Proyecto de programación

EIF204 Programación 2

Profesores:

Georges Alfaro Salazar
Santiago Caamaño Polini (coordinador)
Jennifer Fuentes Bustos

Descripción del proyecto

Se construirá un sistema sencillo de matrícula para una universidad. El sistema contempla la administración de escuelas, cursos, profesores y estudiantes.

Objetivos

El objetivo del proyecto es aplicar los conceptos teóricos, principios y técnicas estudiadas en clase. En particular, se busca incorporar dichos conceptos y técnicas de manera gradual con cada entrega. Es importante que al desarrollar el proyecto se escriba código correctamente estructurado, debidamente encapsulado y fácilmente reutilizable.

Plan del proyecto

El proyecto se completará en cinco partes. Se harán entregas parciales en las fechas indicadas más adelante. En cada una de las entregas se incorporarán diferentes elementos al modelo y se implementarán nuevas funcionalidades. Para cada entrega, se deberán incluir los diagramas de clase y los archivos de proyecto correspondientes. Cada profesor indicará cual es el ambiente de desarrollo (IDE) a utilizar. Los archivos de proyecto tienen que entregarse en el formato adecuado para su revisión. No se recibirán proyectos en un formato diferente al indicado. Los diagramas de clase se elaborarán usando alguna aplicación para ese efecto y se adjuntarán en formato PDF.

Detalle del problema

La aplicación a construir es un sistema sencillo de matrícula para una universidad. Se trata de un programa estándar de consola que permita al usuario ejecutar cada una de las funciones necesarias.

Consideraciones de implementación

El código del proyecto debe estar estructurado adecuadamente, respetando los principios de diseño estudiados en clase.

En particular:

- Las clases deben separar correctamente la declaración de la interfaz y su implementación, usando archivos de cabecera (archivos .h) y código fuente (archivos .cpp) por aparte. Los archivos de cabecera deberán siempre usar guardas (se puede especificar la directiva `#pragma` once cuando se utilice Visual Studio).
- Las clases de entidad no deberán contener código de entrada/salida (como salida a la consola usando `cout`, por ejemplo). Sin embargo, se pueden mostrar mensajes de comprobación al efectuar las pruebas del programa, para corroborar que se están ejecutando las funciones de manera correcta.
- El manejo de la interfaz con el usuario (vista) debe hacerse por medio de una o varias clases diseñadas para tal efecto.
- Se debe poder especificar el nombre del archivo que contendrá todos los datos de la aplicación como un parámetro en la línea de comando. SI el archivo indicado no existe, se informará al usuario (pero no se creará el archivo). Al cerrar el programa, éste consultará al usuario si debe o no actualizar los datos del archivo (si el archivo indicado al inicio no existe, se creará en el momento de salir de la aplicación, cuando el usuario haya solicitado guardar la información).
- Cuando se necesite manejar conjuntos de datos, se definirán e implementarán por medio de colecciones (contenedores). Es importante que el diseño general del programa considere la reutilización de código. En todos los casos, las colecciones (contenedores) y otras estructuras deben implementarse de la forma más general posible.
- No se podrán utilizar las colecciones provistas por la STL (*Standard Template Library*) de C++.

Evaluación

El proyecto se realizará de manera individual o en grupos de dos estudiantes, como máximo. Como se indicara al inicio del curso, cada profesor asignará los grupos de trabajo.

Los materiales se entregarán por medio del aula virtual, en el espacio designado para ello. El espacio estará abierto para la entrega hasta el último día de la semana indicada. No se recibirán documentos ni materiales de otro modo, ya sea en dispositivos USB ni por correo electrónico, salvo por indicación manifiesta del profesor. Queda a criterio del profesor utilizar un medio de entrega diferente cuando lo considere conveniente.

No se admitirá la entrega tardía de ninguna parte del proyecto. En este caso, se calificará esa parte con nota 0 (cero). Es importante notar que de cualquier manera cada etapa debe terminarse para completar la siguiente, aunque ya no sea calificada.

Las fechas de entrega y el valor porcentual de cada parte son como se detalla en la tabla siguiente:

Sem.	Desde	Hasta	Entrega	Valor (porc)
1	13/02/2017	19/02/2017		
2	20/02/2017	26/02/2017		
3	27/02/2017	05/03/2017		
4	06/03/2017	12/03/2017		
5	13/03/2017	**19/03/2017**	**Parte #1**	**6%**
6	20/03/2017	26/03/2017		
7	27/03/2017	02/04/2017		
8	03/04/2017	**09/04/2017**	**Parte #2**	**6%**
	10/04/2017	16/04/2017	(Semana Santa)	
9	17/04/2017	23/04/2017		
10	24/04/2017	30/04/2017		
11	01/05/2017	**07/05/2017**	**Parte #3**	**6%**
12	08/05/2017	14/05/2017		
13	15/05/2017	21/05/2017		
14	22/05/2017	**28/05/2017**	**Parte #4**	**6%**
15	29/05/2017	04/06/2017		
16	05/06/2017	11/06/2017		
17	12/06/2017	**18/06/2017**	**Parte #5**	**6%**

Requerimientos del proyecto

Parte 1

Descripción

La universidad es el concepto central o principal del sistema. Se requiere registrar el nombre, la dirección y el número de teléfono de la universidad. Una vez que se hayan registrado los datos de la universidad, solamente se permitirá cambiar la dirección o el número de teléfono, pero no el nombre. La universidad está formada por varias escuelas o departamentos, cada una con un nombre que la identifica. Al igual que ocurre con la universidad, no es posible cambiar el nombre de la escuela o departamento una vez que se haya ingresado.

Funcionalidad requerida

Las operaciones que la aplicación debe implementar en esta primera parte son:

- Inclusión de los datos de la universidad
- Actualización de los datos de la universidad (dirección y teléfono)
- Registro de escuelas
- Consulta de la lista de escuelas que pertenece a la universidad. Al mostrar la lista de escuelas, se indicará también el nombre de la universidad

Observe que no se solicita ninguna opción para mantenimiento de los datos de las escuelas.

Parte 2

Descripción

Cada una de las diferentes escuelas de la universidad coordina varios cursos. Cada curso tiene una sigla y un nombre descriptivo. Una escuela puede definir varios cursos, pero un curso solamente puede pertenecer a una única escuela.

Funcionalidad requerida

En esta parte, deberán implementarse todas las operaciones de gestión de cursos: inclusión de cursos nuevos, eliminación, modificación y consulta. Estas son las cuatro operaciones o funciones básicas del almacenamiento persistente, denominadas por las siglas CRUD (*Create*, *Retrieve*, *Update*, *Delete*), o ABC en español (Altas, Bajas, Cambios). En este momento, no se requiere todavía que el programa implemente la persistencia de datos utilizando archivos, sino que esta capacidad será diferida a una entrega posterior. Las operaciones se construirán usando colecciones (contenedores). Puede usar clases o métodos de prueba para facilitar la comprobación del programa.

Hay que agregar a la interfaz de usuario las opciones necesarias para permitir que se realicen las operaciones de gestión:

- Inclusión de cursos
- Recuperación de los datos de un curso
- Recuperación de la lista de cursos de una escuela particular
- Recuperación de la lista de todos los cursos impartidos por la universidad, organizados por escuela
- Modificación de los datos de un curso (en este momento, solamente se permitirá cambiar el nombre del curso)
- Eliminación de un curso

Parte 3

Descripción

En la universidad, existen profesores que imparten los cursos. Cada escuela cuenta con un grupo de profesores con los que trabaja. Un profesor solamente puede estar adscrito a una única escuela. En cada escuela, se designa un profesor como director por un período establecido. Para cada profesor, se registrará su número de cédula, nombre y los dos apellidos. Se debe indicar cuáles son los cursos impartidos por cada profesor. Como pueden existir varios grupos de cada curso, de la misma manera habrá diferentes profesores para cada uno.

Hay estudiantes registrados en la universidad cuya información debe registrarse y controlarse también. Los estudiantes matriculan varios cursos y pagan un arancel por los cursos en los que están inscritos (matrícula, pago de créditos y recargos). Los estudiantes pueden ser nacionales o extranjeros. Algunos estudiantes nacionales pueden disfrutar de una beca, indicada por un porcentaje de descuento. Los estudiantes extranjeros no pueden contar con ningún tipo de beca, y se les cobra un porcentaje de recargo que es fijo.

Cada curso tiene un número de créditos, que se usa para saber cuándo debe cobrarse. Un curso tiene normalmente 3 o 4 créditos. El costo por crédito es de ¢10,000. Además, a la hora de matricular se cancela un monto fijo por cargos administrativos de ¢15,000. Los estudiantes extranjeros deben pagar un recargo de 40% sobre el monto total.

Funcionalidad requerida

Esta parte comprende la mayor parte de la funcionalidad del programa, que quedará casi completa. Todas las clases de entidad estarán probadas y cualquier cambio o ajuste a la estructura de clases deberá terminarse aquí.

Se incluirán las opciones necesarias para gestionar la información de los profesores y asignarlos a los cursos de cada escuela.

- Inclusión de los datos de un profesor
- Modificación de los datos de un profesor (se puede cambiar el nombre y los apellidos del profesor, pero no el número de cédula)
- Recuperación de los datos de un profesor, según su número de cédula
- Recuperación de la lista de profesores de una escuela.
- Asignación y desasignación de profesores a un curso.
- Recuperación de la lista de cursos asignados a un profesor (carga académica)
- Consulta de la lista de profesores de un curso.
- Consulta de la lista de cursos indicando los profesores (esta es una modificación de las opciones de la parte anterior)
- Recuperación de la lista de escuelas con el detalle de los cursos indicando los profesores (esta es una modificación de las opciones de la parte anterior).

Se incluirán también las opciones de registro y control de la información sobre los estudiantes.

- Inclusión de estudiantes. Para cada estudiante, se registrará su número de cédula, número de carnet, nombre, apellidos, nacionalidad y porcentaje de beca (cuando sea aplicable). Se usará una clase diferente para representar los datos de cada tipo de estudiante.
- Recuperación de los datos de un estudiante (por número de cédula o carnet)
- Modificación de los datos básicos del estudiante (nombre, apellidos y nacionalidad)
- Matrícula del estudiante en varios cursos
- Modificación de la información de matrícula (inclusión y exclusión)
- Cálculo de los aranceles para cada estudiante individual (indicando cursos y créditos por curso) y en general (indicando créditos totales y detalles de cálculo: becas y recargos)
- Obtención de las listas de curso (estudiantes matriculados)
- Obtención de la lista de cursos por estudiante.

Parte 4

Descripción

En esta entrega, se deben implementar de manera general todas las clases e interfaces del programa, usando programación genérica (plantillas). En especial, el manejo de todos los datos agrupados (colecciones), se hará de manera generalizada, es decir, no habrá colecciones específicas para implementar conjuntos particulares. Se implementarán los conjuntos (clases compuestas o agregadas) por medio de delegación a colecciones genéricas. Se emplearán colecciones diferentes si tienen una estructura, funcionalidad o usan técnicas de almacenamiento diferentes. Por ejemplo, pueden haber clases genéricas para manejar un arreglo de almacenamiento contiguo y otras para implementar una lista enlazada, pero si dos conjuntos requieren la funcionalidad de una lista, deberán delegar su implementación en dicha lista genérica. Deberá utilizar iteradores y otros patrones de diseño para simplificar la estructura del programa.

Funcionalidad requerida

No hay funcionalidades nuevas que implementar. El objetivo principal de esta entrega es la simplificación de la estructura del programa según los principios y patrones de diseño estudiados.

Parte 5

Descripción

En esta entrega, todos los datos manejados por el programa deberán serializarse empleando un archivo (en modo binario o de texto)

Se implementarán también todos los reportes necesarios en el sistema, y se utilizarán excepciones para controlar los posibles errores en la ejecución del programa, asegurando la robustez de la aplicación.

En especial, deberá cuidarse que:

- El programa efectúe correctamente las operaciones de lectura y escritura en los archivos
- Todos los formatos de los datos de entrada se verifiquen adecuadamente (validaciones)
- El manejo de memoria (asignación y desasignación) sea controlado efectivamente.
- El programa verifique la consistencia (integridad) de los datos que maneja.

Funcionalidad requerida

Los datos del sistema deberán hacerse persistentes, en esta última parte, por medio del uso de archivos. El programa permitirá especificar (ya sea para cargar o guardar los datos) el nombre del archivo a utilizar.

Las operaciones que la aplicación debe implementar en esta última parte son:

- Registro de notas finales para los estudiantes, en cada uno de los cursos matriculados. El registro de notas se hará por grupo, en lugar de seleccionar cada estudiante individualmente.
- Reporte de notas por grupo.
- Reporte de notas por estudiante.
- Resguardo de los datos capturados por el sistema utilizando un archivo (de texto o binario). Se solicita guardar los datos en un único archivo, que será seleccionado por el usuario al iniciar el programa, como se indicara antes.
- Recuperación automática de los datos del archivo al iniciar el programa. Para poder revisar el programa, se incluirá un archivo con datos de ejemplo a la hora de entregar el proyecto.