Programación Web

Informe de Prácticas

Práctica 1. Introducción a Java

Daniel Ortega León

Damián MartÍnez Ávila

Introducción

Esta práctica servirá para iniciarnos en el desarrollo con el lenguaje Java, se hará uso de los elementos habituales del lenguaje, E/S de ficheros y de consola, desarrollo de ficheros de propiedades y patrones de diseño como Singleton y Factory.

Ejercicio 1

El ejercicio 1 nos pide que implementemos un gestor de Contactos con las funcionalidades CRUE (Create, Read, Update, Delete) y de búsqueda, donde cada Contacto debe tener nombre, apellidos, fecha de nacimiento, email y un conjunto de intereses asignados mediante un sistema de tagging.

Decisiones de diseño e implementación

A la hora de empezar a implementar, primero creamos la clase Contacto con los atributos anteriormente mencionados, sus getters/setters y su respectivo constructor. Hay que tener en cuenta que los intereses se definen en una enumeración dentro de la clase. Una vez creada, podemos seguir con la clase GestorContactos, la cual tiene como atributo un ArrayList de Contacto que representa los contactos del gestor. En esta clase implementamos el patrón de diseño Singleton para que solo pueda existir una instancia de esta clase en nuestro programa.

Se procede a explicar la implementación de las funciones para las funcionalidades CRUE :

-Create: darAlta(): Función que pide los atributos de un contacto mediante consola para finalmente crear una nueva instancia de contacto, añadirla al ArrayList y por último guarda los datos en un fichero .dat.

-Read: cargarDatos(): Al crear una instancia de la clase GestorContactos desde el constructor se llama a esta función, la cual se encarga de leer objetos de tipo contacto del fichero “fich.dat” y los añade al vector hasta que se llegue al final del archivo y mande una EOFException.

-Update: actualizarContacto(): Permite modificar los atributos de un contacto el cual es pasado por parámetro mediante la función buscarContacto().

-Delete: darBaja(): Permite eliminar un contacto el cual es pasado por parámetro mediante la función buscarContacto() y se recorre la lista de contactos. Si coincide el email del contacto del parámetro y el del contacto de la lista, este último es eliminado de la lista y se guardan los datos en el .dat.

La función buscarContacto() permite obtener un objeto según el criterio de búsqueda seleccionado, que puede ser el nombre y los apellidos, el email, un interés o la fecha de nacimiento.

El main recibe el nombre de un fichero properties que nos indica dónde se guardan los ficheros para almacenar nuestros datos. Finalmente, hemos implementado una clase ControlDeErrores para limitar las entradas por consola.

Fuentes Consultadas

Hemos recurrido a los ejemplos de moodle para saber cómo implementar el patrón de diseño Factory, la documentación de java para temas como la serialización de ficheros y las funcionalidades de ciertas funciones y también ciertos foros como StackOverflow.

Análisis de Dificultades Encontradas

A la hora de almacenar los contactos en un fichero binario, nos daba ciertos problemas al no saber que era necesario que la clase que se quería guardar en el fichero fuera

serializable, la serialización es un mecanismo donde un objeto se puede representar como una secuencia de bytes que incluye los datos del objeto

Hemos tenido algunos problemas con la clase Scanner ya que cuando haces un nextInt() y despues no funciona el nextLine() debido a que coge el "\n" del nextInt(), algo que ocurre en otros lenguajes pero no caímos en el momento de la implementación. También, cuando cierro un Scanner que esté en un método externo al main, los Scanner dejan de funcionar porque parece ser que se cierran todos los flujos de entrada por consola y por ello no hemos cerrado ningún Scanner que no pertenezca al main.

Ejercicio 2

En el ejercicio 2 se pide implementar un tablón de anuncios, en el que pueden haber varios tipos de estos. También se debe implementar un gestor de anuncios que controle el estado de cada anuncio.

Decisiones de diseño e implementación

Teniendo en cuenta que necesitamos un gestor de usuarios, podemos reutilizar las clases del ejercicio 1 con algún pequeño cambio, ya que nos piden que se puedan añadir nuevos intereses, por lo que no se puede usar enumeraciones y hemos optado por crear una clase que lea y escriba en un fichero los distintos intereses.

Creamos la clase Anuncio con los atributos especificados, sus getters/setters, constructor y una enumeración con los posibles estados en los que puede encontrarse el anuncio. Hemos aplicado el patrón de diseño Factory para crear los distintos tipos de anuncios que nos especifica el problema.

Necesitamos implementar también un gestor de anuncios que se encargue de controlar sus estados y todo lo relativo a las funcionalidades CRUE y busqueda de los anuncios excepto de la creación del propio anuncio, lo único que hace respecto a esto es añadir los nuevos anuncios en el ArrayList .

Por último, requerimos de una clase TablonAnuncios para que cada Contacto pueda tener su propio tablón en el cual se pueda iniciar el proceso de crear un anuncio mediante la factoría y mostrar los anuncios destinados a cada usuario.

Una vez definidas todas las clases necesarias, comenzamos con el main, en el cual creamos una instancia (única gracias al patrón de diseño Singleton) de cada clase (GestorAnuncios, GestorContactos, TablonAnuncios, Intereses) para poder usar todas las funcionalidades.

El main recibe el nombre de un fichero properties que nos indica dónde se guardan los ficheros para almacenar nuestros datos. Finalmente, hemos implementado una clase ControlDeErrores para limitar las entradas por consola.

Fuentes Consultadas

Hemos recurrido a los ejemplos de moodle para saber cómo implementar el patrón de diseño Singleton, la documentación de java para temas como la serialización de ficheros y las funcionalidades de ciertas funciones y también ciertos foros como StackOverflow.

Análisis de Dificultades Encontradas

Al haber decidido guardar los anuncios en un fichero binario, nos hemos encontrado el problema de que al leerlos, como son distintos tipos de anuncios y no se puede saber cual es cual no se podrían cargar en el sistema. Hemos decidido solucionarlo creando un fichero para cada tipo de anuncio.

La única forma de controlar los anunciosFlash que se nos ha ocurrido es actualizarlos cuando se invoca a la función que muestra el tablon de un usuario, aun sabiendo que esta clase no debería de gestionar los estados es la mejor manera que hemos visto de implementarlo

También ha costado acostumbrarse a la forma de administrar los proyectos en java y Eclipse en general nos ha resultado algo complejo hasta que nos hemos acostumbrado

Está hecho para funcionar en Java 7, la versión que está instalada en la uco. En principio, no debería haber problemas para ejecutar los ejercicios en otras versiones de java posteriores