Universidad del Valle de Guatemala
Programación Orientada a Objetos
Sección 21

Javier Anleu 17149
Odalis Reyes 17032

Laboratorio 5
Polimorfismo

REQUISITOS FUNCIONALES

- El usuario debe de poder ingresar los datos de los estudiantes: número de DPI y el nombre.
- El usuario debe de poder ingresar los promedios de 1ero, 2do y 3ero básico (de los egresados de secundaria). Asimismo, debe de ingresar las notas del examen de matemática, historia y español.
- Para los aspirantes desvinculados de secundaria, el usuario debe de ingresar los datos del inciso (3); además, debe de ingresar la nota del examen de aptitud.
- Para los egresados de bachillerato, el usuario debe de ingresar el promedio de 4to y 5to bachillerato, y también debe de ingresar la nota del examen de historia.
- Para los aspirantes desvinculados de bachillerato, el usuario debe de ingresar los mismos datos del inciso (4).
- Permite modificar los datos de los aspirantes o eliminar a un aspirante.
- Poder mostrar los datos de un aspirante específico (se utilizará el puesto en el que están en el escalafón)
- El programa debe de calcular el escalafón de todos los aspirantes, y estos se deben de mostrar de mayor a menor (según las notas).
- • Mostrarle al usuario si las notas de los aspirantes desvinculados graduados de secundaria son mayores a un número establecido por el usuario.
- Mostrarle al usuario si el 50% de los aspirantes desvinculados graduados de bachillerato tienen un promedio mayor a 80.

PROPUESTA DE SOLUCIÓN

- Se usará una clase de herencia para los estudiantes, de nombre Aspirer. Tiene como atributos el nombre, número de PDI, tipo de estudiante, si es desvinculado o no, y la nota de historia.
- Se agrega un atributo de tipo ObjectId en la clase madre, de este modo se podrán guardar todos los datos de los estudiantes en una base de datos.
- Las clases hijas de Aspire serán Bachelor y Secondary. En Secondary, se tiene como atributos el promedio de las notas de 1ero a 3ero básico (este será como un solo atributo, tanto para esta clase como para Bachelor. Dependiendo del tipo de estudiante, se pedirá las notas de básico o de bachillerato); asimismo, las notas del

examen de matemática, historia y español. En Bachelor tendrá como atributos el promedio de las notas de 4to y 5to, y las notas del examen de historia.

- Por medio de polimorfismo, se creará una clase que tenga como método calcular los promedios para el escalafón (no importa si es secundaria o bachillerato).
- En una nueva clase tendrá como métodos agregar, modificar y eliminar un estudiante. Esta misma clase servirá para organizar los estudiantes de mayor a menor (según las notas), tendrá los métodos para calcular si el dato ingresado por el usuario es mayor al promedio de las notas de los aspirantes desvinculados de secundaria. Y por último, se calculará si el 50% de los aspirantes desvinculados de bachillerato tienen un promedio mayor a 80 puntos.

IDENTIFICACIÓN DE CLASES

Execute: Será capaz de interactuar con el usuario; sin embargo, no debe de realizar ninguna operación. Aquí se deberá de importar la clase Scanner para que se puedan leer los datos de las dos clases antes mencionadas.

GUI: Esta clase se hará la interfaz gráfica. Es el medio por el cual el usuario ingresará los datos y le solicitará al controlador que realice los cálculos y le muestra la información deseada. Tiene los siguientes eventos:

- Ingreso de un aspirante
- Muestra los datos.
- Modifica un dato de uno de los aspirantes
- Muestra los datos de un aspirante
- Calcula si el 50% de los graduados de bachillerato desvinculados tienen un promedio mayor a 80 pts.
- Calcula si el promedio de los aspirantes desvinculados graduados de secundaria es mayor a un número dado por el usuario

Aspirer: Es una clase que actúa como la clase padre de los aspirantes. Sus atributos son el ID para la base de datos (ObjectID), el nombre del aspirante (String), el número de DPI (String), el tipo de aspirante (boolean, true si es graduado de bachillerato, false si es graduado de secundaria) si es desvinculado o no (boolean, true desvinculado, false recién graduado), el promedio de sus últimos años de estudio (double), la nota obtenida en el examen de historia (double), y la nota obtenida para el escalafón (double). Su constructor únicamente incluye el nombre, DPI, tipo, boolean para saber si es desvinculado, su promedio en los últimos años de estudio, y la nota del examen de historia. Sus métodos son getters y setters para los atributos

Nota: Es una interfaz y tiene como propósito implementar los métodos a las demás clases que estén relacionadas con esta mediante polimorfismo.

- *Calcular:* Es un método abstracto que servirá para calcular las notas del escalafón.

Secondary: Es la clase que se utiliza para modelar a los aspirantes graduados de secundaria, y es heredada de Aspirer. Tiene como atributos además la nota del examen de matemática, de español, y la del examen de aptitud. En esta se genera un constructor, el cual incluye el constructor de la clase padre con super y asigna además los valores de las notas del examen de español, matemática, y de aptitud (en el caso que no sea desvinculado se coloca 0). Se heredan los getters y setters, y tiene un toString. En el caso del método escalafon, se definirá de la siguiente manera:

1. if (desvinculado==false)
    a. double averagetests = (GetNotaMatematica()+getNotaHistoria()+getNotaEspanol())/3;
    b. double grade = averagetests*0.4 + average*0.6;
    c. Se le aplica DecimalFormat a grade, y se le hace un parseDouble. Este valor es el que se devuelve.
2. Else
    a. double averagetests = (GetNotaMatematica()+getNotaHistoria()+getNotaEspanol())/3;
    b. double grade = (averagetests*0.6+average*0.4 + getNotaAptitud())/2;
    c. Se le aplica DecimalFormat a grade, y se le hace un parseDouble. Este valor es el que se devuelve.

Bachelor: Es la clase hija de Aspirer y tiene como propósito modelar los datos necesarios para calcular el escalafón de un estudiante de bachillerato, asimismo, heredará sus atributos. Además, tendrá un toString de la clase.

1. If (desvinculado==false)
    a. double grade = average*0.6 + getNotaHistoria()*0.4;
    b. Se le aplica DecimalFormat a grade y se le hace un parseDouble. Este valor es el que se devuelve.
2. Else
    a. double grade = average*0.4 + getNotaHistoria()*0.6;
    b. Se le aplica DecimalFormat a grade y se le hace un parseDouble. Este valor es el que se devuelve.

Evaluator: Esta clase tiene como propósito usar los datos de Aspirer (esto implica que usará los datos de las clases hijas también por medio de polimorfismo). Además, en esta clase se conectara la base de datos para que de esta manera, los métodos de la clase trabajen con ello. Se crearán los Querys necesarios para poder manejar la información de manera correcta. Se hará un solo atributo, y este servirá para organizar a los estudiantes en el escalafón; además, se implementarán varios métodos y estos son:

- *Organize*: es el método para ordenar la base de datos de mayor a menor en base a la nota del escalafón. Es un void y devuelve el ArrayList.

- *AddSecondary*: es el método para agregar un aspirante graduado de secundaria a la base de datos. Tiene como parámetros los atributos de la clase, genera el objeto de tipo Secondary, y lo agrega a la lista. Es un void.
- *AddBachelor*: es el método para agregar un aspirante graduado de bachillerato al ArrayList. Tiene como parámetros los atributos de la clase, genera el objeto de tipo Bachelor, y lo agrega a la lista. Es un void.
- *DisplayAspirers*: es el método para mostrar todos los aspirantes. Muestra su posición en la base de datos, su nombre completo, su tipo (de secundaria o bachillerato y desvinculados o no), y la nota que obtuvieron en el escalafón. Se muestra en el orden según el método Organize con un ArrayList.
- *FindAspirer*: es el método para encontrar un aspirante específico. Utilizará el número que aparece en pantalla al mostrar los aspirantes al usuario como parámetro, pero le restará uno para utilizarlo como índice en el ArrayList local para obtener el Id de la base de datos y obtener el objeto como Aspirer.
- *ModifySecondary*: es el método para modificar un atributo del aspirante graduado de secundaria. Tiene como parámetros la posición del aspirante (se usa FindAspirer), un int para saber qué modificación quiere hacer, y el resto de parámetros son los atributos de Secondary. Trabajando en conjunto con el GUI, se llenarán los parámetros. Dependiendo del que se quiera modificar, a ese se le asignará un valor y a los demás, en caso de ser String, se colocará "", en caso de ser numéricos 0, y en caso de ser booleans false. Con un if, se modificará el atributo específico que se quiere cambiar y se modifica con los métodos de MongoDB para modificarlo en la base de datos. En el parámetro que indica qué atributo se modifica, 1 es el nombre, 2 el DPI, 3 el tipo, 4 si es desvinculado, 5 la nota del examen de historia, 6 la nota del examen de matemática, 7 la nota del examen de español, y 8 la nota del examen de aptitud.
- *ModifyBachelor*: es el método para modificar un atributo del aspirante graduado de secundaria. Tiene como parámetros la posición del aspirante (se usa FindAspirer), un int para saber qué modificación quiere hacer, y el resto de parámetros son los atributos de Bachelor. Trabajando en conjunto con el GUI, se llenarán los parámetros. Dependiendo del que se quiera modificar, a ese se le asignará un valor y a los demás, en caso de ser String, se colocará "", en caso de ser numéricos 0, y en caso de ser booleans false. Con un if, se modificará el atributo específico que se quiere cambiar y se modifica con el setter. En el parámetro que indica qué atributo se modifica, 1 es el nombre, 2 el DPI, 3 el tipo, 4 si es desvinculado, y 5 la nota del examen de historia.
- *DeleteAspirer*: es el método que se utiliza para eliminar un aspirante de la lista. Al igual que FindAspirer, usa el número de la posición en el escalafón, le resta uno, y de esa forma lo encuentra en el ArrayList local y obtiene el ID para la base de datos. Con el ID, ya hace el remove.
- *BetterThan*: es el método que se utiliza para ver si el promedio de notas de los aspirantes desvinculados graduados de secundaria es mayor a un valor dado por el usuario. Su parámetro es el valor que quiere usar el usuario. Primero calcula el promedio de todos los aspirantes desvinculados graduados de secundaria, y luego lo

compara con el valor dado. Si es mayor, devuelve un String que dice “si”. De lo contrario, devuelve un String que dice “no”.

- *High80*: es el método que se utiliza para ver si el 50% de los aspirantes desvinculados graduados de bachillerato tienen un promedio mayor a 80. Primero cuenta los estudiantes desvinculados graduados de bachillerato, y luego cuenta cuántos tienen un promedio de notas de sus últimos años de estudio mayor a 80. Realiza la división y si es mayor a 0.5, devuelve un String que dice “si”. De lo contrario, devuelve un String que dice “no”.