensaje* (y desaparezca si se mueve el ratón fuera de esas celdas).
- Que al hacer doble click en un código de centro (primera columna) aparezca un popup con JOptionPane que muestre el nombre del centro y las abreviaturas definidas en él.

Abajo puedes ver una captura parcial de ejemplo con esta tarea resuelta, y un ejemplo de popup.