TRABAJO FINAL DESARROLLO DE ALGORITMOS

Robles Lopez - Francisco Nicolás – FAI 3869 – LCC

Estructuras Utilizadas

- Se utilizó una clase llamada Alumnos para la creación de los objetos, en donde se definieron sus atributos (Nombre, Apellido, legajo, grado, promedio), su constructor, métodos get y métodos para realizar distintas operaciones.

Métodos:

Método toString (): El método toString () devuelve una representación en forma de cadena de los atributos del objeto Alumnos.

Método verificaCompuesto(): Es un método privado que verifica si una cadena está compuesta solo por letras (Usando ASCII). Este método se utiliza en el método esValido() para verificar la validez del nombre y apellido del alumno.

Método esValido(): Este método verifica si nombre, apellido, promedio y legajo cumplen ciertas condiciones para considerar al alumno válido. Retorna true si el alumno es válido y false en caso contrario.

Método equals (): Este método compara el legajo de un alumno con los legajos de los alumnos desaprobados (almacenados en un arreglo). Retorna verdadero si se encuentra el alumno dentro del arreglo, sino retorna falso.

Método noRepite (): El método utiliza dos bucles while anidados para recorrer la matriz grados y buscar si el legajo del objeto actual se repite en algún otro objeto de la matriz. Se comienza verificando si el objeto actual no es nulo y si el legajo coincide con this.legajo. Si se encuentra una coincidencia, se establece la variable seRepite en true sino la variable seRepite permanece en false.

Método cantLegajos(): Este método cuenta la cantidad de legajos que tiene un archivo.txt (En este caso la ListaDesaprobados).

Método cargaDesaprobados(): El método se encarga de cargar un arreglo con todos los legajos que hay cargados en un archivo.txt (ListaDesaprobados)

- Se usó una matriz para almacenar los objetos tipo “Alumnos”, en donde las filas representaban los grados y las columnas representaban a la capacidad de un grado (Los alumnos se cargaban en las columnas).

- Se recurrió a utilizar tres arreglos en donde uno de ellos almacenaría aquellos alumnos que estuvieran en el último grado y superen un cierto promedio, para luego hacer uso de esa información en otra instrucción (Los egresados), otro se encargaría de buscar las posiciones del carácter “;” para separar la información de un alumno e irlas registrando en la variable “UnAlumno” y el ultimo carga aquellos legajos de alumnos desaprobados.

Funcionamiento de pasar de grado

El método “aumentarGrado” recibe una matriz de alumnos grados y un arreglo de alumnos egresados. El objetivo es reorganizar los alumnos en función de sus promedios y grados, aumentando el grado de aquellos que aprueban y moviendo a los egresados al arreglo correspondiente.

Funcionamiento:

- Se declaran variables locales para realizar el seguimiento de las posiciones y contar el número de egresados.

- Se crea un arreglo desaprobados en donde se utiliza el método cantLegajos () de la clase Alumnos para asignarle longitud al arreglo y luego se carga con los legajos de los alumnos desaprobados mediante el método cargarDesaprobados.

- Se crea una nueva matriz newGrado de tamaño fijo (7 filas y 30 columnas) para almacenar los alumnos (Las filas representan los grados y las columnas los alumnos).

- Se utilizan bucles for y while para recorrer la matriz de alumnos grados de forma descendente.

- Para cada alumno en una posición no nula de la matriz se verifica si el alumno está en la lista de desaprobados mediante el método equals(). Si es así, se asigna a la posición correspondiente en newGrado según su grado actual.

- Si el alumno no está en la lista de desaprobados y su promedio es mayor o igual a 6 y aún no se han egresado alumnos, se asigna al arreglo egresados y se incrementa la variable itEgresados.

- Si su promedio es mayor o igual a 6 y ya se han egresado alumnos, se asigna a la posición correspondiente en newGrado según su grado actual y se incrementa la variable correspondiente a ese grado.

Y finalizaría retornando la matriz newGrado la cual sería asignada a “grados”.

Modularización y Reutilización de código

Se usó de códigos anteriores y códigos de practica:

- Carga de matrices (con algunos cambios).
- carga de arreglos.
- Métodos de ordenamiento (Con algunos cambios).
- La lectura de archivos.
- Verificaciones de objetos.
- Contador de espacios vacios.

Modularizacion:

Cada módulo del codigo se encarga de hacer una tarea específica:

DevuelvePosAlum: Este método como su nombre lo indica se encarga de ubicar a un alumno en un grado solicitado y retorna un String indicando si no se encontró o dando la posicion de donde se encuentra.

vacantesTotales: Este modulo suma la cantidad de estudiantes de cada grado para luego retornar la suma total (Lugares ocupados) y restarlo con la capacidad total que tienen todos los grados.

MostrarEGresados: Muestra a los egresados en forma descendente (Usando el ordenamiento Burbuja).

CargarGrado: Este módulo sirve para que cargar un arreglo con alumnos de un grado solicitado.

OrdenarAlumnosBm: Este módulo ordena el arreglo creado con “cargarGrado” de manera ascendente (Usando ordenamiento Burbuja Mejorada).

MostrarAlum: Se encarga de mostrar la información del arreglo ya ordenado. (este módulo se encargar de llamar a “CargarGrado” y “OrdenarAlumnosBm”).

calculaProm: Calcula el promedio de un grado.

aumentarGrado: Este módulo aumenta el grado de aquellos alumnos que cumplan ciertas condiciones y los que no, se mantienen en su grado. Al terminar de registrar a los alumnos, retorna una matriz con la nueva organización.

cargarAlumnos: Este módulo lee los alumnos de un archivo.txt y se encarga de pasar esa información a un objeto tipo Alumno para poder cargarlo a la matriz “grados”. (Usa al módulo “crearAlumno” para que la información leída del archivo sea un objeto Alumno).

cantAlumnos: Cuenta la cantidad de alumnos de los grados y retorna el resultado.

Valores Nulos

- Como se puede ver en el codigo cada vez que se recorría un grado, el limite era que la posicion en la que se encuentre el iterador sea distinto de nulo para poder ejecutar otra serie de instrucciones, ya que si la posicion era nula significaba que no había más alumnos dentro del grado y no tendría sentido recorrer toda la matriz sabiendo que no habría más alumnos.

- A la hora de realizar los ordenamientos de un arreglo verificaba si las posiciones del iterador eran nulos, ya que comparar a un alumno con nada no sirve.