TP N°4

Bacchetta, Tomás

El programa “Cuartel” permite dar de alta, modificar, y eliminar soldados, que a su vez pueden asignarse a escuadrones. Para estos escuadrones luego, se simulan expediciones, de donde se obtendrán datos aleatorios que serán cargados en los atributos de los soldados y de los escuadrones.

Las incursiones necesitan un escuadrón lleno, con las cuatro clases existentes, y al momento de enviarlas a una misión se seleccionará una locación, que según el mapa mientras más arriba más tiempo tardaran cierta cantidad de segundos (entre 1 y 20), siendo para el programa un segundo equivalente a una hora. Este detalle se verá reflejado en las estadísticas pertinentes.

En el apartado Incursiones se puede acceder a un registro conectado a una base de datos, que muestra las estadísticas históricas de todas las incursiones realizadas, que engloban cierta selección de datos, y enlazadas con los nombres de los soldados que estuvieron presentes en dicha misión. El rango de cada soldado de terminará cuantas bajas, curaciones, recursos, etc; habrán recolectado. Las incursiones se van guardando a medida que se los escuadrones vuelven de las misiones. Los informes referidos a soldados son independientes de las bases de datos, por ende ignoran las estadísticas históricas, sólo tomando los valores en memoria o serializados.

El registro en la base de datos es independiente de los datos personales de los soldados, por lo que, aunque se eliminen soldados, quedarán registrados sus nombres en las incursiones particulares donde participaron. Serán también independientes sus respectivas estadísticas, que pueden verse en sus bios, pero no así las de los escuadrones. Si se eliminara el registro, o una parte de él, las estadísticas de los escuadrones se verán afectadas. Esto se debe a que los escuadrones son fijos y no se pueden eliminar, sino vaciar de soldados.

La tabla del registro se puede ordenar por valores, con el criterio que se deseé, como así también se podrá filtrar qué escuadrones mostrar.

La conexión con la base de datos es imprescindible para poder iniciar el programa, siendo facilitado un backup de la misma, REGISINC.bak en la carpeta de la solución de este proyecto, que deberá ser importada en el SQL Studio.

Los escuadrones tienen un límite de 4 soldados, y todos deben ser de clase diferente. Las clases se eligen al momento de dar de alta un soldado, y determinan también las chances de

conseguir diferentes atributos a la hora de realizar las expediciones.
Las estadísticas de los soldados se borran a medida que se los va expulsando, pero la de los escuadrones es persistente (no importa si se expulsa a todos sus integrantes).
El estado del programa puede guardarse en formato .xml o .json. El informe principal se puede guardar como texto plano .txt
Dentro de la carpeta de la solución hay un archivo .xml llamado TESTCARGA, donde ya hay datos cargados con los que se puede corregir más fácilmente.
Implementación de la teoría:

Clase 10: Excepciones

En los archivos de clases ArchivoIncorrectoOCorruptoExcepcion.cs y ErrorDeArchivoExcepcion.cs se definen expeciones propias, que son lanzadas en Archivo.cs y sus clases derivadas. Luego se usan tries en el código de FrmCuartel.cs, en los métodos de apertura y guardado de archivos.
La primera excepción está pensada para la apertura de archivos (puede ser que un archivo fuese editado a mano y arruinado y el programa podría, en primera instancia, tratar de leerlo, por ende es conveniente comprobar que sea compatible con el programa).
La segunda excepción se encarga del guardado, que el directorio exista, que sea válido, y que no esté vacío.
Es imprescindible controlar excepciones en los apartados de carga y guardado ya que la complejidad de la interacción con el sistema operativo y las unidades de almacenamiento son caldo de cultivo para diversos errores a veces no tan evidentes.

Clase 11: Pruebas Unitarias

Se prueban algunas sobrecargas de operadores de las clases Cuartel y Escuadrón. Se encuentran en TestMiscelaneos.cs, en proyecto Test.

Clase 12: Tipos Genéricos

Se utilizan tipos genéricos para el manejo de archivos, ya que parto del desconocimiento de qué tipo de objeto voy a guardar. En las clases derivadas de Archivos.cs son utilizados los tipos genéricos, y son asignados a un tipo a la hora de instanciar estas clases, tanto en los métodos

Abrir(), Guardar(), GuardarComo() de clase FrmCuartel.cs, como en el método btnGuardarInforme_click de la clase FrmInforme.cs, aunque en ese caso al ser txt, el tipo admitido va a limitarse a ser string. También la interface IManejoDeArchivos utiliza tipos genéricos ya que los métodos que define pueden devolver o recibir cualquier tipo (limitado a tipo de clase).

Clase 13: Interfaces

También se utilizan interfaces con IManejoDeArchivos, que naturalmente se encarga de definir los métodos que van a utilizar las clases derivadas de la clase abstracta Archivo. Por las razones expuestas más arriba, también se la combina con los tipos genéricos.

Clase 14: Archivos y Serialización

En Archivo.cs y sus clases derivadas como así también la interfaz IManejoDeAchivos, se permite la interacción con archivos, tanto por serialización (para guardar y abrir el estado del programa) como por texto plano (para guardar el informe). Se utilizan lógicamente también en FrmCuartel.cs (para serialización), en FrmInforme.cs (y para guardado de archivos). Como no guarda información de clases polimórficas, se puede elegir tanto xml como json para el guardado y apertura del estado de todo el proyecto, aunque de ser diferente el caso se podría usar xml para un guardado general, y json para un guardado de algún dato específico.

Clase 15 y Clase 16: SQL

En IncursionDB.cs y en EscuadronDB.cs están los métodos que interactúan con la base de datos de registro de incursiones. También se utiliza la base de datos para extraer estadísticas a mostrar en el formulario FrmVerEscuadron.cs. Para realizar informes que tengan que ver con escuadrones o características de la incursión, el programa se comunica también con la base de datos.

Cada incursión que se haga será guardada en la base de datos.

Para la lectura existe una clase llamada RegistroIncursión que es la que se instanciará agregando los datos de las tablas de Incursión como también la de escuadrón (que tiene los nombres de los soldados que participaron en esa única expedición).

Clase 17: Delegados y Expresiones Lambda

Se utiliza un delegado con la firma FinalDeIncursion(Incursion incursion) en el namespace Entidades (visible en el archivo Incursion.cs), donde se encapsulará el evento AvisarFinDeIncursion.

Se utiliza una expresión Lambda pasada como parámetro que suplanta a un delegado Action a la hora de instanciar un nuevo hilo en FrmCuartel.cs, en el método btnDesplegar_Click.

Clase 18: Hilos

En el método btnDesplegar_Click, disparado al presionar el botón Desplegar en el FrmCuartel.cs, se genera un hilo, que ejecutará el método de instancia RealizarIncursion de la clase Instancia. Antes de terminar de ejecutarse el método, y por ende, antes de que finalice el ciclo de vida del hilo, se invoca al evento AvisarFinDeIncursion.

El resultado es que cada incursión que se realice tendrá su hilo, que nace al momento del despliegue, y muere cuando los soldados regresan a la base.

Clase 19: Eventos

El evento AvisarFinDeIncursion tiene suscripto un método, CargarYMostrarResultadosIncursion, de FrmCuartel, que entre otras cosas se encarga de alertar al usuario el fin de una incursión (o sea, el final de vida de un hilo), por medio de un cuadro de mensaje. Entonces al terminar el hilo de ejecutar RealizarIncursion, se ejecutará a continuación el método descripto anteriormente.

Clase 20: Métodos de extensión

En Extensiones.cs se crearon métodos de extensión específicamente para lidiar con las diferencias de tiempo iniciales y finales de las incursiones, teniendo en cuenta que para el programa los segundos son horas, como así también la obtención del valor redondeado de estas diferencias.