

Guillermo Arredondo Renero

C.U. 000197256

PROYECTO FINAL ALGORITMOS: Programación Orientada a Objetos

GIMNASIO

Grupo 003 (LICOM11101-003)

Otoño 2020

ALGORITMOS Y PROGRAMAS (LICOM11101)

María de Lourdes Alicia Domenzain Ortega

ÍNDICE

**Introducción**

Presentación del problema a resolver

**Planificación del Proyecto**

Análisis de resolución

Diagramas UML

**Conversión a Programación en JAVA**

Códigos en JAVA

Clases útiles

Clases de proyecto

Main de pruebas

Clases interfaces

Pruebas de solución y Cambios correspondientes

**Manual de Usuario para Interfaz Gráfica**

Descripción de diseño de Interfaz Gráfica

**Conclusiones**

**Introducción**

En general, un gimnasio de todo tipo de deportes presenta el problema de no poder manejar las grandes cantidades de información que se generan debido al amplio número de alumnos inscritos en él. Dado que yo practico gimnasia y el gimnasio en el que estoy inscrito es muy grande me surgió la idea de poder crear un programa de Interfaz Gráfica de Usuario que pudiera ayudar al registro y control de los diversos gimnastas que atienden a entrenamientos en él. Por mi propia experiencia, para los organizadores e incluso para los competidores de los gimnasios es complicado y tedioso llevar la cuenta y registro de los diversos eventos, movimientos y nuevos logros que se adquieren a lo largo de su trayectoria deportiva.

Por lo tanto, este gimnasio se piensa que entre los requerimientos se desea conocer qué gimnastas están registrados y tener una tabla con sus nombres, registrar e identificar el nivel correspondiente de cada gimnasta de acuerdos a los movimientos gimnásticos que realice y sus respectivas dificultades, así como las competencias en las que a participado. Asimismo, se piensa en la importancia de los procesos estadísticos del gimnasio, para poder conocer los horarios, las edades, estilos, tiempo haciendo gimnasia, las competencias a las que ha asistido, entre otros datos. Buscando solucionar estos problemas, la idea de este proyecto es poder agilizar y simplificar el manejo de los datos ingresados a un gimnasio, desde inscripciones de nuevos gimnastas hasta el cálculo de las proteínas que debe consumir durante el día para mantenerse fuerte y evitar lesiones.

**Planificación del Proyecto**

Una vez que el gimnasio desea conocer los datos respecto a los gimnastas inscritos se considera establecer dos clases bases: por un lado, la clase Gimnasio será la que represente al gimnasio cualquiera que desee conocer la información, el cual entre sus características principales contará con una matriz dentro de la cual se establecerán los gimnastas que formen parte del gimnasio (clase Gimnasta). Como en todo gimnasio deportivo es importante poder distinguir entre los diversos grupos de competidores que estén inscritos en él. Dado a que este gimnasio está diseñado para gimnastas y similares propiamente, es importante informar que se diseñará una separación por grupos de acuerdo al nivel gimnástico con el que cuentan los ingresados. La investigación demuestra que los gimnastas son calificados en niveles del 1 al 10 y agregando el nivel FIG (que representa gimnastas que cuentan con la autorización, habilidad y requerimientos para participar en eventos de talla mundial en representación de sus países). Más adelante se explicará el funcionamiento del cálculo de nivel y como se utilizará en el programa.

Por otro lado, buscando solucionar el registro de logros y propiciando la motivación de los atletas al conocer sus logros a lo largo de ciertos periodos (considerados de 3 años para poder agilizar las búsquedas), se planea que dentro de la clase Gimnasta, además de establecer las características propias del gimnasta, agregar dos arreglos que contengan la información de movimientos y competencias presentadas por cada gimnasta. Asimismo, como atleta reconozco la importancia de comer saludable y mantener una dieta correcta para evitar lesiones e incrementar tu progreso. Por lo tanto, considero de suma importancia agregar el cálculo de una dieta. Dado a que un gimnasta debe estar fuerte para poder llevar a cabo los movimientos exigentes de gimnasia, considero a las proteínas el grupo alimenticio más importante. Es necesario investigar, cuál podría ser una correcta proporción para un atleta, pero también es necesario aclarar que este cálculo deberá ser un tanto general, puesto que cada cuerpo es diferente. Debido a ello, se considera apropiado que el método devuelva el valor medio recomendado, para proporcionar un rango propicio para cualquier atleta bajo ciertas características (en especial atención al peso).

Por último, la clase competencia será la primera en crearse y debe poder representar las características principales de una competencia deportiva. Des este modo, se considera importante el nombre de la competencia, el lugar y el año en que se llevó a cabo, la categoría que representa (pensado para representar en qué categoría participó el atleta) y el logro obtenido por el competidor (no forzosamente medalla). La clase movimiento gimnástico deberá ser un poco más complicada, puesto que hay muchas características que definen un movimiento, además que pueden existir movimientos muy similares. De hecho, un movimiento gimnástico se define por dos características, su nombre y su posición de ejecución, pero una vez que la posición abarca un conjunto mayor, se considera el nombre como elemento de comparación.

Ahora bien, pensando propiamente es muy difícil perder una habilidad, por lo que no se considera que puedas dejar de hacer un movimiento (baja movimiento), por esto mismo, no se agrega la posibilidad. Por el contrario, las competencias son pensadas que cuentan con un registro previo y una asistencia, de este modo, las competencias pueden darse de alta, baja y, además, agregar un logro obtenido. Así, en métodos de contar competencias se considera como inasistencia el no tener logro en una competencia (representado con el número 0, pues así se instancia una competencia).

**Diagramas UML**

Clases Útiles anexadas al proyecto para facilitar algoritmos

**Manejador Arreglos Genéricos**

|  |
| --- |
| **ManejadorArreglosGenericos** |
|  |
| **+**posMayor (T[], int) <T extends Comparable <T>> : int  **+**posMenor (T[], int) <T extends Comparable <T>> : int  **+**cuantosMayorX (T[], int, T) <T extends Comparable <T>> : int  **+**cualesMayX (T[], int, T) <T extends Comparable <T>> : ArrayList <Integer>  +cuantosMenX (T[], int, T) <T extends Comparable <T>> : int  **+**cualesMenX (T[], int, T) <T extends Comparable <T>> : ArrayList<Integer>  **+**swap (T[], int, int) <T>  +correKDerecha (T[], int, int) <T>  +correKIzquierda (T[], int, int) <T>  +unCorrimientoDer (T[], int, int) <T>  +unCorrimientoIzq (T[], int, int) <T>  +posMinimo (T[], int, int) <T extends Comparable <T>> : int  +selecDirecta (T[], int) <T extends Comparable <T>>  +buscaSecDesorden (T[], int, T) <T> : int  +buscaSecOrdenada (T[], int, T) <T extends Comparable <T>> : int  +buscaBinaria (T[], int, T) <T extends Comparable <T>> : int  +inserta (T[], int, int, T)  +insertaEnOrden (T[], int, T) <T extends Comparable <T>> : int  +insertaFinal (T[], int, T) <T> : int  +insertaInicio (T[], int, T) <T> : int  +elimina (T[], int, int) <T> : int  +eliminaEnOrden (T[], int, T) <T extends Comparable <T>> : int  +eliminaEnDesorden (T[], int) <T> : int |

**Manejador Matrices Genérico**

*Esta clase no se pudo utilizar debido a que la matriz instanciada dentro de la clase Gimnasio no cumple el requisito de estar siempre llena, o como mínimo tener el mismo número de elementos registrados en cada uno de sus renglones; por lo tanto, se recurrió a una trasposición de los métodos y se aprovechó el doble el uso de ManejadorArreglosGenericos.*

|  |
| --- |
| **ManejadorMatricesGenerico** |
|  |
| **+**posMenorCol (T[][], int, int) <T extends Comparable <T>> : int  +posMayorCol (T[][], int, int) <T extends Comparable <T>> : int  +posMenorRen (T[][], int, int) <T extends Comparable <T>> : int  +posMayorRen (T[][], int, int) <T extends Comparable <T>> : int  +posMenor (T[][], int, int) <T extends Comparable <T>> : int[]  +posMayor (T[][], int, int) <T extends Comparable <T>> : int[]  +esSimetrica (T[][], int) <T> : boolean  +buscaRenglonDesordenado (T[][], int, int, T) <T> : int[]  +buscaRenglonOrdenado (T[][], int, int, T) <T extends Comparable <T>> : int[]  +buscaColumna (T[][], int, int, T) <T> : int  +buscaEnMatriz (T[][], int, int, T) <T> : int[]  +matEqualsOther (T[][], int, int, T[][], int, int) <T> : boolean |

Clases propias del proyecto Gimnasio

**Competencia (para arreglos en Clase Gimnasta)**

|  |
| --- |
| **Competencia** |
| **-**nombre: String  -lugar: String  -anioFecha: int  -logro: int  -categoria: String |
| + Competencia(String)  + Competencia(String, String, String, int)  + Competencia(String, String, String, int, int)  + getNombre():String  + getLugar(): String  + getAnioFecha(): int  + getLogro(): int  + getCategoria(): String  + setCategoria(String)  + setNombre(String)  + setLugar(String)  + setAnioFecha(int)  + setLogro (int):  + compareTo (Competencia): int  + hashCode (): int  + equals (Object): boolean  + toString (): String |

**Movimiento de Gimnasia (para arreglo en clase Gimnasta)**

|  |
| --- |
| **MovimientogGym** |
| - nombre: String  - aparato: String  - posicionEjecucion: String  - dificultad: char  - obligatorio: boolean |
| **+**MovimientoGym (String)  +MovimientoGym (String, String, String, char, boolean)  +getNombre (): String  +getAparato (): String  +getPosicionEjecucion (): String  +getDificultad (): char  +isObligatorio (): boolean  +setNombre (String)  +setAparato (String)  +setPosicionEjecucion (String)  +setDificultad (char)  +setObligatorio (boolean)  +compareTo (MovimientoGym): int  +hashCode (): int  +equals (Object): boolean  +toString (): String |

**Gimnasta**

|  |
| --- |
| **Gimnasta** |
| **-** nombre: String  - curp: String  - edad: int  - genero: char  - antiguedad: int  - estilo: String  - horario: int  - peso: double  - movimientos: MovimientoGym[]  - competencias: Competencia[]  - numCompetencias: int  - numMovimientos: int  - MAX\_MOVIMIENTOS: int  - MAX\_COMPETENCIAS: int |
| + Gimnasta (String)  + Gimnasta (String, String, int, String, int, double)  + getNombre (): String  + getCurp (): String  + getGenero (): char  + getAntiguedad (): int  + getEstilo (): String  + getHorario (): int  + getEdad (): int  + getPeso (): double  + getNumCompetencias (): int  + getNumMovimientos (): int  + getPrimeraCompetenciaAnio (int): String  + getUnaCompetencia (String): String  + getUnMovimiento (String): String  + getListaMovimAparato (String): String  + setNombre (String)  + setGenero (char)  + setAntiguedad (int)  + setEstilo (String)  + setHorario (int)  + setPeso (double)  + setLogroCompetencia (String, int): String  + altaMovimiento (String, String, String, char, boolean): boolean  + altaCompetencia (String, String, String, int): boolean  + bajaCompetencia (String): String  + numMovimientosAparato (String): int  + numMovimientosDificultad (char): int  + numMovimientosObligatorios (): int  + mayDificultad (): char  + calculaNivel (): int  + calculaProteinas (boolean): double  + participoCompetencia (String): boolean  + conMovimiento (String): boolean  + movimientoMasDificil (): String  + mejorAparato (): String  + mejorCompetencia (): String  + compareTo (Gimnasta): int  + hashCode (): int  + equals (Object): boolean  + toString (): String |

Métodos a destacar:

*calculaNivel()*, este método es utilizado por el método *actualizaGym()* de la clase Gimnasio, lo que hace es desplegar una serie de condiciones a través de las cuales se evalúa a cada gimnasta, este método no es del todo apegado a la realidad por satisfacer practicidad en el método y poder alcanzar cualquiera de los niveles. Explicación:

En gimnasia, como se dijo anteriormente, existen niveles del 1-10 y se añade FIG; sin embargo, entre los niveles 1-5 (dependiendo el gimnasio) son considerados niveles que fomentan la recreación y la motivación de los gimnastas para seguir avanzando. Es hasta el nivel 6 con el inicio de las competencias serias en las cuales se empieza a considerar los movimientos y la dificultad ejecutable. Como tal los jueces y tus profesores son quienes establecen que nivel eres correspondiente a la buena ejecución y la dificultad de movimientos gimnásticos, existe una tabla de requisitos acorde a la cual se presentaron las condiciones en el método. Para facilitar la elaboración de este método se decidió implementar tres métodos extra que son utilizados por él para calculas la mayor dificultad de movimientos, el número de movimientos obligatorios con los que cuenta y el número de movimientos de una determinada dificultad con los que cuenta.

Se reitera que no existe el método para dar de baja movimientos debido a que se considera inapropiado y un tanto extraño que algo así pueda existir. Son muchos los casos posibles, pero en su mayoría un atleta no olvida lo que alguna vez aprendió.

**Gimnasio**

|  |
| --- |
| **Gimnasio** |
| - nombre: String  - ubicacion: String  - encargado: String  - competidores: Gimnasta[][]  - NIVELES: String[]  - numCompsNivel: Integer[]  - MAX\_COMPETIDORES: int  - MAX\_NIVELES: int |
| + Gimnasio()  + Gimnasio(String, String, String)  + isGimnasioVacio(): boolean  + getNombre(): String  + getUbicacion(): String  + getEncargado(): String  + getMAX\_NIVELES(): int  + getNumCompsNivel(int): int  + getNumCompsNivel(String): int  + getUnGimnasta(String): String  + getUnGimnasta(int, int): String  + getUnMovimientoGimnasta(String, String): String  + getUnaCompetenciaGimnasta(String, String): String  + getProteinasGimnasta(String, boolean): double  + getMovMasDificilGimnasta(String): String  + getGimnastasNivel(String): String  + getGimnastasNivel(int): String  + getListaMovimientosGimnastaAparato(String, String): String  + setNombre(String)  + setUbicacion(String)  + setEncargado(String)  + setLogroCompetenciaGimnasta(String, String, int): String  + altaGimnasta(String, String, int, String, int, double): boolean  + altaMovimientoGimnasta(String, String, String, String, char, boolean): boolean  + altaCompetenciaGimnasta(String, String, String, String, int): boolean  + bajaCompetenciaGimnasta(String, String): String  + bajaGimnasta(String): String  + actualizaGimnasio()  + cuantosGimnastasConMovimiento(String): int  + gimnastasHorario(int): int  + gimnastasEdad(int): int  + cuantosGimnastasEnCompetencia(String): int  + gimnastaMayDificul(): String  + cuantasExpertiseAparato(String): int  + cuantasEstilo(String): int  + cuantasAntiguedad(int): int  + mejoresCompetenciasGimnastasNivel(String): String  + hashCode(): int  + equals(Object): boolean  + compareTo(Gimnasio): int  + toString(): String |

Métodos a destacar:

*actualizaGimnasio():*este método hace uso del método calcula nivel de cada gimnasta y lo compara con el correspondiente nivel en el que se encuentra en ese momento, en caso de ser diferentes se guarda en una variable auxiliar, se elimina del nivel en el que se encontraba y se inserta en el nivel correspondiente.

*gimnastaMayDificul():* Establece una búsqueda de cual es el gimnasta con mayor dificultad en todos sus movimientos, esto hace uso de *mayDificultad()* de la clase Gimnasta y recorre uno por uno cada gimnasta buscando cual es el que tenga una mayor dificultad.

*get* *getListaMovimientosGimnastaAparato():* Este método llama a su correspondiente *getListaMovimientosAparato()* en clase Gimnasta, genera un ArrayList de String con el nombre de todos los movimientos que cumplan el requisito del Aparato de entrada. El toString del ArrayList se traspasa al método en clase Gimnasio.

Clase de Interfaz Gráfica

|  |
| --- |
| **GimnasioVista** |
| # pantallaGral : JTextArea  # leerArchivo: JButton  # tablaDataBt : JButton  # nutricionBt : JButton  # altaLogrosBt : JButton  # altaGimnastasBt : JButton  # actualizaGymBt : JButton  # infoComplBt : JButton  # buscaGimnasBt : JButton  # bajaCompBt : JButton  # bajaGimnastaBt : JButton  # mayDificultad : JButton  # expAparato : JButton  # mejoresCompetenciasNivel : JButton  # gimNivel : JButton  # cuantosMovimiento : JButton  # cuantosCompetencia : JButton  # horario : JButton  # edad : JButton  # estilo : JButton  # antiguedad : JButton  # logroCompBt : JButton  # numComp : JButton  # buscaLogro : JButton  # inicioLb : JLabel  # estadCuantos : JLabel  # nutricionTxtF : JTextField  # estadTxtF : JTextField  # inscripcionesTxtF1 : JTextField  # inscripcionesTxtF2 : JTextField  # inscripcionesTxtF3 : JTextField  # inscripcionesTxtF4 : JTextField  # logrosTxtF1 : JTextField  # pantallaGimnastas : JTextField  # logrosTxtF2 : JTextField  # logrosTxtF3 : JTextField  # logrosTxtF4 : JTextField  # logrosTxtF5 : JTextField  # nutMuscRdBt : JRadioButton  # obligatorioBt : JRadioButton  # movBt : JRadioButton  # compBt : JRadioButton  # listMovBt : JRadioButton  # movMasDificil : JRadioButton  # estilosCBox1 : JComboBox<String>  # estilosCBox2 : JComboBox<String>  # nivelesCBox1 : JComboBox<String>  # nivelesCBox2 : JComboBox<String>  # horarioCBox1 : JComboBox<String>  # horarioCBox2 : JComboBox<String>  # textoGrande : JScrollPane |
| +GimnasioVista(String)  +main(String[]) |

**Conversión a Programación en JAVA**

Códigos en JAVA

Clases Útiles:

Como se explicó anteriormente, la única clase auxiliar utilizada fue la de ManejadorArreglosGenericos, una vez que la matriz en Gimnasio no cumplía los requisitos necesarios para utilizar adecuadamente las búsquedas en matriz. Entre los diversos métodos de la clase ManejadorArreglosGenericos los más utilizados fueron *buscaSecDesorden*, *insertaFinal* y *eliminaEnDesorden*.

[Inicia código ManejadorArreglosGenericos]

import java.util.ArrayList;

/\*Guillermo Arredondo

\* 23 octubre 2020

\* METODOS DEL MANEJADOR ARREGLOS PARA CUALQUIER TIPO DE DATO

\*/

public class ManejadorArreglosGenericos {

public static <T extends Comparable<T>> int posMayor(T [] a, int n) {

T max=a[0];

int indiceMax=0;

for(int i=1; i<n; i++) {

if(max.compareTo(a[i])<0) {

max=a[i];

indiceMax=i;

}

}

return indiceMax;

}

public static <T extends Comparable<T>> int posMenor(T[] a, int n) {

T min=a[0];

int indiceMin=0;

for(int i=1; i<n; i++) {

if(min.compareTo(a[i])>0) {

min=a[i];

indiceMin=i;

}

}

return indiceMin;

}

public static <T extends Comparable<T>> int cuantosMayorX(T[] a, int n, T comp) {

int numMayores=0;

for(int i=0; i<n; i++) {

if(a[i].compareTo(comp)>0) {

numMayores++;

}

}

return numMayores;

}

public static <T extends Comparable<T>> ArrayList<Integer> cualesMayX(T[] a, int n, T comp){

ArrayList<Integer> lista=new ArrayList<Integer>();

for(int i=0; i<n; i++) {

if(a[i].compareTo(comp)>0) {

lista.add(i);

}

}

return lista;

}

public static <T extends Comparable<T>> int cuantosMenX(T[] a, int n, T comp) {

int numMenores=0;

for(int i=0; i<n; i++) {

if(a[i].compareTo(comp)<0) {

numMenores++;

}

}

return numMenores;

}

public static <T extends Comparable<T>> ArrayList<Integer> cualesMenX(T[] a, int n, T comp){

ArrayList<Integer> lista=new ArrayList<Integer>();

for(int i=0; i<n; i++) {

if(a[i].compareTo(comp)<0) {

lista.add(i);

}

}

return lista;

}

public static <T> void swap(T[] a, int i1, int i2) {

T aux=a[i1];

a[i1]=a[i2];

a[i2]=aux;

}

public static <T> void invierte(T[] a, int n) {

T aux;

for(int i=0; i<n; i++) {

n--;

aux=a[i];

a[i]=a[n];

a[n]=aux;

}

}

public static <T> void correKDerecha(T[] a, int n, int k) {

for(int j=1; j<=k; j++) {

for(int i=n-1; i>0; i--) {

a[i]=a[i-1];

}

a[0]=null;

}

}

public static <T> void correKIzquierda(T[] a, int n, int k) {

for(int j=1; j<=k; j++) {

for(int i=0; i<n-1; i++) {

a[i]=a[i+1];

}

a[n-1]=null;

}

}

public static <T> void unCorrimientoDer(T[] a, int n, int pos) {

for(int i=n; i>pos; i--) {

a[i]=a[i-1];

}

}

public static <T> void unCorrimientoIzq(T[] a, int n, int pos) {

for(int i=pos; i<n-1; i++) {

a[i]=a[i+1];

}

}

public static <T extends Comparable<T>> int posMinimo(T[] a, int n, int inicio) {

int indiceMin=inicio;

for(int i=inicio+1; i<n; i++) {

if(a[indiceMin].compareTo(a[i])>0) {

indiceMin=i;

}

}

return indiceMin;

}

public static <T extends Comparable<T>> void selecDirecta(T[] a, int n) {

int indiceMin;

for(int i=0; i<n; i++) {

indiceMin=posMinimo(a, n, i);

swap(a, indiceMin, i);

}

}

public static <T> int buscaSecDesorden(T[] a, int n, T busca) {

int i=0;

while(i<n && !a[i].equals(busca)) {

i++;

}

if(i==n) {

i=-1;

}

return i;

}

public static <T extends Comparable<T>> int buscaSecOrdenada(T[] a, int n, T busca) {

int i=0;

while(i<n && a[i].compareTo(busca)<0) {

i++;

}

if(i==n || !a[i].equals(busca)) {

i=-i-1;

}

return i;

}

public static <T extends Comparable<T>> int buscaBinaria(T[] a, int n, T busca) {

int res;

int inicio=0;

int fin=n-1;

int mitad=(inicio+fin)/2;

while(inicio<=fin && !a[mitad].equals(busca)) {

if(busca.compareTo(a[mitad])<0) {

fin=mitad-1;

}

else {

inicio=mitad+1;

}

mitad=(inicio+fin)/2;

}

if(fin<inicio) {

res=-inicio-1;

}

else {

res=mitad;

}

return res;

}

public static <T> int inserta(T[] a, int n, int pos, T x) {

if(n<a.length) {

unCorrimientoDer(a, n, pos);

a[pos]=x;

n++;

}

return n;

}

public static <T extends Comparable<T>> int insertaEnOrden(T[] a, int n, T x) {

int pos;

if(n<a.length) {

pos=buscaBinaria(a, n, x);

if(pos<0) {

pos=-pos-1;

unCorrimientoDer(a, n, pos);

a[pos]=x;

n++;

}

}

return n;

}

public static <T> int insertaFinal(T[] a, int n, T x) {

return inserta(a, n, n, x);

}

public static <T> int insertaInicio(T[] a, int n, T x) {

return inserta(a, n, 0, x);

}

public static <T> int elimina(T[] a, int n, int pos) {

unCorrimientoIzq(a, n, pos);

n--;

a[n]=null;

return n;

}

public static <T extends Comparable<T>> int eliminaEnOrden(T[] a, int n, T x) {

int pos;

pos=buscaBinaria(a, n, x);

if(pos>=0) {

unCorrimientoIzq(a, n, pos);

n--;

a[n]=null;

}

return n;

}

public static <T> int eliminaEnDesorden(T[] a, int n, T x) {

int pos;

pos=buscaSecDesorden(a, n, x);

if(pos>=0) {

unCorrimientoIzq(a, n, pos);

n--;

a[n]=null;

}

return n;

}

}

Clases del proyecto :

Competencia

Es una clase básica utilizada para generar un arreglo unidimensional como atributo de la clase Gimnasta.

[Inicia código Competencia]

/\*Guillermo Arredondo

\* 21 de noviembre de 2020

\* Clase Competencia para arreglo de clase Gimnasta

\*/

public class Competencia {

private String nombre;

private String lugar;

private int anioFecha;

private int logro;

private String categoria;

public Competencia(String nombre) {

this.nombre=nombre;

}

public Competencia(String nombre, String lugar, String categoria, int anioFecha) {

this.nombre=nombre;

this.lugar=lugar;

this.categoria=categoria;

this.anioFecha=anioFecha;

this.logro=0;

}

// el metodo siguiente no es utilizado, puesto que se usa en un método a parte

public Competencia(String nombre, String lugar, String categoria, int anioFecha, int logro) {

this.nombre=nombre;

this.lugar=lugar;

this.categoria=categoria;

this.anioFecha=anioFecha;

this.logro=logro;

}

public String getNombre() {

return nombre;

}

public String getLugar() {

return lugar;

}

public int getAnioFecha() {

return anioFecha;

}

public int getLogro() {

return logro;

}

public String getCategoria() {

return categoria;

}

public void setCategoria(String categoria) {

this.categoria = categoria;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setLugar(String lugar) {

this.lugar = lugar;

}

public void setAnioFecha(int anioFecha) {

this.anioFecha = anioFecha;

}

public void setLogro(int logro) {

this.logro = logro;

}

public int compareTo(Competencia otro) {

return this.nombre.compareTo(otro.nombre);

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((nombre == null) ? 0 : nombre.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Competencia other = (Competencia) obj;

if (nombre == null) {

if (other.nombre != null) {

return false;

}

} else if (!nombre.equals(other.nombre)) {

return false;

}

return true;

}

public String toString() {

StringBuilder cad=new StringBuilder();

cad.append("\n Competencia de Gimnasia de nombre: "+nombre);

cad.append("\n ubicada en: "+lugar);

cad.append("\n durante el año: "+anioFecha);

cad.append("\n de categoria: "+categoria);

cad.append("\n resultado: "+logro);

return cad.toString();

}

}

MovimientoGym

Una clase básica que sirve para crear un segundo arreglo unidimensional como atributo en la clase Gimnasta.

[Inicia código de clase MovimientoGym]

/\*Guillermo Arredondo

\* 21 de noviembre 2020

\* Clase movimiento gimnasitico para arreglo de clase Gimnasta

\*/

public class MovimientoGym {

private String nombre;

private String aparato;

private String posicionEjecucion;

private char dificultad; // la dificultad esta basado en letras

private boolean obligatorio;

public MovimientoGym(String nombre) {

this.nombre=nombre;

}

public MovimientoGym(String nombre, String aparato, String posicionEjecucion, char dificultad, boolean obligatorio) {

this.nombre=nombre;

this.aparato=aparato;

this.posicionEjecucion=posicionEjecucion;

this.dificultad=dificultad;

this.obligatorio=obligatorio;

}

public String getNombre() {

return nombre;

}

public String getAparato() {

return aparato;

}

public String getPosicionEjecucion() {

return posicionEjecucion;

}

public char getDificultad() {

return dificultad;

}

public boolean isObligatorio() {

return obligatorio;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setAparato(String aparato) {

this.aparato = aparato;

}

public void setPosicionEjecucion(String posicionEjecucion) {

this.posicionEjecucion = posicionEjecucion;

}

public void setDificultad(char dificultad) {

this.dificultad = dificultad;

}

public void setObligatorio(boolean obligatorio) {

this.obligatorio = obligatorio;

}

public int compareTo(MovimientoGym otro) {

return this.nombre.compareTo(otro.nombre);

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((nombre == null) ? 0 : nombre.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

MovimientoGym other = (MovimientoGym) obj;

if (nombre == null) {

if (other.nombre != null) {

return false;

}

} else if (!nombre.equals(other.nombre)) {

return false;

}

return true;

}

public String toString() {

StringBuilder cad=new StringBuilder();

cad.append("\n Movimiento Gimnastico de nombre: "+nombre);

cad.append("\n sobre el aparato: "+aparato);

cad.append("\n en posicion de ejecucion: "+posicionEjecucion);

cad.append("\n de dificultad: "+dificultad);

if(obligatorio) {

cad.append("\n es de tipo obligatorio");

}

else {

cad.append("\n no es de tipo obligatorio");

}

return cad.toString();

}

}

Gimnasta

Clase con varios atributos entre los que se engloban dos arreglos unidimensionales de las clases MovimientoGym y Competencia respectivamente. Esta clase es utilizada para rellenar la matriz de la clase Gimnasio.

[Inicia código de clase Gimnasta]

import java.util.ArrayList;

import java.util.Calendar;

import java.util.GregorianCalendar;;

/\*Guillermo Arredondo

\* 21 de noviembre 2020

\* Clase Gimnasta para Matriz Gimnasio

\*/

public class Gimnasta {

private String nombre;

private String curp;

private int edad;

private char genero;

private int antiguedad;

private String estilo;

private int horario;

private double peso;

private MovimientoGym[] movimientos;

private Competencia[] competencias;

private int numCompetencias;

private int numMovimientos;

private final int MAX\_MOVIMIENTOS=100;

private final int MAX\_COMPETENCIAS=10; //pensemoslo en lapso de 3 años porque sino es un numero muy grande e innecesario

// cada 3 años es bueno hacer un reciclaje del sistema, llamemoslo apoyo a la re-inscripcion

public Gimnasta(String curp) {

this.curp=curp.toUpperCase();

}

public Gimnasta(String nombre, String curp, int antiguedad, String estilo, int horario, double pesoKg) {

int ahoNac, ahoActual, mesActual, mesNac;

String aho, mes;

Calendar aux=new GregorianCalendar();

this.movimientos=new MovimientoGym[MAX\_MOVIMIENTOS];

this.competencias=new Competencia[MAX\_COMPETENCIAS];

this.nombre=nombre;

this.curp=curp.toUpperCase();

this.genero=curp.charAt(10);

aho=curp.substring(4, 6);

ahoNac=Integer.parseInt(aho);

if(ahoNac>20) {

ahoNac+=1900;

}

else {

ahoNac+=2000;

}

mes=curp.substring(6, 8);

mesNac=Integer.parseInt(mes);

//Calendar considera Enero como mes 0

mesActual=aux.get(Calendar.MONTH)+1;

ahoActual=aux.get(Calendar.YEAR);

this.edad=ahoActual-ahoNac;

if(mesNac>mesActual) {

edad--;

}

this.antiguedad=antiguedad;

this.estilo=estilo;

this.horario=horario;

this.peso=pesoKg;

this.numCompetencias=0;

this.numMovimientos=0;

}

public String getNombre() {

return nombre;

}

public String getCurp() {

return curp;

}

public char getGenero() {

return genero;

}

public int getAntiguedad() {

return antiguedad;

}

public String getEstilo() {

return estilo;

}

public int getHorario() {

return horario;

}

public int getEdad() {

return edad;

}

public double getPeso() {

return peso;

}

public int getNumCompetencias() {

return numCompetencias;

}

public int getNumMovimientos() {

return numMovimientos;

}

// este metodo fue una idea para buscar gimnastas activas desde el incio del anio, pero despues murio la idea

// es muy complicado y la verdad no hace algo que alguien quisiera saber en realidad

public String getPrimeraCompetenciaAnio(int anio) {

String res="Ninguna competencia registrada";

int i=0;

while(i<numCompetencias && competencias[i].getAnioFecha()!=anio) {

i++;

}

if(i!=numCompetencias) {

res=competencias[i].toString();

}

return res;

}

public String getUnaCompetencia(String nombre) {

String res="Ninguna competencia registrada";

Competencia aux=new Competencia(nombre);

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(competencias, numCompetencias, aux);

if(pos>=0) {

res=competencias[pos].toString();

}

return res;

}

public String getUnMovimiento(String nombre) {

String res="Sin movimiento realizable";

MovimientoGym aux=new MovimientoGym(nombre);

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(movimientos, numMovimientos, aux);

if(pos>=0) {

res=movimientos[pos].toString();

}

return res;

}

public String getListaMovimAparato(String aparato) {

ArrayList<String> aux=new ArrayList<String>();

for(int i=0; i<numMovimientos; i++) {

if(movimientos[i].getAparato().equalsIgnoreCase(aparato)) {

aux.add(movimientos[i].getNombre());

}

}

if(aux.isEmpty()) {

aux.add("Sin movimientos");

}

return aux.toString();

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setGenero(char genero) {

this.genero = genero;

}

public void setAntiguedad(int antiguedad) {

this.antiguedad = antiguedad;

}

public void setEstilo(String estilo) {

this.estilo = estilo;

}

public void setHorario(int horario) {

this.horario = horario;

}

public void setPeso(double peso) {

this.peso = peso;

}

public String setLogroCompetencia(String nomComp, int logro) {

String res="Competencia no encontrada";

Competencia aux=new Competencia(nomComp);

int pos=ManejadorArreglosGenericos.buscaSecDesorden(competencias, numCompetencias, aux);

if(pos>=0) {

competencias[pos].setLogro(logro);

res=competencias[pos].toString();

}

return res;

}

public boolean altaMovimiento(String nombre, String aparato, String posicionEjecucion, char dificultad, boolean obligatorio) {

boolean res=false;

MovimientoGym mov;

int n;

// existen movimientos con mismo nombre y diferente dificultad o posicion por lo que no se pregunta si ya existe

n=numMovimientos;

dificultad=Character.toUpperCase(dificultad);

mov=new MovimientoGym(nombre, aparato, posicionEjecucion, dificultad, obligatorio);

numMovimientos=ManejadorArreglosGenericos.insertaFinal(movimientos, numMovimientos, mov);

if(numMovimientos!=n) {

res=true;

}

return res;

}

//pensando que es registro antes de llevarse a cabo la competencia, no se pide el logro, y se espera que se establezca posteriormente

public boolean altaCompetencia(String nombre, String lugar, String categoria, int anioFecha) {

boolean res=false;

Competencia aux=new Competencia(nombre);

int n;

int pos=ManejadorArreglosGenericos.buscaSecDesorden(competencias, numCompetencias, aux);

//se considera que no hay competencias con el mismo nombre

if(pos<0) {

aux=new Competencia(nombre, lugar, categoria, anioFecha);

n=numCompetencias;

numCompetencias=ManejadorArreglosGenericos.insertaFinal(competencias, numCompetencias, aux);

if(numCompetencias!=n) {

res=true;

}

}

return res;

}

public String bajaCompetencia(String nombre) {

String res="Competencia no encontrada";

Competencia aux=new Competencia(nombre);

int pos=ManejadorArreglosGenericos.buscaSecDesorden(competencias, numCompetencias, aux);

if(pos>=0) {

res=competencias[pos].toString();

numCompetencias=ManejadorArreglosGenericos.elimina(competencias, numCompetencias, pos);

}

return res;

}

public int numMovimientosAparato(String aparato) {

int contador=0;

for(int i=0; i<numMovimientos; i++) {

if(movimientos[i].getAparato().equalsIgnoreCase(aparato)) {

contador++;

}

}

return contador;

}

/\* los dos metodos siguientes son para uso meramente interno, a traves de los cuales se facilita el proceso del metodo

\* calcula nivel

\*/

public int numMovimientosDificultad(char dificultad) {

int contador=0;

dificultad=Character.toUpperCase(dificultad);

for(int i=0; i<numMovimientos; i++) {

if(movimientos[i].getDificultad()==dificultad) {

contador++;

}

}

return contador;

}

public int numMovimientosObligatorios() {

int contador=0;

for(int i=0; i<numMovimientos; i++) {

if(movimientos[i].isObligatorio()) {

contador++;

}

}

return contador;

}

// el siguiente metodo tambien fue utilizado como criterio en calcula nivel, sin enbargo este sí tiene otros usos en Gimnasio

public char mayDificultad() {

char mayor='!'; //bandera: no hay movimientos registrados, es menor que cualquier dificultad

for(int i=0; i<numMovimientos; i++) {

if(mayor<movimientos[i].getDificultad()) {

mayor=movimientos[i].getDificultad();

}

}

return mayor;

}

/\* metodo para calcular el nivel de cada uno de los gimnastas, es utilizado para colocar de forma adecuada los gimnastas en

\* su correspondiente nivel en la matriz

\*/

public int calculaNivel() {

int nivel=1;

/\*para facilitar la comprension los niveles van a ser representados como en la realidad

\* en clase Gimnasio será necesario restar un uno a este metodo para encontrarlo en el arreglo de niveles

\* el principal punto a evaluar es el 75% de eficiencia en movimiento

\* se considera que para entrar en la alta de Movimiento ya se considera suficiente eficiencia en realizacion

\*

\* en gimnasia los niveles del 1-4 y a veces el 5 son basicamente para recreacion y motivacion

\* por lo que es el inicio competitivo y el criterio de movimientos de acuerdo a dificultad los que aportan el correspondiente nivel

\*/

if(numMovimientos==0 && numCompetencias<2) {

if(numMovimientosObligatorios()<5) {

nivel=1;

}

else if(numMovimientosObligatorios()<10) {

nivel=2;

}

else if(numMovimientosObligatorios()<15 || mayDificultad()=='A') {

nivel=3;

}

else if(numMovimientosObligatorios()<20 || numMovimientosDificultad('A')>5) {

nivel=4;

}

else if(numMovimientosObligatorios()<20 && mayDificultad()=='B') {

nivel=5;

}

}

else if(numCompetencias>=2 && mayDificultad()>'B'){

if(numMovimientosDificultad('B')<10 && mayDificultad()=='C') {

nivel=6;

}

else if(numMovimientosDificultad('C')<10 && numMovimientosDificultad('D')<5) {

nivel=7;

}

else if(numMovimientosDificultad('D')<10) {

nivel=8;

}

else if((numMovimientosDificultad('D')<10 && numMovimientosDificultad('E')<5) || mayDificultad()=='F') {

nivel=9;

}

else if(numMovimientosDificultad('F')<10 || mayDificultad()=='G') {

nivel=10;

}

else if(numMovimientosDificultad('F')<20 || mayDificultad()>='H') {

nivel=11;

}

}

return nivel;

}

public double calculaProteinas(boolean masMusculo) {

/\* despues de cada entrenamiento es recomendable tomar 20 gramos de proteina

\* se considera un entrenamiento por dia

\* de acuerdo a expertos en nutriologia se considera que un atleta debe, en forma general (hay que recordar que cada

\* cuerpo es diferente) debe consumor las proporciones entre 1.2 para mujeres y 1.5 para hombres

\*/

double proteinas=20;

double proporcion;

if(masMusculo) {

proporcion=1.5;

}

else {

proporcion=1.2;

}

switch(this.genero) {

case 'M':proteinas+=(peso\*proporcion);

break;

case 'H':

proporcion+=2.5;

proteinas+=(peso\*proporcion);

break;

default:

}

return proteinas;

}

public boolean participoCompetencia(String nombre) {

boolean res=false;

Competencia aux=new Competencia(nombre);

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(competencias, numCompetencias, aux);

//pensando que puede no darse de baja una Competencia, por descuido, si no hay logro es porque no asistio

if(pos>=0 && competencias[pos].getLogro()!=0) {

res=true;

}

return res;

}

public boolean conMovimiento(String nombre) {

boolean res=false;

MovimientoGym aux=new MovimientoGym(nombre);

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(movimientos, numMovimientos, aux);

if(pos>=0) {

res=true;

}

return res;

}

public String movimientoMasDificil() {

String res="Ningun movimiento registrado";

char mayor=' '; // en tablas ASCII el espacio es mayor que cualquier letra del alfabeto

for(int i=0; i<numMovimientos; i++) {

if(mayor<movimientos[i].getDificultad()) { // la menor dificultad es la letra A

mayor=movimientos[i].getDificultad();

res=movimientos[i].toString();

}

}

return res;

}

public String mejorAparato() {

String res="Ningun movimiento aprendido";

char mayDif=' ';

for(int i=0; i<numMovimientos; i++) {

if(movimientos[i].getDificultad()>mayDif) {

mayDif=movimientos[i].getDificultad();

res=movimientos[i].getAparato();

}

}

return res;

}

public String mejorCompetencia() {

String res="Ninguna competencia registrada";

int mayorLogro=-1;

for(int i=0; i<numCompetencias; i++) {

if(competencias[i].getLogro()>mayorLogro) {

mayorLogro=competencias[i].getLogro();

res=competencias[i].getNombre();

}

}

return res;

}

public int compareTo(Gimnasta otro) {

return this.curp.compareTo(otro.curp);

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((curp == null) ? 0 : curp.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Gimnasta other = (Gimnasta) obj;

if (curp == null) {

if (other.curp != null) {

return false;

}

} else if (!curp.equals(other.curp)) {

return false;

}

return true;

}

public String toString() {

StringBuilder cad=new StringBuilder();

cad.append("\n Gimnasta de nombre: "+nombre);

cad.append("\n de curp: "+curp);

cad.append("\n de edad: "+edad);

cad.append("\n de genero: "+genero);

cad.append("\n con antiguedad de: "+antiguedad+" año(s)");

cad.append("\n compite en estilo: "+estilo);

cad.append("\n entrena en horario: "+(horario/100)+":");

if(horario%100==0) {

cad.append("00");

}

else {

cad.append(horario%100);

}

cad.append("\n peso: "+peso+" Kg");

cad.append("\n numero de competencias: "+numCompetencias);

for(int i=0; i<numCompetencias; i++) {

cad.append("\n participo en: "+competencias[i].toString());

}

cad.append("\n capaz de realizar: "+numMovimientos+" movimiento(s)");

for(int j=0; j<numMovimientos; j++) {

cad.append("\n "+movimientos[j].toString());

}

return cad.toString();

}

}

Gimnasio

Clase final y la que engloba a todas las clases anteriores, asimismo es la clase incorporada a la interfaz gráfica. Dentro de esta se llama a la mayoría de los métodos encontrados en las clases anteriores. El atributo principal de esta clase es la matriz de clase Gimnasta.

[Inicia código de clase Gimnasio]

import java.util.ArrayList;

/\*Guillermo Arredondo

\* 21 de noviembre 2020

\* Clase Gimnasio

\*/

public class Gimnasio {

private String nombre;

private String ubicacion;

private String encargado;

private Gimnasta[][] competidores;

private final String[] NIVELES= {"EXCEL 1", "EXCEL 2","EXCEL 3","EXCEL 4","EXCEL 5","EXCEL 6","EXCEL 7","EXCEL 8","EXCEL 9","EXCEL 10","FIG"};

private Integer[] numCompsNivel;

private final int MAX\_COMPETIDORES=20;

private final int MAX\_NIVELES=11;

public Gimnasio() {

this.competidores=new Gimnasta[MAX\_NIVELES][MAX\_COMPETIDORES];

this.numCompsNivel=new Integer[MAX\_NIVELES];

for(int i=0; i<MAX\_NIVELES; i++) {

numCompsNivel[i]=0;

}

}

public Gimnasio(String nombre, String ubicacion, String encargado) {

this();

this.nombre=nombre;

this.ubicacion=ubicacion;

this.encargado=encargado;

}

public boolean isGimnasioVacio() {

boolean res=true;

int j=0;

while(j<MAX\_NIVELES && numCompsNivel[j]==0) {

j++;

}

if(j!=MAX\_NIVELES) {

res=false;

}

return res;

}

public String getNombre() {

return nombre;

}

public String getUbicacion() {

return ubicacion;

}

public String getEncargado() {

return encargado;

}

public int getMAX\_NIVELES() {

return MAX\_NIVELES;

}

public int getNumCompsNivel(int nivel) {

int res=-1;

nivel--;

if(nivel>=0 && nivel<MAX\_NIVELES) {

res=numCompsNivel[nivel];

}

return res;

}

public int getNumCompsNivel(String nivel) {

int res=-1;

nivel=nivel.toUpperCase();

int pos=ManejadorArreglosGenericos.buscaSecDesorden(NIVELES, MAX\_NIVELES, nivel);

if(pos>=0) {

res=numCompsNivel[pos];

}

return res;

}

public String getUnGimnasta(String curp) {

String res="Persona no encontrada";

curp.toUpperCase();

Gimnasta aux=new Gimnasta(curp);

int j, i;

boolean encontre=false;

j=0;

while(!encontre && j<MAX\_NIVELES) {

i=0;

while(i<numCompsNivel[j] && !competidores[j][i].equals(aux)) {

i++;

}

if(i!=numCompsNivel[j]) {

res=competidores[j][i].toString();

encontre=true;

}

j++;

}

return res;

}

public String getUnGimnasta(int nivel, int pos) {

String res="";

if(nivel>=0 && pos>=0 && nivel<MAX\_NIVELES && pos<numCompsNivel[nivel]) {

res=competidores[nivel][pos].getNombre();

}

return res;

}

public String getUnMovimientoGimnasta(String curpGimnasta, String nombreMov) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].getUnMovimiento(nombreMov);

encontre=true;

}

j++;

}

return res;

}

public String getUnaCompetenciaGimnasta(String curpGimnasta, String nombreCompetencia) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].getUnaCompetencia(nombreCompetencia);

encontre=true;

}

j++;

}

return res;

}

public double getProteinasGimnasta(String curpGimnasta, boolean masMusculo) {

double res=-1; //bandera de error, no encontro gimnasta

curpGimnasta=curpGimnasta.toUpperCase();

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].calculaProteinas(masMusculo);

encontre=true;

}

j++;

}

return res;

}

public String getMovMasDificilGimnasta(String curpGimnasta) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].movimientoMasDificil();

encontre=true;

}

j++;

}

return res;

}

public String getGimnastasNivel(String nivel) {

nivel=nivel.toUpperCase();

String res="No hay gimnastas registradas en "+nivel;

StringBuilder cad=new StringBuilder();

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(NIVELES, MAX\_NIVELES, nivel);

if(pos>=0) {

for(int i=0; i<numCompsNivel[pos]; i++) {

cad.append("\t"+competidores[pos][i].getNombre());

}

res=cad.toString();

}

return res;

}

public String getGimnastasNivel(int nivel) {

String res="No hay gimnastas registradas en "+nivel;

StringBuilder cad=new StringBuilder();

if(nivel>=0 && nivel<MAX\_NIVELES) {

for(int i=0; i<numCompsNivel[nivel]; i++) {

cad.append(" "+competidores[nivel][i].getNombre());

}

res=cad.toString();

}

return res;

}

public String getListaMovimientosGimnastaAparato(String curpGimnasta, String aparato) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].getListaMovimAparato(aparato);

encontre=true;

}

j++;

}

return res;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setUbicacion(String ubicacion) {

this.ubicacion = ubicacion;

}

public void setEncargado(String encargado) {

this.encargado = encargado;

}

public String setLogroCompetenciaGimnasta(String curpGimnasta, String nomComp, int logro) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].setLogroCompetencia(nomComp, logro);

encontre=true;

}

j++;

}

return res;

}

public boolean altaGimnasta(String nombre, String curp, int antiguedad, String estilo, int horario, double pesoKg) {

boolean res=false;

Gimnasta g;

int n;

//dar de alta por default en el nivel 1

g=new Gimnasta(nombre, curp, antiguedad, estilo, horario, pesoKg);

n=numCompsNivel[0];

numCompsNivel[0]=ManejadorArreglosGenericos.insertaFinal(competidores[0], numCompsNivel[0], g);

if(numCompsNivel[0]!=n) {

res=true;

}

return res;

}

public boolean altaMovimientoGimnasta(String curpGimnasta, String nombre, String aparato, String posicionEjecucion, char dificultad, boolean obligatorio) {

boolean res=false;

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].altaMovimiento(nombre, aparato, posicionEjecucion, dificultad, obligatorio);

encontre=true;

}

j++;

}

return res;

}

public boolean altaCompetenciaGimnasta(String curpGimnasta, String nombre, String lugar, String categoria, int anioFecha) {

boolean res=false;

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].altaCompetencia(nombre, lugar, categoria, anioFecha);

encontre=true;

}

j++;

}

return res;

}

public String bajaCompetenciaGimnasta(String curpGimnasta, String nombre) {

String res="Gimnasta no encontrado";

Gimnasta g=new Gimnasta(curpGimnasta);

int j=0;

int pos;

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].bajaCompetencia(nombre);

encontre=true;

}

j++;

}

return res;

}

public String bajaGimnasta(String curp) {

String res="Gimnasta no encontrado";

int j=0;

int pos;

Gimnasta g=new Gimnasta(curp);

boolean encontre=false;

while(!encontre && j<MAX\_NIVELES) {

pos=ManejadorArreglosGenericos.buscaSecDesorden(competidores[j], numCompsNivel[j], g);

if(pos>=0) {

res=competidores[j][pos].toString();

numCompsNivel[j]=ManejadorArreglosGenericos.elimina(competidores[j], numCompsNivel[j], pos);

encontre=true;

}

j++;

}

return res;

}

public void actualizaGimnasio() {

int nivel;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

nivel=competidores[j][i].calculaNivel()-1;

if(nivel!=j) {

numCompsNivel[nivel]=ManejadorArreglosGenericos.insertaFinal(competidores[nivel], numCompsNivel[nivel], competidores[j][i]);

numCompsNivel[j]=ManejadorArreglosGenericos.elimina(competidores[j], numCompsNivel[j], i);

}

}

}

}

public int cuantosGimnastasConMovimiento(String nombre) {

int contador=0;

boolean aux;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

aux=competidores[j][i].conMovimiento(nombre);

if(aux) {

contador++;

}

}

}

return contador;

}

public int gimnastasHorario(int horario) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].getHorario()==horario) {

contador++;

}

}

}

return contador;

}

public int gimnastasEdad(int edad) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].getEdad()==edad) {

contador++;

}

}

}

return contador;

}

public int cuantosGimnastasEnCompetencia(String nombreComp) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].participoCompetencia(nombreComp)) {

contador++;

}

}

}

return contador;

}

public String gimnastaMayDificul() {

String res="Ninguna gimnasta registrada";

char mayor=' ';

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].mayDificultad()>mayor) {

mayor=competidores[j][i].mayDificultad();

res=competidores[j][i].toString();

}

}

}

return res;

}

public int cuantasExpertiseAparato(String aparato) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].mejorAparato().equalsIgnoreCase(aparato)) {

contador++;

}

}

}

return contador;

}

public int cuantasEstilo(String estilo) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].getEstilo().equalsIgnoreCase(estilo)) {

contador++;

}

}

}

return contador;

}

public int cuantasAntiguedad(int antiguedad) {

int contador=0;

for(int j=0; j<MAX\_NIVELES; j++) {

for(int i=0; i<numCompsNivel[j]; i++) {

if(competidores[j][i].getAntiguedad()==antiguedad) {

contador++;

}

}

}

return contador;

}

public String mejoresCompetenciasGimnastasNivel(String nivel) {

ArrayList<String> aux=null;

String res="No hay gimnastas registradas en nivel";

nivel=nivel.toUpperCase();

int pos;

pos=ManejadorArreglosGenericos.buscaSecDesorden(NIVELES, MAX\_NIVELES, nivel);

if(pos>=0) {

aux=new ArrayList<String>();

for(int i=0; i<numCompsNivel[pos]; i++) {

aux.add(competidores[pos][i].mejorCompetencia());

}

res=aux.toString();

}

return res;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((ubicacion == null) ? 0 : ubicacion.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj) {

return true;

}

if (obj == null) {

return false;

}

if (getClass() != obj.getClass()) {

return false;

}

Gimnasio other = (Gimnasio) obj;

if (ubicacion == null) {

if (other.ubicacion != null) {

return false;

}

} else if (!ubicacion.equals(other.ubicacion)) {

return false;

}

return true;

}

public int compareTo(Gimnasio otro) {

return this.ubicacion.compareTo(otro.ubicacion);

}

public String toString() {

StringBuilder cad=new StringBuilder();

cad.append("\n Gimnasio de nombre: "+nombre);

cad.append("\n ubicado en: "+ubicacion);

cad.append("\n bajo direccion de: "+encargado);

for(int j=0; j<MAX\_NIVELES; j++) {

if(numCompsNivel[j]!=0) {

cad.append("\n en el nivel "+NIVELES[j]);

cad.append("\t hay los siguientes "+numCompsNivel[j].toString()+" gimnastas:");

}

for(int i=0; i<numCompsNivel[j]; i++) {

cad.append("\n "+competidores[j][i].toString());

}

cad.append("\n ");

}

return cad.toString();

}

}

Main de pruebas:

Una vez que las clases MovimientoGym y Competencia, son clases sumamente básicas que no cuentan con ningún método extraordinario, no se llevaron a cabo pruebas únicas para estos métodos. Sin embargo, se considera que son englobadas dentro de las pruebas en clase Gimnasta (PruebaGimnasta) y en Gimnasio (PruebaGimnasioArchivo).

PruebaGimnasta

Se lleva a cabo una instanciación de un Gimnasta a partir de constantes y se prueban tanto altas de movimientos y competencias como bajas de competencias, búsquedas y métodos estadísticos con errores en escritura, y eliminaciones intercaladas. Todos los métodos probaron resultar efectivos a lo largo de la prueba.

[Inicia código de PruebaGimnasta]

import java.util.Scanner;

/\*Guillermo Arredondo

\* 30 de noviembre 2020

\* Clase de prueba de gimnasta

\*/

public class PruebaGimnasta {

public static void main(String[] args) {

Gimnasta g=new Gimnasta("Amaya", "PEBA061102MDFxxx", 5, "olimpica", 1630, 47);

System.out.println(g.toString());

if(g.altaCompetencia("USA Open", "Texas", "Junior Elite", 2018)) {

System.out.println("Alta competencia exitosa");

}

else {

System.out.println("Error en alta competencia ");

}

System.out.println(g.setLogroCompetencia("USA Open", 3));

if(g.altaCompetencia("Mex", "Mahalo", "Junior", 2016)) {

System.out.println("Alta competencia exitosa");

}

else {

System.out.println("Error en alta competencia ");

}

System.out.println(g.setLogroCompetencia("Mex", 1));

if(g.altaCompetencia("Patricia", "Noa", "Junior", 2016)) {

System.out.println("Alta competencia exitosa");

}

else {

System.out.println("Error en alta competencia ");

}

//quito una letra para reconocer que no encuentre la competencia y marque error en alta

System.out.println(g.setLogroCompetencia("Patrici", 3));

if(g.altaMovimiento("Salida Backflip", "viga", "agrupado", 'C', true)) {

System.out.println("Alta movimiento exitosa");

}

else {

System.out.println("Error en alta movimiento");

}

if(g.altaMovimiento("Arabe", "viga", "extendido", 'C', true)) {

System.out.println("Alta movimiento exitosa");

}

else {

System.out.println("Error en alta movimiento");

}

if(g.altaMovimiento("Alemana", "barras", "carpado", 'A', true)) {

System.out.println("Alta movimiento exitosa");

}

else {

System.out.println("Error en alta movimiento");

}

System.out.println(g.toString());

System.out.println(g.bajaCompetencia("USA Open"));

System.out.println(g.bajaCompetencia("USA Open"));

System.out.println(g.conMovimiento("Salida Backflip"));

System.out.println(g.calculaProteinas(false));

System.out.println(g.getListaMovimAparato("viga"));

System.out.println(g.getListaMovimAparato("barras"));

System.out.println(g.participoCompetencia("USA Open"));

System.out.println(g.calculaNivel());

System.out.println(g.numMovimientosDificultad('B'));

System.out.println(g.getNumMovimientos());

System.out.println(g.numMovimientosAparato("viga"));

System.out.println(g.mejorCompetencia());

System.out.println(g.mejorAparato());

System.out.println(g.mayDificultad());

System.out.println(g.getUnaCompetencia("Patricia"));

System.out.println(g.getUnMovimiento("Alemana"));

System.out.println(g.movimientoMasDificil());

}

}

Por practicidad y tiempo, no se realizó una clase de pruebas con constantes en la clase Gimnasio, sino que se optó por hacerlo dentro de la clase PruebaGimnasioArchivo, de este modo, se probarían tanto lectura de archivos como ingreso de datos en la matriz y en arreglos. Por si fuera necesario, se agregaron algunas pruebas con constantes dentro de la misma clase.

PruebaGimnasioArchivo

Clase que acude a un archivo de texto para hacer pruebas de validación tanto de lecturas como de métodos de la clase Gimnasio, para facilitar el control y la evaluación de las lecturas de textos de más de una palabra se utilizaron diversas impresiones a lo largo del código. Se hicieron diversas pruebas de carga de datos, en el siguiente código se muestra únicamente la versión final. Al final de las lecturas de archivo se incorporan métodos que a su vez prueban errores de escritura, y datos no registrados para verificar la correcta validación de búsquedas y recorridos de arreglos y matriz.

[Inicia código de PruebaGimnasioArchivo]

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

/\*Guillermo Arredondo

\* 30 noviembre 2020

\* Prueba con Archivo de la Clase Gimnasio

\*/

public class PruebaGimnasioArchivo {

public static void main(String[] args) {

File entrada=new File("gimnasio.txt");

Gimnasio g;

String nombre, ubicacion, encargado, curp, estilo, aparato, posicionEjecucion, categoria;

int antiguedad, horario, logro, anio;

double peso;

char dificultad;

boolean obligatorio;

try(Scanner lec=new Scanner(entrada)){

nombre=lec.next();

lec.nextLine();

ubicacion=lec.nextLine();

encargado=lec.next();

g=new Gimnasio(nombre, ubicacion, encargado);

System.out.println("Gimnasio esta vacio? "+g.isGimnasioVacio());

while(lec.hasNext()) {

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

curp=lec.next();

System.out.println(curp);

antiguedad=lec.nextInt();

System.out.println(antiguedad);

estilo=lec.next();

System.out.println(estilo);

horario=lec.nextInt();

System.out.println(horario);

peso=lec.nextDouble();

System.out.println(peso);

if(g.altaGimnasta(nombre, curp, antiguedad, estilo, horario, peso)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

aparato=lec.next();

System.out.println(aparato);

posicionEjecucion=lec.next();

System.out.println(posicionEjecucion);

dificultad=lec.next().charAt(0);

System.out.println(dificultad);

obligatorio=lec.nextBoolean();

System.out.println(obligatorio);

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

aparato=lec.next();

posicionEjecucion=lec.next();

dificultad=lec.next().charAt(0);

obligatorio=lec.nextBoolean();

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

aparato=lec.next();

posicionEjecucion=lec.next();

dificultad=lec.next().charAt(0);

obligatorio=lec.nextBoolean();

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

}

System.out.print(g.toString());

g.actualizaGimnasio();

System.out.print(g.bajaGimnasta("CANA010126HDFxxx"));

System.out.print(g.toString());

if(g.altaGimnasta("Abraham Chalita","CANA010126HDFxxx" , 3, "Olímpica", 1430, 69.6)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("E R R O R en alta con constantes");

}

if(g.altaMovimientoGimnasta("CANA010126HDFxxx", "Arabe", "suelo", "extendida", 'A', true)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

if(g.altaCompetenciaGimnasta("CANA010126HDFxxx", "TEC GYM", "TEC Monterrey", "Patito", 2018)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta("CANA010126HDFxxx", "TEC Gym", 10));

System.out.println("Gimnasio esta vacio? "+g.isGimnasioVacio());

System.out.println("Gimnastas en nivel EXCEL1 "+g.getGimnastasNivel("EXCEL 1"));

System.out.println(g.getProteinasGimnasta("CAFM020621MDFXXX", true));

System.out.println(g.getListaMovimientosGimnastaAparato("CAFM020621MDFXXX", "suelo"));

System.out.println(g.getGimnastasNivel("EXCEL 6"));

System.out.println(g.cuantasAntiguedad(7));

System.out.println(g.cuantasAntiguedad(3));

System.out.println(g.cuantosGimnastasConMovimiento("Mortal atras"));

System.out.println(g.gimnastaMayDificul());

System.out.println(g.cuantosGimnastasEnCompetencia("Nacional Juvenil"));

System.out.println(g.cuantosGimnastasEnCompetencia("Sirius"));

System.out.println(g.mejoresCompetenciasGimnastasNivel("EXCEL 6"));

System.out.println(g.mejoresCompetenciasGimnastasNivel("Excel 6"));

}

catch(FileNotFoundException fnfe) {

System.err.println("E R R O R "+fnfe);

System.exit(-1);

}

}

}

Considero importante aclarar los aspectos respecto al previo código: primero, al copiarlo de JAVA a Word se producen algunos cambios en sangrías que son importantes considerar. Por lo que puedo observar esto se debe al espacio de texto en Word. Asimismo, este código resulto no probar del todo los métodos, puesto que esta clase nunca marcó un error en el método *getListaMovimientosGimnastaAparato*; sin embargo, una vez cargado a la interfaz gráfica fue visible un error en compilación que daba como resultado un ciclo infinito. El método fue solucionado, pero se desconoce la razón por la que el archivo no marcara el error correspondiente (siendo la misma información y el mismo método).

Clases interfaces:

GimnasioVista

Clase para diseñar la vista de la interfaz gráfica, como es común entre las clases de tipo vista hereda de la interfaz JFrame. Se incluyen comentarios a lo largo del código para simplificar el seguimiento de cada conjunto de líneas de programación.

[Inicia código de GimnasioVista]

import java.awt.Color;

import java.awt.GridLayout;

import javax.swing.\*;

import javax.swing.border.Border;

/\*Guillermo Arredondo

\* 30 nov 2020

\* Clase Vista de interfaz Gimnasio

\*/

public class GimnasioVista extends JFrame{

protected JTextArea pantallaGral;

protected JButton leerArchivo, tablaDataBt, nutricionBt, altaLogrosBt, altaGimnastasBt, actualizaGymBt, infoComplBt;

protected JButton buscaGimnasBt, bajaCompBt, bajaGimnastaBt, mayDificultad, expAparato, mejoresCompetenciasNivel, gimNivel;

protected JButton cuantosMovimiento, cuantosCompetencia, horario, edad, estilo, antiguedad, logroCompBt, numComp, buscaLogro;

protected JLabel inicioLb, estadCuantos;

protected JTextField nutricionTxtF, estadTxtF, inscripcionesTxtF1,inscripcionesTxtF2,inscripcionesTxtF3, inscripcionesTxtF4, logrosTxtF1, pantallaGimnastas;

protected JTextField logrosTxtF2, logrosTxtF3, logrosTxtF4, logrosTxtF5;

protected JRadioButton nutMuscRdBt, obligatorioBt, movBt, compBt, listMovBt, movMasDificil;

protected JComboBox<String> estilosCBox1, estilosCBox2, nivelesCBox1, nivelesCBox2, horarioCBox1, horarioCBox2;

protected JScrollPane textoGrande;

public GimnasioVista(String titulo) {

super(titulo);

String[] listaEstilos={"Rítmica", "Olimpica", "Tricking", "Parkour", "Recreativa", "Adultos"};

String[] listaNiveles={"EXCEL 1", "EXCEL 2","EXCEL 3","EXCEL 4","EXCEL 5","EXCEL 6","EXCEL 7","EXCEL 8","EXCEL 9","EXCEL 10","FIG"};

String[] listaHorarios= {"08:00", "10:00", "12:00", "14:00", "14:30", "15:00", "16:00", "16:30", "19:00", "19:30", "20:30"};

JPanel panel=new JPanel();

panel.setLayout(new GridLayout(5,6));

Border gap=BorderFactory.createEmptyBorder(5, 5, 5, 5);

panel.setBorder(gap);

JPanel nutricionPn=new JPanel();

nutricionPn.setLayout(new GridLayout(2,2));

Border gap2=BorderFactory.createTitledBorder(gap, "NUTRICION");

Border gap3=BorderFactory.createBevelBorder(WIDTH, Color.BLUE, Color.BLUE);

Border gap4=BorderFactory.createCompoundBorder(gap3, gap2);

nutricionPn.setBorder(gap4);

JPanel inscripcionesPn=new JPanel();

inscripcionesPn.setLayout(new GridLayout(2,7));

gap2=BorderFactory.createTitledBorder(gap, "INSCRIPCIONES");

gap3=BorderFactory.createBevelBorder(WIDTH, Color.CYAN, Color.CYAN);

gap4=BorderFactory.createCompoundBorder(gap3, gap2);

inscripcionesPn.setBorder(gap4);

JPanel gimnastasPn=new JPanel();

gimnastasPn.setLayout(new GridLayout(2,1));

gap2=BorderFactory.createTitledBorder(gap, "GIMNASTAS");

gap3=BorderFactory.createBevelBorder(WIDTH, Color.GREEN, Color.GREEN);

gap4=BorderFactory.createCompoundBorder(gap3, gap2);

gimnastasPn.setBorder(gap4);

JPanel logrosPn=new JPanel();

logrosPn.setLayout(new GridLayout(4,5));

gap2=BorderFactory.createTitledBorder(gap, "LOGROS- COMPETENTCIAS Y MOVIMIENTOS");

gap3=BorderFactory.createBevelBorder(WIDTH, Color.MAGENTA, Color.MAGENTA);

gap4=BorderFactory.createCompoundBorder(gap3, gap2);

logrosPn.setBorder(gap4);

JPanel estadPn=new JPanel();

estadPn.setLayout(new GridLayout(4,5));

gap2=BorderFactory.createTitledBorder(gap, "ESTADISTICAS GYM");

gap3=BorderFactory.createBevelBorder(WIDTH, Color.WHITE, Color.WHITE);

gap4=BorderFactory.createCompoundBorder(gap3, gap2);

estadPn.setBorder(gap4);

JPanel tablasDatos=new JPanel();

tablasDatos.setLayout(new GridLayout(1,2));

gap2=BorderFactory.createTitledBorder(gap, "Impresión de Datos");

gap3=BorderFactory.createBevelBorder(WIDTH, Color.WHITE, Color.WHITE);

gap4=BorderFactory.createCompoundBorder(gap3, gap2);

tablasDatos.setBorder(gap4);

this.inicioLb=new JLabel("Bienvenido a Gym");

this.leerArchivo=new JButton("Archivo");

this.nutricionTxtF=new JTextField(25);

this.nutricionBt=new JButton("Nutricion");

this.nutMuscRdBt=new JRadioButton("Desea más músculo");

this.estilosCBox1=new JComboBox<String>(listaEstilos);

this.estilosCBox2=new JComboBox<String>(listaEstilos);

this.altaGimnastasBt=new JButton("Nuevos Gimnastas");

this.gimNivel=new JButton("Gimnastas en nivel");

this.inscripcionesTxtF1=new JTextField("Nombre",30);

this.inscripcionesTxtF2=new JTextField("CURP",30);

this.inscripcionesTxtF3=new JTextField("Años en gimnasio",30);

this.inscripcionesTxtF4=new JTextField("Peso en Kg",30);

this.buscaGimnasBt=new JButton("Busca");

this.bajaGimnastaBt=new JButton("BAJA GIMNASTA");

this.pantallaGimnastas=new JTextField(70);

this.nivelesCBox1=new JComboBox<String>(listaNiveles);

this.nivelesCBox2=new JComboBox<String>(listaNiveles);

this.mayDificultad=new JButton("Mayor Dif.");

this.expAparato=new JButton("Exp. Aparato");

this.mejoresCompetenciasNivel=new JButton("Mejores Competencias en Nivel");

this.horarioCBox1=new JComboBox<String>(listaHorarios);

this.horarioCBox2=new JComboBox<String>(listaHorarios);

this.logrosTxtF1=new JTextField(30);

this.logrosTxtF2=new JTextField(30);

this.logrosTxtF3=new JTextField(30);

this.logrosTxtF4=new JTextField(30);

this.logrosTxtF5=new JTextField(30);

this.obligatorioBt=new JRadioButton("Obligatorio");

this.movBt=new JRadioButton("Es Movimiento");

this.compBt=new JRadioButton("Es Competencia");

this.buscaLogro=new JButton("Busca");

this.infoComplBt=new JButton("INFO GYM");

this.tablaDataBt=new JButton("Tabla de datos");

this.actualizaGymBt=new JButton("ActualizaGym");

this.altaLogrosBt=new JButton("Altas");

this.bajaCompBt=new JButton("Baja competencia");

this.listMovBt=new JRadioButton("Lista de Movimientos de Gimnasta");

this.logroCompBt=new JButton("Logro en Competencia");

this.movMasDificil=new JRadioButton("Movimiento más dificil de Gimnasta");

this.estadTxtF=new JTextField(40);

this.estadCuantos=new JLabel("Cuantos:");

this.cuantosMovimiento=new JButton("Movimiento");

this.cuantosCompetencia=new JButton("Competencia");

this.horario=new JButton("Horario");

this.edad=new JButton("Edad");

this.estilo= new JButton("Estilo");

this.numComp=new JButton("Competidores en Nivel");

this.antiguedad=new JButton("Años en gym");

this.pantallaGral=new JTextArea("Información sobre GYM",30, 50);

this.textoGrande=new JScrollPane(pantallaGral);

panel.add(inicioLb);

panel.add(leerArchivo,1);

// primer panel: destinado a calcular nutricion del gimnasta solicitado

nutricionPn.add(nutricionTxtF);

nutricionPn.add(nutricionBt);

nutricionPn.add(nutMuscRdBt);

panel.add(nutricionPn, 2);

/\* segundo panel: destinado a altas gimnastas unicamente, para facilitar interaccion hay un

\* text field por cada parametro necesario y Combo Box para otros de eleccion

\*/

inscripcionesPn.add(inscripcionesTxtF1);

inscripcionesPn.add(inscripcionesTxtF2);

inscripcionesPn.add(inscripcionesTxtF3);

inscripcionesPn.add(inscripcionesTxtF4);

inscripcionesPn.add(altaGimnastasBt);

inscripcionesPn.add(estilosCBox1);

inscripcionesPn.add(horarioCBox1);

panel.add(inscripcionesPn, 3);

/\* tercer panel: destinado a explotar la informacion de la matriz de gimnastas, aporta algunas

\* estadisticas y permite un campo a traves del cual es necesario siempre colocar el CURP del

\* gimnasta con el que se desea trabajar

\*/

gimnastasPn.add(pantallaGimnastas);

gimnastasPn.add(buscaGimnasBt);

gimnastasPn.add(bajaGimnastaBt);

gimnastasPn.add(mejoresCompetenciasNivel);

gimnastasPn.add(mayDificultad);

gimnastasPn.add(expAparato);

gimnastasPn.add(nivelesCBox1);

gimnastasPn.add(gimNivel);

panel.add(gimnastasPn, 4);

/\*se establece un grupo de botones para evitar doble eleccion de movimientos y competencias

\* sumamente importante para las altas y busquedas

\*/

ButtonGroup group = new ButtonGroup();

group.add(movBt);

group.add(compBt);

/\*cuarto panel: logros, en este se explota el uso de los arreglos instanciados en cada gimnasta

\* por un lado tenemos los movimientos y por el otro las competencias, importante distinguir

\*/

logrosPn.add(logrosTxtF1);

logrosPn.add(logrosTxtF2);

logrosPn.add(logrosTxtF3);

logrosPn.add(logrosTxtF4);

logrosPn.add(movBt);

logrosPn.add(compBt);

logrosPn.add(buscaLogro);

logrosPn.add(altaLogrosBt);

logrosPn.add(obligatorioBt);

logrosPn.add(listMovBt);

logrosPn.add(logroCompBt);

logrosPn.add(bajaCompBt);

logrosPn.add(movMasDificil);

logrosPn.add(logrosTxtF5);

panel.add(logrosPn,5);

/\*un boton grande que actualiza el gimnasio, un metodo muy importante y complejo para poder

\* colocar a cada gimnasta en su nivel correspondiente

\*/

panel.add(actualizaGymBt, 6);

/\*boton grande para poder desplegar dos diferentes toString, el primero despliega un metodo de estilo

\* imprimeMatriz en el cual se presentan por niveles los nombres de todos los gimnastas en

\* el gimnasio, el segundo es el toString de todo el gimnasio en general

\*/

tablasDatos.add(tablaDataBt);

tablasDatos.add(infoComplBt);

panel.add(tablasDatos,7);

/\*

\* quinto panel: estadisticas: este panel sirve para obtener todas las estadisticas

\* que puedan ser requeridad por el Gimnasio acerca del tipo de competidores que hay

\* inscritos

\*/

estadTxtF.setBounds(0, 0, 10, 60);

estadPn.add(estadTxtF);

estadPn.add(estadCuantos);

estadPn.add(horario);

estadPn.add(cuantosMovimiento);

estadPn.add(cuantosCompetencia);

estadPn.add(horarioCBox2);

estadPn.add(estilo);

estadPn.add(edad);

estadPn.add(numComp);

estadPn.add(estilosCBox2);

estadPn.add(antiguedad);

estadPn.add(nivelesCBox2);

panel.add(estadPn,8);

/\*por ultimo la pantalla en la cual se despliega la informacion grande de varios metodos

\* cuenta con ScrollPane para poder visualizar mas facilmente

\*/

panel.add(textoGrande, 9);

this.add(panel);

this.setVisible(true);

this.setBounds(50, 0, 1800, 900);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

GimnasioVista yo=new GimnasioVista("PRUEBA:...");

}

}

GimnasioControlador

Es una clase que hereda de la clase GimnasioVista y es utilizada para incorporar las diversas acciones que llevará a cabo el programa a través de interacción con el usuario.

[Inicia código GimnasioControlador]

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

/\*Guillermo Arredondo

\* 30 noviembre 2020

\* Clase Controlador, interfaz final

\*/

public class GimnasioControlador extends GimnasioVista{

private Gimnasio g;

public void leeArchivo() {

File entrada=new File("gimnasio.txt");

Gimnasio g;

String nombre, ubicacion, encargado, curp, estilo, aparato, posicionEjecucion, categoria;

int antiguedad, horario, logro, anio;

double peso;

char dificultad;

boolean obligatorio;

try(Scanner lec=new Scanner(entrada)){

nombre=lec.next();

lec.nextLine();

ubicacion=lec.nextLine();

encargado=lec.next();

g=new Gimnasio(nombre, ubicacion, encargado);

// Para asegurar una buena lectura, y correcto funcionamiento, se dejaron varias impresiones para checar las lecturas

while(lec.hasNext()) {

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

curp=lec.next();

System.out.println(curp);

antiguedad=lec.nextInt();

System.out.println(antiguedad);

estilo=lec.next();

System.out.println(estilo);

horario=lec.nextInt();

System.out.println(horario);

peso=lec.nextDouble();

System.out.println(peso);

if(g.altaGimnasta(nombre, curp, antiguedad, estilo, horario, peso)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

aparato=lec.next();

System.out.println(aparato);

posicionEjecucion=lec.next();

System.out.println(posicionEjecucion);

dificultad=lec.next().charAt(0);

System.out.println(dificultad);

obligatorio=lec.nextBoolean();

System.out.println(obligatorio);

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

aparato=lec.next();

posicionEjecucion=lec.next();

dificultad=lec.next().charAt(0);

obligatorio=lec.nextBoolean();

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

aparato=lec.next();

posicionEjecucion=lec.next();

dificultad=lec.next().charAt(0);

obligatorio=lec.nextBoolean();

if(g.altaMovimientoGimnasta(curp, nombre, aparato, posicionEjecucion, dificultad, obligatorio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

lec.nextLine();

nombre=lec.nextLine();

System.out.println(nombre);

ubicacion=lec.nextLine();

System.out.println(ubicacion);

categoria=lec.nextLine();

System.out.println(categoria);

anio=lec.nextInt();

System.out.println(anio);

logro=lec.nextInt();

System.out.println(logro);

if(g.altaCompetenciaGimnasta(curp, nombre, ubicacion, categoria, anio)) {

System.out.println("Alta exitosa");

}

else {

System.out.println("Error en alta");

}

System.out.println(g.setLogroCompetenciaGimnasta(curp, nombre, logro));

g.actualizaGimnasio();

this.g=g;

}

}

catch(FileNotFoundException fnfe) {

System.err.println("E R R O R "+fnfe);

System.exit(-1);

}

}

public GimnasioControlador(String titulo) {

super(titulo);

this.g=new Gimnasio();

leerArchivo.addActionListener(new EscuchadorArchivo());

super.infoComplBt.addActionListener(new EscuchadorImprime());

tablaDataBt.addActionListener(new EscuchadorImprime());

super.actualizaGymBt.addActionListener(new EscuchadorBotones1());

super.nutricionBt.addActionListener(new EscuchadorBotones1());

super.buscaGimnasBt.addActionListener(new EscuchadorBotones1());

super.mejoresCompetenciasNivel.addActionListener(new EscuchadorBotones1());

super.expAparato.addActionListener(new EscuchadorBotones1());

super.bajaGimnastaBt.addActionListener(new EscuchadorBotones2());

super.altaGimnastasBt.addActionListener(new EscuchadorBotones2()); //Nuevos Gimnastas

super.mayDificultad.addActionListener(new EscuchadorBotones2());

super.cuantosCompetencia.addActionListener(new EscuchadorBotones2());

super.horario.addActionListener(new EscuchadorBotones2());

super.antiguedad.addActionListener(new EscuchadorBotones2());

super.estilo.addActionListener(new EscuchadorBotones2());

super.logroCompBt.addActionListener(new EscuchadorBotones2());

super.gimNivel.addActionListener(new EscuchadorBotones3());

super.cuantosMovimiento.addActionListener(new EscuchadorBotones3());

super.altaLogrosBt.addActionListener(new EscuchadorBotones3());

super.bajaCompBt.addActionListener(new EscuchadorBotones3());

super.edad.addActionListener(new EscuchadorBotones3());

super.numComp.addActionListener(new EscuchadorBotones3());

super.buscaLogro.addActionListener(new EscuchadorBotones3());

super.listMovBt.addActionListener(new EscuchadorRadioBotones());

super.movMasDificil.addActionListener(new EscuchadorRadioBotones());

nutricionTxtF.setText("Ingrese curp del gimnasta");

estadTxtF.setText("Ingrese Info a buscar");

logrosTxtF1.setText("Nombre Competencia o Movimiento");

logrosTxtF2.setText("Lugar o Aparato");

logrosTxtF3.setText("Categoria o Posición de Ejecucion");

logrosTxtF4.setText("Año de competencia o dificultad de movimiento");

logrosTxtF5.setText("Posición resultante en competencia");

pantallaGimnastas.setText("Ingrese curp del gimnasta");

}

private class EscuchadorArchivo implements ActionListener{

public void actionPerformed(ActionEvent arg0) {

leeArchivo();

pantallaGral.setText("Archivo Cargado y Listo para usarse \n\n"+g.toString());

}

}

private class EscuchadorImprime implements ActionListener{

public void actionPerformed(ActionEvent e) {

String comando=e.getActionCommand();

StringBuilder txt=new StringBuilder();

if(comando.charAt(0)=='T') {

if(!g.isGimnasioVacio()) {

for(int j=0; j<g.getMAX\_NIVELES(); j++) {

txt.append("Gimnastas nivel"+(j+1));

if(g.getNumCompsNivel(j+1)!=0) {

txt.append(g.getGimnastasNivel(j)+"\n");

}

else {

txt.append("\t 0 Gimnastas \n");

}

}

}

else {

txt.append("Gimnasio sin gimnastas");

}

}

else {

txt.append(g.toString());

}

pantallaGral.setText(txt.toString());

}

}

private class EscuchadorBotones1 implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

String comando=e.getActionCommand();

String salida, entrada;

boolean musculo;

double aux;

int aux2;

switch(comando.charAt(0)) {

case 'N':

entrada=nutricionTxtF.getText();

musculo=nutMuscRdBt.isSelected();

System.out.println(musculo);

aux=g.getProteinasGimnasta(entrada, musculo);

salida=Double.toString(aux);

nutricionTxtF.setText(salida+" gramos al día");

break;

case 'B':

entrada=pantallaGimnastas.getText();

salida="Resultado de búsqueda en Gimnastas: ";

salida=salida.concat(g.getUnGimnasta(entrada));

pantallaGral.setText(salida);

break;

case 'A':

g.actualizaGimnasio();

pantallaGral.setText("Gimnasio actualizado");

break;

case 'M':

entrada=(String)nivelesCBox1.getSelectedItem();

salida=g.mejoresCompetenciasGimnastasNivel(entrada);

pantallaGral.setText("Mejores Competencias en el nivel "+entrada+" "+salida);

break;

case 'E':

entrada=logrosTxtF2.getText();

aux2=g.cuantasExpertiseAparato(entrada);

salida="Gimnastas con experiencia en aparato "+entrada;

salida=salida.concat(":\n"+Integer.toString(aux2));

pantallaGimnastas.setText(salida);

break;

}

}

}

private class EscuchadorBotones2 implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

String comando=e.getActionCommand();

String nombre, curp, entrada1, entrada2, aux2;

String salida="";

boolean alta;

int antiguedad, aux, horario;

double pesoKg;

switch(comando.charAt(0)) {

case 'N':

entrada1=(String)estilosCBox1.getSelectedItem();

aux2=(String)horarioCBox1.getSelectedItem();

entrada2=aux2.substring(0, 2);

entrada2=entrada2.concat(aux2.substring(3,5));

horario=Integer.parseInt(entrada2);

nombre=inscripcionesTxtF1.getText();

curp=inscripcionesTxtF2.getText();

antiguedad=Integer.parseInt(inscripcionesTxtF3.getText());

pesoKg=Double.parseDouble(inscripcionesTxtF4.getText());

alta=g.altaGimnasta(nombre, curp, antiguedad, entrada1, horario, pesoKg);

if(alta) {

salida="Alta exitosa";

}

else {

salida="E R R O R E N A L T A";

}

break;

case 'B':

curp=pantallaGimnastas.getText();

salida="Gimnasta dad@ de baja: ";

salida=salida.concat(g.bajaGimnasta(curp));

break;

case 'M':

salida="Gimnasta mayor dificultad ";

salida=salida.concat(g.gimnastaMayDificul());

break;

case 'C':

entrada1=estadTxtF.getText();

aux=g.cuantosGimnastasEnCompetencia(entrada1);

salida="Gimnastas en competencia "+entrada1;

salida=salida.concat(" "+Integer.toString(aux));

break;

case 'H':

aux2=(String)horarioCBox2.getSelectedItem();

entrada2=aux2.substring(0, 2);

entrada2=entrada2.concat(aux2.substring(3,5));

horario=Integer.parseInt(entrada2);

aux=g.gimnastasHorario(horario);

salida="Gimnastas en horario "+aux2;

salida=salida.concat(" son "+Integer.toString(aux));

break;

case 'A':

antiguedad=Integer.parseInt(estadTxtF.getText());

aux=g.cuantasAntiguedad(antiguedad);

salida="Gimnastas con antiguedad: "+antiguedad;

salida=salida.concat(" "+Integer.toString(aux));

break;

case 'E':

entrada1=(String)estilosCBox2.getSelectedItem();

aux=g.cuantasEstilo(entrada1);

salida="Gimnastas de estilo "+entrada1;

salida=salida.concat(" son "+Integer.toString(aux));

break;

case 'L':

entrada1=pantallaGimnastas.getText();

nombre=logrosTxtF1.getText();

entrada2=logrosTxtF5.getText();

aux=Integer.parseInt(entrada2);

salida=g.setLogroCompetenciaGimnasta(entrada1, nombre, aux);

break;

}

pantallaGral.setText(salida);

inscripcionesTxtF1.setText("Nombre");

inscripcionesTxtF2.setText("CURP");

inscripcionesTxtF3.setText("Años en gimnasio");

inscripcionesTxtF4.setText("Peso en Kg");

logrosTxtF5.setText("Posición resultante en competencia");

pantallaGimnastas.setText("Ingrese curp del gimnasta");

estadTxtF.setText("Ingrese Info a buscar");

}

}

private class EscuchadorBotones3 implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

String comando=e.getActionCommand();

String curp, nombre, txt2, txt3;

String salida="";

char txt4;

boolean alta;

int txt5, aux, edad;

switch(comando.charAt(0)) {

case 'M':

txt2=estadTxtF.getText();

aux=g.cuantosGimnastasConMovimiento(txt2);

salida="Gimnastas con movimiento "+txt2;

salida=salida.concat(" "+Integer.toString(aux));

break;

case 'A':

curp=pantallaGimnastas.getText();

nombre=logrosTxtF1.getText();

txt2=logrosTxtF2.getText();

txt3=logrosTxtF3.getText();

if(movBt.isSelected()) {

txt4=logrosTxtF4.getText().charAt(0);

alta=g.altaMovimientoGimnasta(curp, nombre, txt2, txt3, txt4, obligatorioBt.isSelected());

obligatorioBt.setSelected(false);;

}

else{

txt5=Integer.parseInt(logrosTxtF4.getText());

alta=g.altaCompetenciaGimnasta(curp, nombre, txt2, txt3, txt5);

}

if(alta) {

salida="Alta exitosa";

}

else {

salida="E R R O R E N A L T A";

}

break;

case 'B':

nombre=logrosTxtF1.getText();

curp=pantallaGimnastas.getText();

if(comando.charAt(1)=='a') { //Baja competencia

salida=g.bajaCompetenciaGimnasta(curp, nombre);

}

else if(movBt.isSelected()) { //Busca

salida=g.getUnMovimientoGimnasta(curp, nombre);

}

else if(compBt.isSelected()) {

salida=g.getUnaCompetenciaGimnasta(curp, nombre);

}

break;

case 'E':

txt2=estadTxtF.getText();

edad=Integer.parseInt(txt2);

aux=g.gimnastasEdad(edad);

salida="Número de gimnastas de edad "+edad+" años";

salida=salida.concat(": "+Integer.toString(aux));

break;

case 'C':

txt2=(String)nivelesCBox2.getSelectedItem();

aux=g.getNumCompsNivel(txt2);

salida="Número de competidores en nivel "+txt2;

salida=salida.concat(": "+Integer.toString(aux));

break;

case 'G':

txt2=(String)nivelesCBox1.getSelectedItem();

salida="Gimnastas en nivel "+txt2+": ";

salida=salida.concat(g.getGimnastasNivel(txt2));

break;

}

pantallaGral.setText(salida);

logrosTxtF1.setText("Nombre Competencia o Movimiento");

logrosTxtF2.setText("Lugar o Aparato");

logrosTxtF3.setText("Categoria o Posición de Ejecucion");

logrosTxtF4.setText("Año de competencia o dificultad de movimiento");

pantallaGimnastas.setText("Ingrese curp del gimnasta");

estadTxtF.setText("Ingrese Info a buscar");

}

}

private class EscuchadorRadioBotones implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

String comando=e.getActionCommand();

String txt, txt2;

String salida="";

txt=pantallaGimnastas.getText();

switch(comando.charAt(0)) {

case 'L': //lista de movimiento

txt2=logrosTxtF2.getText();

System.out.println(txt2);

salida="Lista de movimientos de Gimnasta "+txt+" en aparato "+txt2;

salida=salida.concat(":\n"+g.getListaMovimientosGimnastaAparato(txt, txt2));

break;

case 'M':

salida="Movimiento mas dificil de Gimnasta "+txt;

salida=salida.concat("\n"+g.getMovMasDificilGimnasta(txt));

break;

}

pantallaGral.setText(salida);

pantallaGimnastas.setText("Ingrese curp del gimnasta");

logrosTxtF2.setText("Lugar o Aparato");

}

}

public static void main(String[] args) {

GimnasioControlador yo=new GimnasioControlador("A VER SI AHORA SI....:");

}

}

Pruebas de Solución y Cambios correspondientes

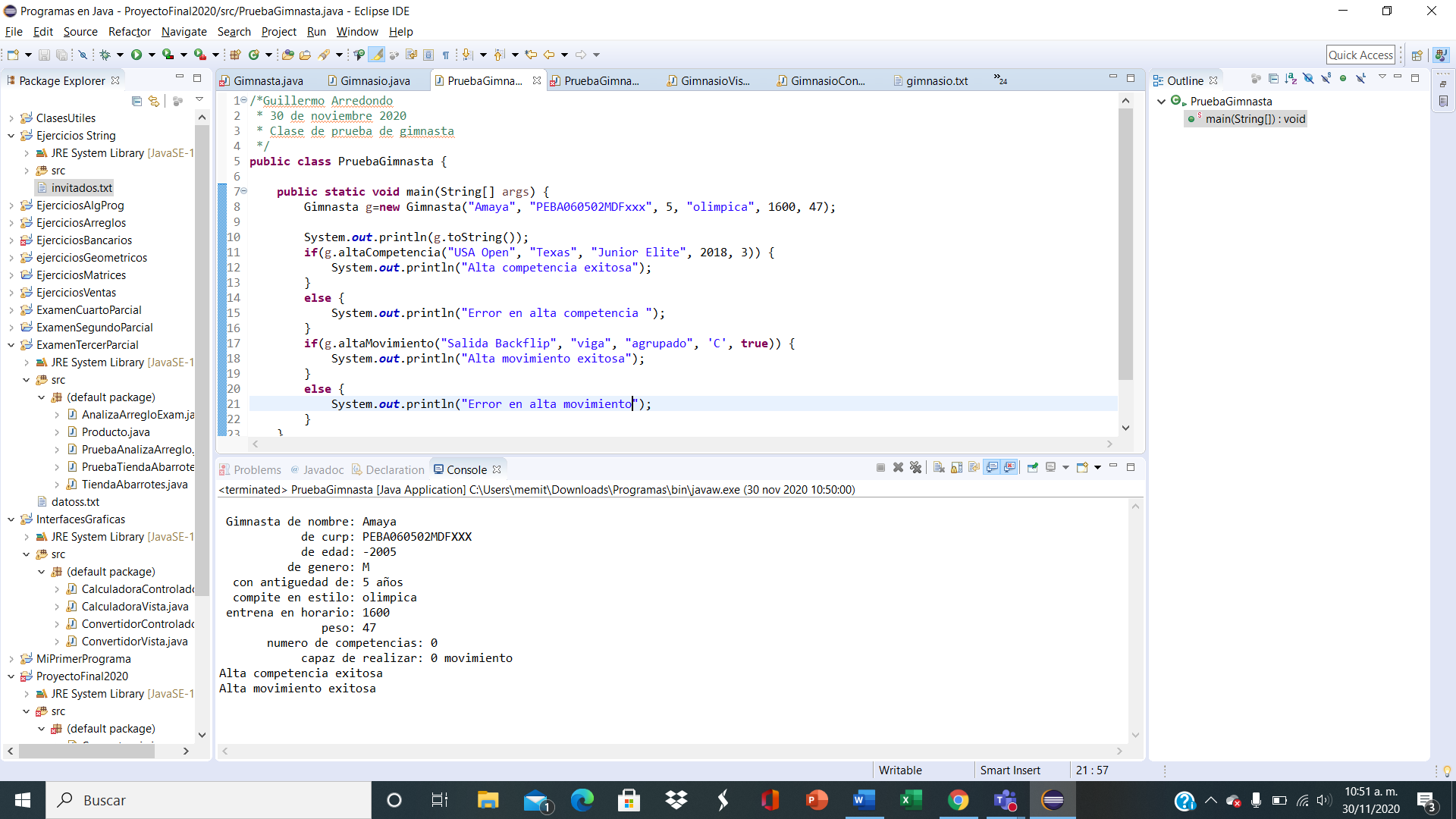


|  |  |  |
| --- | --- | --- |
| **Resumen Resultados** | **Nº Casos** | **%** |
| Satisfactorio | 26 | 74.29 |
| No Satisfactorio | 9 | 25.71 |
| **TOTAL** | **35** | **100** |

Se detectaron cinco principales errores en compilación:

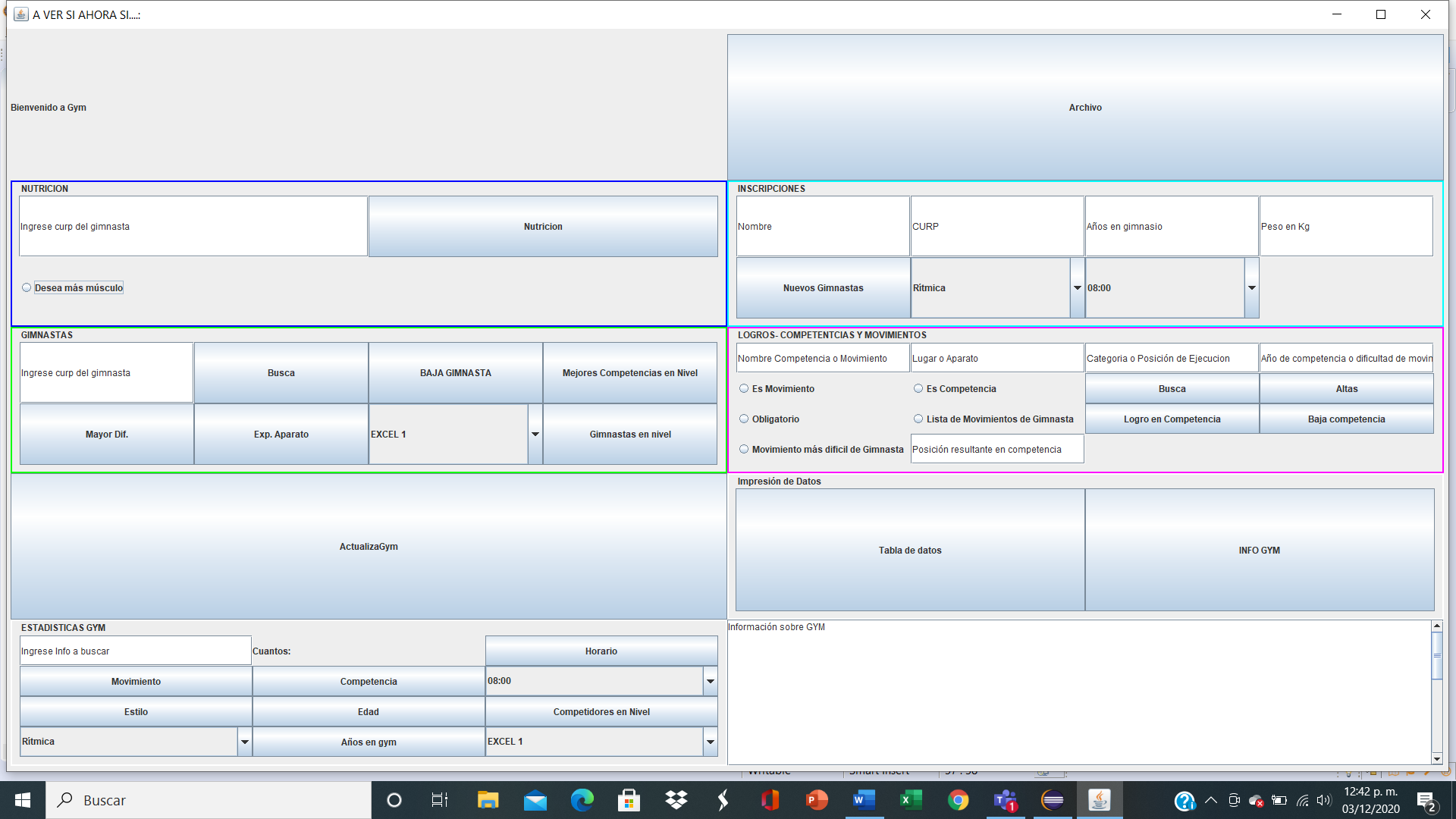
* Error de cálculo de edad: Debido a un mal uso de la clase auxiliar Calendar dentro del código de Clase Gimnasta, la edad resultante de los gimnastas no era un número entero de dos cifras sino el negativo del año en que nació. Se reconoció el error en obtención del año actual. Una vez corregido se amplió el evento satisfactorio para incluir el cálculo de edad con base en año y en mes de nacimiento y actual.
* Error en lectura de archivo: De manera sorprendente resultó que el código leía correctamente, incluso las lecturas de varias palabras, utilizando una lectura en falso para poder evitar el posible error en salto de línea que había ocurrido en proyectos pasados. Después de un análisis línea por línea de código y archivo se descubrió que el error había sido la falta de un dato en el archivo por descuido del autor. Se corrigió y la lectura de archivo resultó exitosa.
* Error en despliegue adecuado de múltiples paneles: La intención original del diseño de la Interfaz Gráfica de Usuario era poder realizar un cambio entre paneles conforme la acción de dar click en botones en un panel del lado izquierdo, esto resultó insatisfactorio, y por practicidad se decidió cambiar de diseño a uno más simple. La nueva organización de paneles funcionó adecuadamente en todos los aspectos.
* Error en generación de lista de movimientos: Como se mencionó anteriormente, curiosamente durante las pruebas con archivo el método getListaMovimientosGimnastaAparato no presentó ningún inconveniente; sin embargo, durante la llamada al método desde una interfaz gráfica, el programa se trabó y entraba a un ciclo infinito. Se desconoce porque no ocurría durante el archivo. Después de un minucioso análisis de los métodos involucrados y la creación de ArrayList, se descubrió que el error se debía a la carencia del incremento de un contador en un ciclo de estructura *while*. Se solucionó adecuadamente y el método funcionó de forma idónea.
* Error en aplicación booleana de más músculo para cálculo de proteínas: Se descubrió que debido al establecimiento de un ButtonGroup para misma variable booleana (que posteriormente resultó innecesaria) se producía un error de imposibilidad de apagar el botón más músculo una vez encendido. Esto producía una deficiencia en el cálculo apropiado y funcionamiento de la interfaz. Accidentalmente, se descubrió que se debía al establecimiento de la agrupación de botones, una vez eliminada y separado los botones, el error se solucionó.

Asimismo, se presentaron algunos errores menores en ejecución de búsquedas, debido al descuido del autor, puesto que ingresaba un arreglo equivocado a la búsqueda y lógicamente nunca se encontraba el resultado esperado. Análogamente, existieron errores menores en interfaz de usuario debido al descuido en el CURP (se colocaba fecha de nacimiento en formato dd/mm/aa cuando debería ser aa/mm/dd).



[Error en cálculo de edad]

**Manual de Usuario para Interfaz Gráfica**



[Vista de Interfaz Gráfica]

La vista anterior es la única desplegable durante el programa, a través de ella se pueden observar diferentes paneles que explicaremos a continuación empezando de izquierda a derecha y de arriba hacia abajo:

En primera instancia encontramos un logo que se mantendrá visible todo el tiempo, en él se lee “Bienvenido a Gym” como un acto de formalidad.

**Archivo**

En segundo lugar, se puede observar un botón en el que se lee Archivo. Este botón lo que hace es llamar a un método de código a través del cual se logra hacer una carga completa y satisfactoria de un archivo previamente diseñado por el autor y creado principalmente para motivos de prueba, este archivo es el que provee al Gimnasio de un nombre, ubicación y encargo específicos, pero no es indispensable para el uso de la interfaz, puesto que al dar de alta contenido de igual forma se podrá hacer, aunque dentro de un gimnasio sin información propia.

**Nutricion:**

A continuación, se observa un panel llamado “NUTRICIÓN” dentro del cual se lleva a cabo la simple acción de calculas las proteínas recomendadas para un determinado gimnasta. Para hacer uso de este panel es necesario seguir los siguientes pasos:

1. Escribir dentro del campo de texto (recuadro blanco del lado izquierdo) el curp del gimnasta al que se le desea calcular las proteínas necesarias. Es importante asegurarse que el curp esté escrito conforme se dio de alta, sin embargo, no hay problema entre distinción de mayúsculas.
2. Posteriormente, indicar si se desea incrementar músculo dando click sobre el botón de selección ubicado debajo del campo de texto.
3. Por último, dar click sobre el botón “Nutrición”.

**Inscripciones:**

Después, observamos el panel “INSCRIPCIONES” dentro del cual se encuentran cuatro cuadros de texto, cada uno cuenta con la información que se debe proporcionar en él. La función de este recuadro es la de poner dar de alta e inscribir un gimnasta en el gimnasio, es importante indicar que esta inscripción dará por sentado que el gimnasta es completamente nuevo, por lo que se colocará dentro del nivel 1. Para poder aumentar el nivel es necesario dar de alta los correspondientes movimientos y competencias de las que es o ha sido partícipe y, posteriormente dar click en el botón “Actualiza Gym” del que se hablará más adelante.

Ahora bien, los pasos a seguir en inscripciones son:

1. Ingresar el nombre del gimnasta a inscribir en el primer cuadro de texto.
2. Ingresar el curp del gimnasta en el segundo cuadro de texto.
3. Ingresar los años que lleva haciendo gimnasia en el tercer cuadro de texto.
4. Ingresar el peso del gimnasta en Kg, no es necesario escribir Kg.
5. Seleccionar un estilo en la primera cada de texto debajo del segundo cuadro de texto.
6. Seleccionar un horario en el que asistirá a entrenar.
7. Dar click en el botón nuevo Gimnast; inmediatamente se desplegará el texto: “alta exitosa” en el panel de texto de hasta abajo.

**Gimnastas:**

Posteriormente, observamos un recuadro llamado “Gimnastas”, el cual está destinado a explotar la información de los gimnastas inscritos. En el se pedirá siempre que se escriba el curp del gimnasta dentro del recuadro blanco.

Busca y baja gimnasta

1. Escribir el curp en el recuadro blanco.
2. Oprimir el botón que se desea ejecutar.
3. Se desplegará información en el área de texto de la parte inferior.

Mejores competencias en nivel y Gimnastas en Nivel

1. Seleccionar un nivel en el recuadro en el que se lee “EXCEL 1”.
2. Dar click en “Mejores Competencias por Nivel” o “Gimnastas en Nivel”.
3. Se desplegará información en el área de texto de la parte inferior.

Mayor dificultad y Expert@ en Aparato

1. Dar click directamente en el botón, esto accionará el comando de búsqueda en toda la matriz.

**Logros- Competencias y Movimientos:**

Como se indica en el título este panel está destinado a la información sobre competencias y nuevos movimientos ejecutables.

En el podemos observar cuatro recuadros de texto en la parte superior, en estos se establece texto sobre la información que puede recibir.

Primer cuadro de texto: nombre de movimiento o competencia.

Segundo cuadro de texto: Lugar de competencia o aparato de movimiento.

Tercer cuadro de texto: Categoría de competencia o posición de ejecución de movimiento.

Cuarto cuadro de texto: Dificultad de movimiento o año de competencia.

Altas:

1. Ingresar curp del gimnasta en la pantalla del panel de la izquierda “Gimnastas”
2. Ingresar toda la información correspondiente al movimiento o competencia según los recuadros antes mencionados
3. En caso de ser movimiento, elegir si es obligatorio o no (botón “Obligatorio”).
4. Elegir uno de los dos botones, “Es movimiento” o “Es competencia”
5. Dar click en “Altas”.
6. Se desplegará información en el texto de la parte inferior de la ventan.

Busca:

1. Ingresar curp del gimnasta en la pantalla del panel de la izquierda “Gimnastas”
2. Ingresar el nombre del movimiento o competencia
3. Elegir uno de los dos botones, “Es movimiento” o “Es competencia”
4. Dar click en “Busca”.

Baja competencia:

1. Ingresar curp del gimnasta en la pantalla del panel de la izquierda “Gimnastas”
2. Ingresar el nombre de competencia

Logro competencia:

1. Ingresar curp del gimnasta en la pantalla del panel de la izquierda “Gimnastas”
2. Ingresar el nombre de competencia
3. Ingresar en el cuadro de texto inferior (en el mismo panel) el número de posición en que quedó el atleta durante la competencia

Movimiento más difícil gimnasta y Lista de movimientos de Gimnasta

1. Ingresar curp del gimnasta en la pantalla del panel de la izquierda “Gimnastas”
2. Dar click en el botón “Movimiento más difícil de gimnasta:” o “Lista de movimientos de Gimnasta”

**Actualiza Gym**

Este botón realiza una de las acciones más importantes del programa, puesto que genera un método que localiza el nivel en el que se encuentra cada uno de los gimnastas y lo compara con el cálculo correspondiente del gimnasta. Si es diferente se elimina al gimnasta de su correspondiente nivel y se inserta en el adecuado.

**Tabla de Datos e Info Gym**

Cada uno de los botones despliega la correspondiente información:

Tabla de datos: Despliega la información de los nombres de los gimnastas en cada uno de los niveles.

Info Gym: Despliega toda la información del gimnasio.

**Estadísticas Gym:**

Este panel está diseñado para la aplicación, principalmente numérica de datos estadísticos que pudieran resultar útiles para el gimnasio.

Movimiento, Competencia, Edad y Años en Gym:

Son utilizables a partir del siguiente proceso:

1. Ingresar el dato correspondiente a cada botón dentro del recuadro de Texto que menciona: “Ingrese Info a Buscar”:
   1. Movimiento: nombre del movimiento que se desea consultar (recordar que este proceso únicamente desplegará el número de gimnastas que puedan realizar ese movimiento).
   2. Competencia: nombre de la competencia que se desea consultar (recordar que este proceso únicamente desplegará el número de gimnastas que hayan participado en dicha competencia). Este proceso no considera aquellas competencias que cuenten con un logro establecido en cero, pues como se explicó antes, esto se considera una competencia sin asistencia.
   3. Edad: ingresar la edad que se desea buscar, no rangos (aunque podría resultar interesante implementarlo). No es necesario agregar la palabra “años”.
   4. Años en Gym: ingresar la cantidad de años haciendo gimnasia de la que se desea obtener el número de gimnastas que lo cumplan.
2. Dar click en el botón del tipo de búsqueda que se empleará.

Estilo, Horario y Competidores en Nivel

1. Seleccionar la información correspondiente a cada caso:
   1. Estilo: Seleccionar el estilo que se desea buscar en el recuadro de selección inferior al botón.
   2. Horario: Seleccionar el horario que se desea buscar en el recuadro de selección inferior al botón.
   3. Competidores en Nivel: Seleccionar el nivel que se desea buscar en el recuadro de selección inferior al botón. Recordar que este método desplegará el número de competidores en un determinado nivel, para encontrar quienes son esos gimnastas regresar el botón “Gimnastas en Nivel” en el Panel **“Gimnastas”.**

**Pantalla de Texto esquina inferior derecha:**

Esta pantalla de texto es para desplegar información únicamente, en caso de escribir algo dentro de ella, el programa no ejecutará ninguna acción.

**Conclusiones**

El proyecto ha demostrado un gran esfuerzo por parte del autor, ha exigido un amplio grado de concentración, investigación, razonamiento y organización. Todas y cada una de suma importancia para el desarrollo de un proyecto adecuado. A pesar de las múltiples dificultades presentadas a lo largo del proyecto, sobre todo en la creación de la interfaz gráfica de usuario, el proyecto a demostrado ser una prueba del conocimiento adquirido a lo largo del curso de Algoritmos y Programas y una excelente prueba de lo mucho que se consiguió aprender y lo divertido que puede ser frustrarse por una falla de compilación. Aunque personalmente ya conocía sobre el funcionamiento de un gimnasio y las especificaciones que establece el deporte de la gimnasia para el establecimiento de niveles, movimientos y competencias, se requirió de una investigación un tanto exhaustiva para realizar el proyecto lo más semejante a la realidad como se pudiera.

Fuera de la formalidad del documento, considero que este proyecto me ha apoyado a crecer como persona en el ámbito de innovación y creatividad, además probó la importancia del trabajo continua y enfocado en más de un área de estudio. Esto debido al cambio entre programar y establecer un buen código junto con el diseño de una interfaz gráfica adecuada, interesante, entendible y bien organizada. Agradezco mucho la realización de este proyecto y aunque no ha quedado perfecto y como esperaba en un inicio que resultara, puedo observar un excelente desempeño y una satisfacción muy grande.

Gracias a este programa ahora los gimnasios pueden solucionar su problema sobre registro y manejo de información de grandes volúmenes de datos sin mayor problema. Se considera interesante poder entender más a fondo la utilización de diferentes componentes de interfaz gráfica de usuario y los cambios de vista. Se espera que en un futuro se puedan implementar y de este modo mejorar el proyecto.