Trabajo Grupal

Fundamentos de informática II

Hecho por :

[redacted]

- Laura Ferrer Haba

[redacted]

Contenido

AUTORES
Laura Ferrer Haba, .

DESCRIPCIÓN DEL PROBLEMA

Hemos hecho un programa que permite crear personajes de Star Wars.

DESCRIPCIÓN DE LAS CLASES

En este apartado del trabajo describiremos la clase y explicaremos la funcionalidad de cada una.

Clase Personaje

Es la clase abstracta **Personaje**. La clase representará cualquier tipo de personaje de Star Wars. Para ello usará los archivos *"Personaje.h"* y *"Personaje.cpp"*. Un personaje viene determinado por su **nombre** (de tipo *string* y privado), **apellido** (de tipo *string* y privado), **raza** (de tipo *string* y privado), **bando** (de tipo *string* y privado), **planeta de nacimiento** (de tipo *string* y privado) y su **experiencia** (de tipo *int* y privado). Todos los atributos tienen sus constructores y modificadores. Un personaje puede ser creado a partir de su nombre, apellido, raza, bando, planeta de nacimiento y su experiencia. Además, los métodos públicos que tiene la clase son:

- **virtual void mostrarPersonaje()**: método que muestra por pantalla todas las características de un personaje. La función no recibe parámetros, no devuelve nada, y muestra por pantalla todas las características de un Personaje.
- **virtual int supervivencia()**: método abstracto puro que devuelve la experiencia de cualquier personaje.

PERSONAJE
Personaje()
Personaje(…)
getNombre
getApellido
getRaza
getBando
getPlanetaNacimiento
getExperiencia

PERSONAJE
setNombre
setApellido
setRaza
setBando
setPlanetaNacimiento
setExperiencia
mostrarPersonaje
supervivencia(int exp) = 0

- **Atributos de la clase Personaje:**

PERSONAJE
nombre
apellido
raza
bando
planetaNacimiento
experiencia

Clase Jedi

Es la clase abstracta **Jedi** que hereda de **Personaje**. La clase Jedi es un tipo de personaje que se caracteriza por la conexión que tiene con la fuerza y los midiclorianos. Para ello usará el archivo *"Jedi.h"* y *"Jedi.cpp"* con las siguientes características:

- **void mostrarJedi()**: método que muestra por pantalla todas las características de un Jedi. La función no recibe parámetros, no devuelve nada y muestra por pantalla todas las características de un Jedi.
- **virtual int poder()**: método abstracto puro que devuelve el poder de cualquier personaje Jedi.
- Un Jedi viene definido, además de por su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, por la **conexión con la fuerza** (de tipo *string* y privado) y **sus midiclorianos** (de tipo *int* y privado).
- Un Jedi se construye **solo** a partir de su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, la conexión con la fuerza, y sus midiclorianos.

JEDI
Jedi()
Jedi(…)
Jedi(…/…)
getConexion
getMidiclorianos
setConexion
setMidiclorianos
mostrarJedi
supervivencia(int exp)
poder(int pod) = 0

- **Atributos de la clase Jedi:**

JEDI
conexion
midiclorianos

Clase GranMaestro

Es la clase **GranMaestro** que hereda de **Jedi**. La clase **GranMaestro** es un tipo de personaje Jedi con las siguientes características:

- Tiene la propiedad habilidad llamada **habilidad** (de tipo *string* y privada), la propiedad número de Aprendices llamada **numAprendices** (de tipo *int* y privada), y la propiedad nombre del Aprendiz más relevante llamada **nombreAprendizRelevante** (de tipo *string* y privada).
- Se construye **únicamente** recibiendo todos los valores de las propiedades que definen a un Gran Maestro.
- Define un método visualizar, mostrando por pantalla todas las características del Gran Maestro.

GRANMAESTRO
GranMaestro()
GranMaestro(…)
GranMaestro(…/…)
getHabilidad
getNumAprendices
getNombreAprendizRelevante
setHabilidad
setNumAprendices
setNombreAprendizRelvante
mostrarGranMaestro
poder(int pod)

- **Atributos de la clase GranMaestro:**

GRANMAESTRO
habilidad
numAprendices
nombreAprendizRelevante

Clase Padawan

Es la clase **Padawan** que hereda de **Jedi**. La clase **Padawan** es un tipo de personaje Jedi con las siguientes características:

- Tiene la propiedad número de maestros llamada **numMaestros** (de tipo *int* y privada), y la propiedad nombre del Maestro más relevante llamada **nombreMaestroRelevante** (de tipo *string* y privada).
- Se construye **únicamente** recibiendo todos los valores de las propiedades que definen a un Padawan.
- Define un método visualizar, mostrando por pantalla todas las características del Padawan.

PADAWAN
Padawan()
Padawan(…)
Padawan(…/…)
getNumMaestros
getNombreMaestroRelevante
setNumMaestros
setNombreMaestroRelevante
mostrarPadwan
poder(int pod)

- **Atributos de la clase Padawan:**

PADAWAN
numMaestros
nombreMaestroRelevante

Clase Sith

Es la clase **Sith** que hereda de **Personaje**. La clase Sith es un tipo de personaje que se caracteriza por el maestro que tuvo el Sith, el aprendiz y el rango de la orden . Para ello usará el archivo *"Sith.h"* y *"Sith.cpp"* con las siguientes características:

- **void mostrarSith()**: método que muestra por pantalla todas las características de un Sith. La función no recibe parámetros, no devuelve nada muestra por pantalla todas las características de un Sith.
- Un Sith viene definido por, además de por su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, por su **maestro** (de tipo *string* y privada), su **aprendiz** (de tipo *string* y privada) y el **rango de la orden** (de tipo *string* y privada).
- Un Sith se construye **solo** a partir de su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, su maestro, su aprendiz y su rango en la orden.

SITH
Sith()
Sith(...)
Sith(…/...)
getMaestro
getAprendiz
getRangoOrden
setMaestro
setAprendiz
setRangoOrden
mostrarSith
supervivencia(int exp)

- **Atributos de la clase Sith:**

SITH
maestro
aprendiz
rangoOrden

Clase SinFuerza

Es la clase abstracta **SinFuerza** que hereda de **Personaje**. La clase SinFuerza es un tipo de personaje que se caracteriza por el gremio en el que se encuentra y los idiomas que habla. Para ello usará el archivo *"SinFuerza.h"* y *"SinFuerza.cpp"* con las siguientes características:

- **void mostrarSinFuerza()**: método que muestra por pantalla todas las características de un SinFuerza. La función no recibe parámetros, no devuelve nada muestra por pantalla todas las características de un SinFuerza.
- **virtual int destreza()**: método abstracto puro que devuelve la destreza de cualquier personaje sin fuerza.
- Un sin fuerza viene definido, además de por su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, por su **gremio** (de tipo *string* y privado) y su **idioma** (de tipo *string* y privado).
- Un Sin fuerza se construye **solo** a partir de su nombre, apellido, raza, bando, planeta de nacimiento, experiencia, gremio, y su idioma.

SINFUERZA
SinFuerza()
SinFuerza(…)
SinFuerza(…/…)
getGremio
getIdioma
setGremio
setIdioma
mostrarSinFuerza
supervivencia(int exp)
destreza(int des) = 0

- **Atributos de la clase SinFuerza:**

SINFUERZA
gremio
idioma

Clase Contrabandista

Es la clase **Contrabandista** que hereda de **SinFuerza**. La clase **Contrabandista** es un tipo de personaje Jedi con las siguientes características:

- Tiene la propiedad arma llamada **arma** (de tipo *string* y privada), y la propiedad nave llamada **nave** (de tipo *string* y privada) y la propiedad contratos llamada **contratos** (de tipo *int* y privada).
- Se construye **únicamente** recibiendo todos los valores de las propiedades que definen a un Contrabandista.
- Define un método visualizar, mostrando por pantalla todas las características del Contrabandista.

CONTRABANDISTA
Contrabandista()
Contrabandista(…)
Contrabandista(…/…)
getArma
getNave
getContratos
setArma
setNave
setContratos
mostrarContrabandista
destreza(int des)

- **Atributos de la clase Contrabandista:**

CONTRABANDISTA
arma
nave
contratos

Clase Mercenario

Es la clase **Mercenario** que hereda de **SinFuerza**. La clase **Mercenario** es un tipo de personaje Jedi con las siguientes características:

- Tiene la propiedad arma llamada **arma** (de tipo *string* y privada), y la propiedad nave llamada **jefe** (de tipo *string* y privada) y la propiedad contratos llamada **asesinatos** (de tipo *int* y privada).
- Se construye **únicamente** recibiendo todos los valores de las propiedades que definen a un Mercenario.
- Define un método visualizar, mostrando por pantalla todas las características del Mercenario.

MERCENARIO
Mercenario()
Mercenario(…)
Mercenario(…/…)
getArma
getJefe
getAsesinatos
setArma
setJefe
setAsesinatos
mostrarMercenario
destreza(in des)

- **Atributos de la clase Mercenario:**

MERCENARIO
arma
jefe
asesinatos

Clase CampareObject

Se trata de una clase que compara los personajes de Star Wars dependiendo de la clase que sean. Para ello, hemos creado tres métodos (uno de la clase Personaje, otro de la clase Jedi y otro de la clase GranMaestro).

El primer método **int comparePersonaje(Personaje* p1, Personaje* p2)** hace que se puedan comparar la experiencia de todos los personajes. El método compara dos personajes de la siguiente forma:

- devuelve 0 si el personaje1 y personaje2 tienen la misma experiencia, e imprime por pantalla "Los personajes *nombreP1* y *nombreP2* tienen la misma experiencia."
- devuelve -1 si el personaje2 tiene menor experiencia que el personaje1, e imprime por pantalla "El personaje *nombreP2* tienes más experiencia que el personaje *nombreP1*."
- devuelve 1 si el personaje1 tiene menor experiencia que el personaje2, e imprime por pantalla "El personaje *nombreP1* tienes más experiencia que el personaje *nombreP2*."

El segundo método **int compareJedi(Jedi* j1, Jedi* j2)** hace que se puedan comparar los midiclorianos de los personajes Jedis. El método compara dos Jedis de la siguiente forma:

- devuelve 0 si el jedi1 y jedi2 tienen la misma experiencia, e imprime por pantalla "Los Jedis *nombreP1* y *nombreP2* tienen el mismo número de midiclorianos."
- devuelve -1 si el jedi2 tiene menor experiencia que el jedi1, e imprime por pantalla "El jedi *nombreP2* tienes más midiclorianos que el jedi *nombreP1*."
- devuelve 1 si el jedi1 tiene menor experiencia que el jedi2, e imprime por pantalla "El jedi *nombreP1* tienes más midiclorianos que el jedi *nombreP2*."

El tercer método i**nt compareGranMaestro(GranMaestro* g1, GranMaestro* g2)** hace que se puedan comparar el número de aprendices de los personajes Gran Maestro. El método compara dos Gran Maestro de la siguiente forma:

- devuelve 0 si el granmaestro1 y granmaestro2 tienen la misma experiencia, e imprime por pantalla "Los grandes maestros *nombreP1* y *nombreP2* tienen el mismo número de aprendices."
- devuelve -1 si el granmaestro2 tiene menor experiencia que el granmaestro1, e imprime por pantalla "El gran maestro *nombreP2* tienes más número de aprendices que el gran maestro *nombreP1*."
- devuelve 1 si el granmaestro1 tiene menor experiencia que el granmaestro2, e imprime por pantalla "El gran maestro *nombreP1* tienes más número de aprendices que el gran maestro *nombreP2*."

COMPAREOBJECT
ComparePersonaje ()
Comparejedi ()
CompareGranMaestro ()

Clase ColeccionGranMaestro

Esta clase trata de crear listas de Gran Maestros a partir de los personajes creados anteriormente. La clase ColeccionGranMaestro tiene un atributo coleccionGM (tipo *lista* y privada). La clase tiene tres métodos que se definen:

- **void insertarGranMaestro(GranMaestro gm)**: inserta un personaje tipo GranMaestro a la lista de GranMaestro.
- **void eliminarGranMaestro():** elimina el primer personaje GranMaestro de la lista de GranMaestro.
- **void mostrarColeccionGranMaestro():** muestra por pantalla todos los personajes insertados de la lista de GranMaestro.
- Un **ejemplo** de esto sería: creamos tres personajes de la clase GranMaestro y los insertamos en la lista y los mostramos por pantalla. En la pantalla aparecerán los tres Gran Maestros. Después eliminaremos los que queramos, y volveremos a mostrar por pantalla. Teniendo así la colección GranMaestro.

COLECCIONGRANMAESTRO
insertarGranMaestro(gm)
eliminarGranMaestro ()
mostrarCoelccionGranMaestro ()

- **Atributos de la clase ColeccionGranMaestro:**

COLECCIONGRANMAESTRO
list<> coleccionGM

Clase ColeccionPadawan

Esta clase trata de crear vectores de Padawan a partir de los personajes creados anteriormente. La clase ColeccionPadawan tiene un atributo coleccionPadaw (tipo *vector* y privado). La clase tiene tres métodos que se definen:

- **void insertarPadawan(Padawan p)** inserta un personaje tipo Padawan a la lista de Padawan.
- **void eliminarPadawan():** elimina el primer personaje Padawan de la lista de Padawan.
- **void mostrarColeccionPadawan():** muestra por pantalla todos los personajes insertados de la lista de Padawan.
- Un **ejemplo** de esto sería: creamos tres personajes de la clase Padawan y los insertamos en el vector y lo mostramos por pantalla. En la pantalla aparecerán los tres Padawanes. Después eliminaremos los que queramos, y volveremos a mostrar por pantalla. Teniendo así la colección Padawan.

COLECCIONPADAWAN
insertarPadawan(p)
eliminarPadawan(pos)
mostrarColeccionPadawan ()

- **Atributos de la clase ColeccionPadawan:**

COLECCIONPADAWAN
vector<> coleccionPadaw

Clase ColeccionContrabandista

Esta clase trata de crear pilas de Contrabandista a partir de los personajes creados anteriormente. La clase ColeccionContrabandista tiene dos atributos, uno coleccionCB (tipo *pila* y privada) y otro auxColeccionCB (tipo *pila* y privado). La clase tiene tres métodos que se definen:

- **void insertarContrabandista(Contrabandista cb)** inserta un personaje tipo Contrabandista a la lista de Contrabandista.
- **void eliminarContrabandista():** elimina el primer personaje Contrabandista de la lista de Contrabandista.
- **void mostrarColeccionContrabandista():** muestra por pantalla todos los personajes insertados de la lista de Contrabandista.
- Un **ejemplo** de esto sería: creamos tres personajes de la clase Contrabandista y los insertamos en la pila y lo mostramos por pantalla. En la pantalla aparecerán los tres Contrabandistas. Después eliminaremos los que queramos, y volveremos a mostrar por pantalla. Teniendo así la colección Contrabandista.

COLECCIONCONTRABANDISTA
insertarContrabandista(cb)
eliminarContrabandista ()
mostrarColeccionContrabandista ()

- **Atributos de la clase ColeccionContrabandista:**

COLECCIONCONTRABANDISTA
stack<> coleccionCB
stack<> auxColeccionCB

Clase ColeccionMercenario

Esta clase trata de crear colas de Mercenario a partir de los personajes creados anteriormente. La clase ColeccionMercenario tiene dos atributos, uno coleccionMercen (tipo *cola* y privada) y otro auxColeccionMercen (tipo *cola* y privado). La clase tiene tres métodos que se definen:

- **void insertarMercenario(Mercenario m)** inserta un personaje tipo Mercenario a la lista de Mercenario.
- **void eliminarMercenario():** elimina el primer personaje Mercenario de la lista de Mercenario.
- **void mostrarColeccionMercenario():** muestra por pantalla todos los personajes insertados de la lista de Mercenario.
- Un **ejemplo** de esto sería: creamos tres personajes de la clase Mercenario y los insertamos en la cola y lo mostramos por pantalla. En la pantalla aparecerán los tres Mercenarios. Después eliminaremos los que queramos, y volveremos a mostrar por pantalla. Teniendo así la colección Mercenario.

COELCCIONMERCENARIO
insertarMercenario(m)
eliminarMerceanrio ()
mostrarColeccionMercenario ()

- **Atributos de la clase ColeccionMercenario:**

COELCCIONMERCENARIO
queue<> coleccionMercen
queue<> auxColeccionMercen

Clase CargaDatos

Esta clase carga los datos de cada clase nieta (GranMaestro, Padawan, Contrabandista y Mercenario), en un txt. La clase tiene doce métodos, un método que crea la lista de cada clase (4 métodos en total), un método que guarda la colección en un txt (4 métodos en total) y un método que agrega el contenido en un txt que no existe (4 métodos en total), cada uno se define como:

- **list<GranMaestro> cargaDatosGM(string filenameGM):** crea la lista del personaje GranMaestro.
- **list<Padawan> cargaDatosPadaw(string filenameP):** crea la lista del personaje Padawan.
- **list<Contrabandista> cargaDatosCB(string filenameCB):** crea la lista del personaje Contrabandista.
- **list<Mercenario> cargaDatosMercen(string filenameM):** crea la lista del personaje Mercenario.
- **int guardarColeccionGM(string filenameGM, list<GranMaestro> listaGM):** guarda la colección del personaje GranMaestro en un txt ya existente, es decir, si hay algo escrito en el txt lo elimina y escribe de nuevo.
- **int guardarColeccionP(string filenameP, vector<Padawan> vectorP):** guarda la colección del personaje Padawan en un txt ya existente, es decir, si hay algo escrito en el txt lo elimina y escribe de nuevo.
- **int guardarColeccionCB(string filenameCB, stack<Contrabandista> pilaCB):** guarda la colección del personaje Contrabandista en un txt ya existente, es decir, si hay algo escrito en el txt lo elimina y escribe de nuevo.
- **int guardarColeccionMercen(string filenameMercen, queue<Mercenario> ColaM):** guarda la colección del personaje Mercenario en un txt ya existente, es decir, si hay algo escrito en el txt lo elimina y escribe de nuevo.
- **int guardarContenidoGM(string filenameGM, list<GranMaestro> listaGM):** guarda la colección del personaje GranMaestro en un txt ya existente y si no hay lo crea, es decir, si hay algo escrito en el txt lo deja y escribe al final del fichero.
- **int guardarContenidoP(string filenameP, vector<Padawan> vectorP):** guarda la colección del personaje Padawan en un txt ya existente y si no hay lo crea, es decir, si hay algo escrito en el txt lo deja y escribe al final del fichero.
- **int guardarContenidoCB(string filenameCB, stack<Contrabandista> pilaCB):** guarda la colección del personaje Contrabandista en un txt ya existente y si no hay lo crea, es decir, si hay algo escrito en el txt lo deja y escribe al final del fichero.
- **int guardarContenidoMercen(string filenameMercen, queue<Mercenario> ColaM):** guarda la colección del personaje Mercenario en un txt ya existente y si no hay lo crea, es decir, si hay algo escrito en el txt lo deja y escribe al final del fichero.

CARGADATOS
list<> cargaDatosGM(filenameGM)
list<> cargaDatosPadaw(filenameP)
list<> cargaDatosCB(filenameCB)
list<> cargaDatosMercen(filenameM)
int guardarColeccionGM()
int guardarColeccionP()

CARGADATOS
int guardarColeccionCB()
int guardarColeccionMercen()
int guardarContenidoGM()
int guardarContenidoP()
int guardarContenidoCB()
int guardarContenidoMercen()

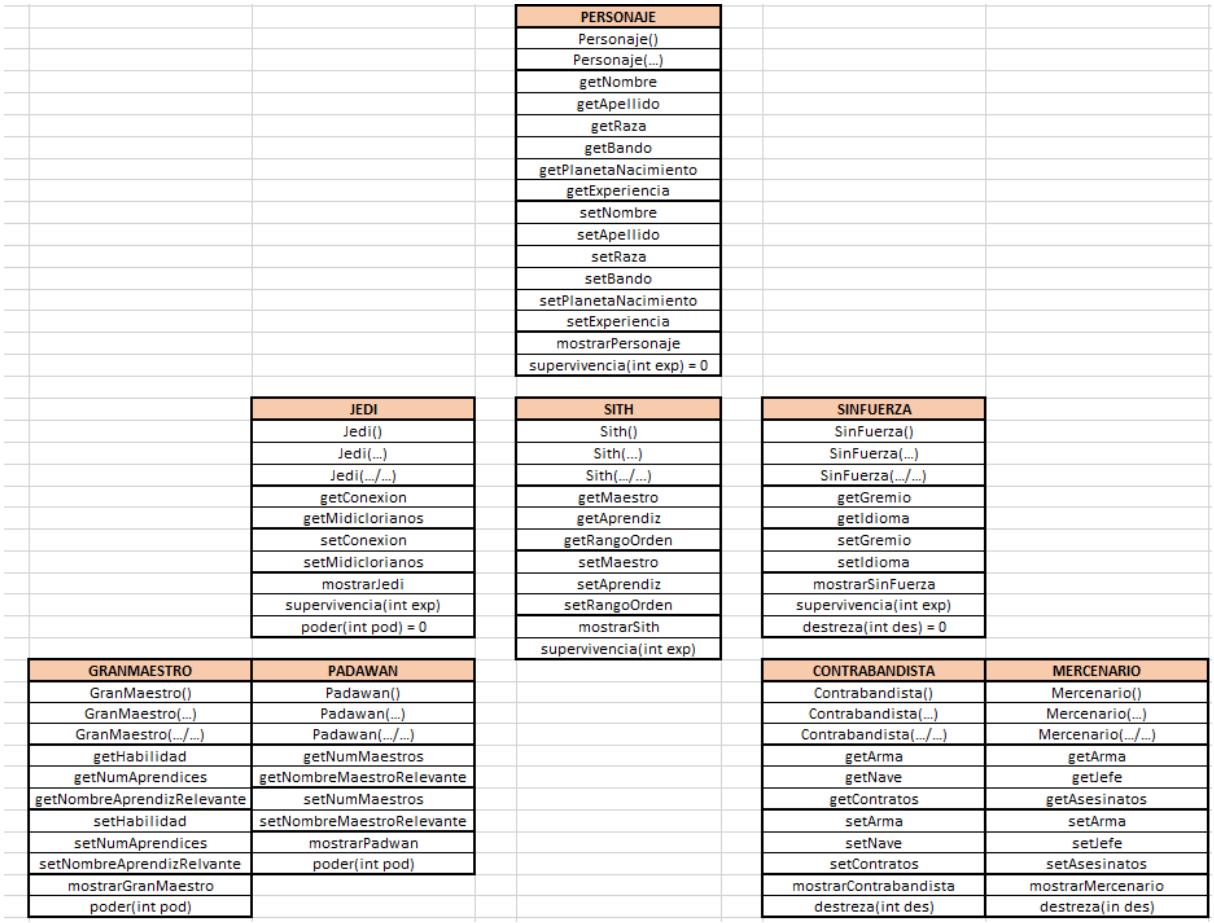
Clase AlgoritmoByO

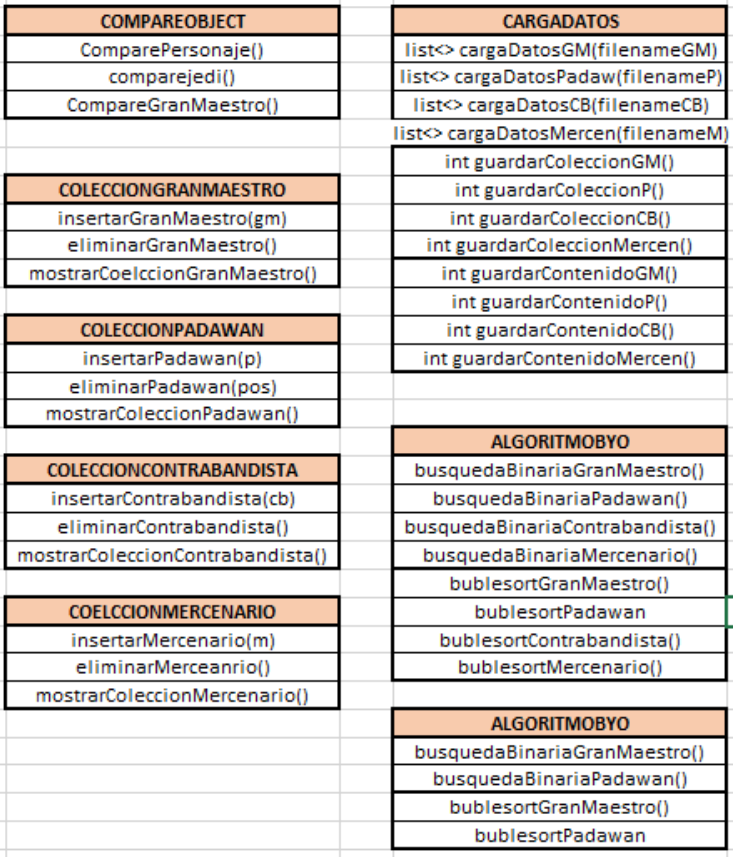
Esta clase tiene dos funciones. La primera es buscar un personaje cualquiera y la segunda ordenar a los personajes según un parámetro. La clase tiene ocho métodos, cuatro de búsqueda binaria para cada clase y otros cuatro de bublesort de cada clase, se definen:

- **bool busquedaBinariaGranMaestro(list<GranMaestro> arra, int numM):** busca un personaje GranMaestro de la lista creada anteriormente de dicho personaje.
- **void bubblesortGranMaestro(list<GranMaestro> *ara):** ordena los personajes GranMaestro de la lista creada anteriormente de dicho personaje, según el parámetro de entrada que tú quieras.
- **bool busquedaBinariaPadawan(vector<Padawan> arra, int numM):** busca un personaje Padawan de la lista creada anteriormente de dicho personaje.
- **void bubblesortPadawan(vector<Padawan> ara):** ordena los personajes Padawan de la lista creada anteriormente de dicho personaje, según el parámetro de entrada que tú quieras.
- **bool busquedaBinariaContrabandista(stack<Contrabandista> arra, int numM):** busca un personaje Contrabandista de la lista creada anteriormente de dicho personaje.
- **void bubblesortContrabandista(stack<Contrabandista> ara):** ordena los personajes Contrabandista de la lista creada anteriormente de dicho personaje, según el parámetro de entrada que tú quieras.
- **bool busquedaBinariaMercenario(queue<Mercenario> arra, int numM):** busca un personaje Mercenario de la lista creada anteriormente de dicho personaje.
- **void bubblesortMercenario(queue<Mercenario> ara):** ordena los personajes Mercenario de la lista creada anteriormente de dicho personaje, según el parámetro de entrada que tú quieras.
- Un **ejemplo de búsqueda binaria** sería: queremos encontrar al personaje que tenga 2 Maestros, para ello creamos un vector con los personajes existentes y pasamos por parámetro el número “dos” para que busque que personaje tiene 2 Maestros.
- Un **ejemplo bublesort** sería: queremos ordenar a los personajes según su experiencia pues para ello utilizamos el mismo vector que creamos anteriormente para la búsqueda de personajes y se lo pasamos por parámetro, este se encargara de ordenar a los Padawanes que haya en el vector según su experiencia.

ALGORITMOBYO
busquedaBinariaGranMaestro()
busquedaBinariaPadawan()
busquedaBinariaContrabandista()
busquedaBinariaMercenario()
bublesortGranMaestro()
bublesortPadawan()
bublesortContrabandista()
bublesortMercenario()

Diagrama de los personajes





DESCRIPCIÓN DE ARCHIVOS

En este apartado del trabajo describiremos de los archivos E/S y explicaremos su estructura.

Un personaje viene definido por un nombre (tipo *string*), apellido (tipo *string*), raza (tipo *string*), bando (tipo *string*), planeta de nacimiento (tipo *string*) y su experiencia (tipo *int*). Esta última característica esta entre el valor 0 y 100, siendo 0 el porcentaje de menor valor y 100 significa que posee la característica completamente. Para cada clase hija y nieta tendrá nuevas características y distintas características.

- La hija **Jedi** tiene las características: la conexión con la fuerza (tipo *string*) y los midiclorianos (tipo *int*).
  - La nieta **GranMaestro** tiene las características: habilidad (tipo *string*), número de aprendices (tipo *string*) y nombre del aprendiz más relevante (tipo *string*).
  - La nieta **Padawan** tiene las características: número de maestros (tipo *string*) y nombre del maestro más relevante (tipo *string*).
- La hija **SinFuerza** tiene las características: gremio (tipo *string*) e idioma (tipo *string*).
  - La nieta **Contrabandista** tiene las características: arma (tipo *string*), nave (tipo *string*) y contratos (tipo *int*).
  - La nieta **Mercenario** tiene las características: arma (tipo *string*), jefe (tipo *string*) y asesinatos (tipo *int*).

Todas las propiedades de los personajes se pueden consultar y modificar en cualquier momento. Un personaje definido por defecto tiene las propiedades de string con el valor cadena vacía (“”) y las de enteros a un valor -1 (valor no aceptable). Además, un personaje se puede crear a partir de todas sus propiedades. Tiene un método que guarda una cadena con toda la información del personaje en el formato:

- **GranMaestro:**
  “Nombre;Apellido;Raza;Bando;PlanetaDeNacimiento;Experiencia;Habilidad;ConexionFuerza;Midiclorianos;NúmeroDeAprendices;NombreDelAprendizMásRelevante”
- **Padawan:**
  “Nombre;Apellido;Raza;Bando;PlanetaDeNacimiento;Experiencia;ConexiónFuerza;MidiclorianosNúmeroDeMaestros;NombreDelMaestroMásRelevante”
- **Contrabandista:**
  “Nombre;Apellido;Raza;Bando;PlanetaDeNacimiento;Experiencia;Gremio;Idioma;Arma;Nave;Contratos”
- **Mercenario:**
  “Nombre;Apellido;Raza;Bando;PlanetaDeNacimiento;Experiencia;Gremio;Idioma;Habilidad;Arma;Jefe;Asesinatos”

(El separador de los campos es el carácter “;”). Cada clase tiene un txt con su nombre que proporciona las características de cada personaje que se hayan creado.