# Que

# Partint de la Practica anterior, corregint les errades que hi hagen, en coneixeries.

# afix la capacitat d'emmagatzemament en fitxer

# afix la capacitat de carregar en fitxer

# En aquesta practica es farà la lectura i escriptura en bloc

# Per a que

Per a practicar l'ús de fitxers en Java i per a practicar la gestió de dades.

# Com

# En primer lloc, començaré dient que les meues classes generals "Colors" i "ReadClient" són iguals que en la pràctica anterior, però estan en un altre paquet.

# La classe "Alumne" té dues variables: "nom" i "nia" (totes dues cadenes de text), i després té un mètode "comprovar dades" que comprova amb una coincidència (match) si la cadena que es demana com a paràmetre compleix un dels dos requisits perquè siga vàlida. Les funcions "demane el nom" i "demane el nia" serveixen per a demanar el nom o el nia per pantalla i fer ús de la funció "comprovar dades" per a saber si el que es demana per pantalla compleix els requisits; si no, es torna a demanar el nom o el nia. Les funcions "toString" i "fromString" retornen les dades de l'objecte segons un format.

# La classe "Mòdul" té com a variables "nom" i "id", amb dos constructors, un que demana tant el nom com el id i l'altre que sols demana el nom i genera un id automàtic amb la funció "generar Id" de la pròpia classe. També té les funcions "demane id" que demana un id i comprova que segueix un format amb una coincidència (match) perquè sols puguen ser 8 números. Per últim, la funció "fromString" que retorna les dades seguint un format.

# La meua classe "Matrícula" és molt similar a la de la pràctica anterior, l'única diferència és que, a més de l'ArrayList de notes i la mitjana, també guarda el nia i l'id del mòdul, i també he afegit el mètode "fromString".

# La meua classe "Alumnes" també és prou similar a la pràctica 2; les úniques diferències més destacades són que ara la meua funció per a donar d'alta un alumne té com a paràmetres d'entrada el nom i el nia perquè siga més fàcil la càrrega d'alumnes, i també he afegit el mètode "fromString".

# En la meua funció "Mòduls" he afegit un altre tipus de funció "alta" per a que cree automàticament un id del mòdul i el mètode "fromString".

# La funció "Matrícules" té algunes noves funcions com "dades," que comprova si les dades són correctes i si no, les torna a demanar. La funció "enllaçar Matrícula" que demana un nia i un id i retorna una matrícula de la llista de matrícules; una nova funció "qualificar" que té com a paràmetre d'entrada la matrícula i un array de notes, i el mètode "fromString".

# En la meua classe principal he afegit unes variables finals, que són les rutes dels fitxers i tres funcions, dues d'elles s'encarreguen de carregar les dades i una altra escriu en els fitxers. La funció "carregar()" té com a paràmetre d'entrada una ruta al fitxer d'on vols carregar i un booleà que li indica en l'ArrayList on vols carregar. "Carregar()" llig l'única línia del fitxer, la separa amb ";" i la guarda en un array que torna a separar en "," i segons el booleà, carrega en l'ArrayList de mòduls o el d'alumnes. "Carregar Matrícules()" fa el mateix però aplicat al format de les matrícules ja que les notes estan separades per espais i el nia i l'id del mòdul en "," i entre matrícules ";".

# El mètode "escriure()" demana una cadena del contingut nou i antic i actualitzat i el fitxer on vols guardar-ho, per a escriure-ho.

# A l'inici de la classe principal crida a les dues funcions "carregar" i al final crida al mètode "escriure" introduint-li la funció de l'objecte que vols guardar, "fromString" i la ruta on ho vols guardar.

# Pseudocòdic

**Clase Alumne:**

Atributos:

- nom (cadena de texto)

- nia (cadena de texto)

Métodos:

- Constructor Alumne(nom, nia)

- Asignar el valor de nom al atributo nom de la instancia

- Asignar el valor de nia al atributo nia de la instancia

- Método estático comprobarDatos(str, option, msgErr)

- Parámetros:

- str (cadena de texto)

- option (booleano)

- msgErr (cadena de texto)

- Retorno: booleano

- Si option es verdadero:

- Definir matches para nombre

- Sino:

- Definir matches para nia

- Si str coincide con matches:

- Retornar verdadero

- Sino:

- Mostrar un mensaje de error msgErr

- Retornar falso

- Método estático pedirName()

- Retorno: cadena de texto

- Hacer

- Pedir nombre i lo guarda en name

- Llama a comprobar datos

- Retornar name

- Método estático pedirNia()

- Retorno: cadena de texto

- Hacer

- Pedir nia i lo guarda en nia

- Llama a comprobar datos

- Retornar nia

- Método toString()

- Retornar nom, " - ", nia

- Método fromString()

- Retornar nom, ",", nia

**Clase Modul:**

Atributos:

- nom (cadena de texto)

- id (cadena de texto)

Constructor Modul(nom)

- Parámetro: nom (cadena de texto)

- Asignar el valor de nom al atributo nom de la instancia

- Llamar al método generarId() y asignar el resultado al atributo id

Constructor Modul(nom, id)

- Parámetros:

- nom (cadena de texto)

- id (cadena de texto)

- Asignar el valor de nom al atributo nom de la instancia

- Asignar el valor de id al atributo id de la instancia

Método estático pedirId()

- Crear una instancia de ReadClient llamada rc

- Inicializar id como cadena de texto

- Inicializar matches como "\\d{8}" (expresión regular para 8 dígitos)

- Inicializar ok como falso

- Hacer

- Solicitar al usuario el ID del módulo

- Leer el ID ingresado y asignarlo a la variable id

- Si id coincide con la expresión regular matches

- Asignar verdadero a ok

- Sino

- Mostrar un mensaje de error "Formato del ID incorrecto"

- Mientras ok sea falso

- Retornar id

Método generarId()

- Definir un rango de valores mínimos y máximos (min y max) para el ID

- Generar un número aleatorio dentro del rango

- Convertir el número aleatorio en cadena de texto y retornarlo

Método fromString()

- Crear una cadena de texto obj que contenga el nombre y el ID del módulo separados por una coma

- Retornar obj

**Clase Matricula:**

Atributos:

- notes (ArrayList de números decimales)

- mitjana (número decimal)

- idModul (cadena de texto)

- nia (cadena de texto)

Constructor Matricula(nia, idModul)

- Parámetros:

- nia (cadena de texto)

- idModul (cadena de texto)

- Asignar el valor de idModul al atributo idModul de la instancia

- Asignar el valor de nia al atributo nia de la instancia

- Inicializar la lista de notes como una lista vacía

Método addNota(nota...)

- Parámetros: lista variable de números decimales (nota)

- Para cada nota en nota

- Agregar la nota a la lista de notes

- Llamar al método updateMitjana()

Método setNota(nota, pos)

- Parámetros:

- nota (número decimal)

- pos (entero)

- Establecer la nota en la posición pos de la lista de notes como la nota proporcionada

- Mostrar un mensaje de éxito "La nota se ha modificado"

- Llamar al método updateMitjana()

Método delNota(pos)

- Parámetros:

- pos (entero)

- Eliminar la nota en la posición pos de la lista de notes

- Llamar al método updateMitjana()

Método mostrarNotes(lista)

- Parámetro:

- lista (booleano)

- Si la lista de notes no está vacía

- Si lista es verdadero

- Para cada nota en notes con índice i

- Mostrar el índice i + 1 y el valor de la nota con formato "%.2f"

- Sino

- Para cada nota en notes con índice i

- Mostrar el valor de la nota con formato "%.2f" y un espacio en blanco

- Mostrar la mitjana con formato en púrpura y blanco

- Retornar 0

- Sino

- Retornar -1

- Mostrar un mensaje de advertencia "Este alumno no tiene notas"

Método updateMitjana()

- Inicializar sumatorio como 0

- Para cada nota en notes

- Sumar la nota a sumatorio

- Calcular la mitjana como sumatorio dividido por el número de notas en notes

- Asignar el valor de mitjana al atributo mitjana de la instancia

- Mostrar un mensaje de éxito "La mitjana se ha actualizado"

Método fromString()

- Inicializar obj como cadena de texto que contiene nia, idModul y una lista de notas

- Para cada nota en notes

- Agregar la nota a obj con un espacio en blanco

- Retornar obj

**Clase Alumnes:**

Atributos:

- list (lista de objetos de tipo Alumne)

- rc (objeto de tipo ReadClient)

Método menu():

Repetir mientras repetir sea verdadero:

Mostrar el menú de Alumnes:

(0) Salir

(1) Alta

(2) Baja

(3) Lista

Leer la opción del usuario y almacenarla en menu

Según el valor de menu:

Caso 0:

Mostrar "Has salido del menú Alumnes"

Establecer repetir como falso

Caso 1:

Llamar al método alta()

Caso 2:

Llamar al método baixa()

Caso 3:

Llamar al método mostrarLista()

De lo contrario:

Mostrar un mensaje de error indicando que se debe introducir un valor válido

Método mostrarLista():

Para cada alumno en la lista:

Mostrar el índice + 1, nombre y NIA del alumno

Método alta():

Limpiar el buffer de entrada

Leer el nombre del alumno y almacenarlo en nom

Leer el NIA del alumno y almacenarlo en nia

Si buscarNia(nia) es igual a -1:

Agregar un nuevo objeto Alumne a la lista con el nombre y NIA proporcionados

Mostrar un mensaje de éxito

De lo contrario:

Mostrar un mensaje de error indicando que el alumno ya existe

Método baixa():

Limpiar el buffer de entrada

Leer el NIA del alumno y almacenarlo en nia

Obtener la posición del alumno en la lista usando buscarNia(nia)

Si la posición es igual a -1:

Mostrar un mensaje de error indicando que el alumno no existe

De lo contrario:

Eliminar el alumno de la lista

Mostrar un mensaje de éxito

Método buscarNia(nia):

Inicializar retorno como -1

Para cada alumno en la lista:

Si el NIA del alumno es igual a nia:

Establecer retorno como la posición del alumno en la lista

Devolver retorno

Método fromString(list: ArrayList de Matricula): String

objs = ""

Para 'i' desde 0 hasta el tamaño de 'list':

objs += list.get(i).fromString() + ";"

Devolver 'objs' como una cadena de texto

**Clase Moduls:**

Atributos:

- list (lista de objetos de tipo Modul)

- rc (objeto de tipo ReadClient)

Método menu():

Repetir mientras repetir sea verdadero:

Mostrar el menú de Modul:

(0) Salir

(1) Alta

(2) Baja

(3) Lista

(4) Matricular Alumno

Leer la opción del usuario y almacenarla en menu

Según el valor de menu:

Caso 0:

Establecer repetir como falso

Mostrar "Has salido del menú Moduls"

Caso 1:

Llamar al método alta()

Caso 2:

Llamar al método baja()

Caso 3:

Llamar al método mostrarLista()

Caso 4:

Llamar al método matricularAlumne()

De lo contrario:

Mostrar un mensaje de advertencia indicando que se debe introducir un valor válido

Método mostrarLista():

Si la lista de módulos está vacía:

Mostrar un mensaje de advertencia indicando que no hay módulos

De lo contrario:

Para cada módulo en la lista de módulos:

Mostrar el índice + 1 y el nombre del módulo

Método alta():

Limpiar el buffer de entrada

Leer el nombre del módulo desde la entrada del usuario y almacenarlo en nom

Si buscarModul(nom) es igual a -1:

Agregar un nuevo objeto Modul a la lista con el nombre proporcionado

Mostrar un mensaje de éxito

De lo contrario:

Mostrar un mensaje de error indicando que el módulo ya existe

Método baja():

Limpiar el buffer de entrada

Leer el nombre del módulo desde la entrada del usuario y almacenarlo en modul

Obtener la posición del módulo en la lista de módulos usando buscarModul(modul)

Si la posición es diferente de -1:

Eliminar el módulo de la lista de módulos

Mostrar un mensaje de éxito

Llamar al método desmatricularAlumnes(modul) para eliminar a los alumnos matriculados en ese módulo

Mostrar un mensaje de éxito indicando que los alumnos se han actualizado

De lo contrario:

Mostrar un mensaje de error indicando que el módulo no se ha encontrado

Método matricularAlumne():

Limpiar el buffer de entrada

Leer el NIA del alumno desde la entrada del usuario y almacenarlo en nia

Leer el nombre del módulo desde la entrada del usuario y almacenarlo en modul

Obtener la posición del módulo en la lista de módulos usando buscarModul(modul)

Obtener la posición del alumno en la lista de alumnos usando buscarNia(nia)

Si el módulo y el alumno existen:

Obtener el objeto Alumne correspondiente al alumno

Si el módulo no existe en la lista de módulos del alumno:

Crear un nuevo objeto Modul con el nombre proporcionado

Agregar el módulo a la lista de módulos del alumno

Mostrar un mensaje de éxito

De lo contrario:

Mostrar un mensaje de error indicando que el alumno ya está matriculado en ese módulo

De lo contrario:

Mostrar un mensaje de error indicando que el módulo o el alumno no se han encontrado

Método desmatricularAlumnes(idModul: String)

matrList = Obtener la lista de matrículas (practica\_03.matriculasList.list)

matrSize = Tamaño de matrList

i = 0

Mientras i sea menor que matrSize:

matr = Obtener la matrícula en la posición i de matrList

Si matr.idMdoul es igual a idModul:

Remover matr de matrList

Decrementar matrSize en 1

Incrementar i en 1

Método buscarModul(idModul: String): entero

retorno = -1

Para 'i' desde 0 hasta el tamaño de la lista (list):

Si el 'id' del objeto en la posición 'i' de list es igual a idModul:

Asignar 'i' a retorno

Devolver 'retorno'

Método fromString(): String

objs = ""

Para 'i' desde 0 hasta el tamaño de la lista (list):

objs += Obtener la representación en cadena del objeto en la posición 'i' de list usando 'fromString()' + ";"

Devolver 'objs' como una cadena de texto

**Clase Matriculas:**

Atributos:

- rc (instancia de ReadClient)

- list (ArrayList de Matricula)

Método menu()

- Variables locales:

- menu (entero)

- repetir (booleano)

- matr (instancia de Matricula)

- Inicializar repetir como verdadero

- Mientras repetir sea verdadero

- Mostrar el menú Avaluar con opciones (0) Salir, (1) Qualificar, (2) Modificar, (3) Traure bolletí de notes

- Leer la opción del menú y asignarla a la variable menu

- Según el valor de menu:

- Caso 0:

- Asignar falso a repetir

- Mostrar "Has salido del menú Avaluar"

- Caso 1:

- Llamar al método dades() y asignar el resultado a matr

- Si matr no es nulo, llamar al método qualificar(matr)

- Caso 2:

- Llamar al método dades() y asignar el resultado a matr

- Si matr no es nulo, llamar al método modificar(matr)

- Caso 3:

- Llamar al método mostrar()

- Por defecto:

- Mostrar un mensaje de advertencia "Debes introducir un valor válido"

Método dades()

- Variables locales:

- matr (instancia de Matricula)

- Si la lista de matrículas en practica\_03.matriculasList no está vacía

- Inicializar matr como nulo

- Inicializar ok como falso

- Hacer

- Solicitar al usuario el NIA de un alumno

- Solicitar al usuario el ID de un módulo

- Llamar al método enlazarMatricula(nia, idModul) y asignar el resultado a matr

- Si matr no es nulo

- Asignar verdadero a ok

- Mostrar un mensaje de éxito "Datos correctos"

- Sino

- Mostrar un mensaje de error "Datos incorrectos"

- Mientras ok sea falso

- Sino

- Mostrar un mensaje de advertencia "No hay matrículas registradas"

- Retornar matr

Método enlazarMatricula(nia, idModul)

- Parámetros:

- nia (cadena de texto)

- idModul (cadena de texto)

- Inicializar matr como nulo

- Obtener la lista de matrículas en practica\_03.matriculasList.list

- Para cada matrícula en la lista

- Obtener el idModul y nia de la matrícula

- Si idModul es igual a idModul proporcionado y nia es igual a nia proporcionado

- Asignar la matrícula a matr

- Retornar matr

Método qualificar(matr)

- Parámetro: matr (instancia de Matricula)

- Si matr no es nulo

- Solicitar al usuario la cantidad de notas a agregar

- Para i desde 0 hasta la cantidad

- Solicitar al usuario una nota dentro del rango de 0.0 a 10.0

- Llamar al método addNota(nota) en matr

- Mostrar un mensaje de éxito "La nota se ha agregado"

Método modificar(matr)

- Parámetro: matr (instancia de Matricula)

- Si la lista de notas en matr no está vacía

- Solicitar al usuario la posición de una nota para modificar

- Solicitar al usuario una nueva nota dentro del rango de 0.0 a 10.0

- Llamar al método setNota(nota, pos) en matr

Método mostrar()

- Inicializar mostrar como cadena de texto vacía

- Obtener la lista de alumnos en practica\_03.alumnesList.list

- Para cada alumno en la lista

- Obtener el nombre y el NIA del alumno

- Agregar el nombre y el NIA a mostrar

- Inicializar nia como el NIA del alumno

- Para cada matrícula en la lista

- Si el NIA de la matrícula es igual a nia

- Agregar la representación en cadena de la matrícula a mostrar

- Mostrar mostrar

Método fromString()

- Inicializar objs como cadena de texto vacía

- Para cada matrícula en la lista

- Agregar la representación en cadena de la matrícula a objs seguida de un punto y coma (;)

- Retornar objs

**Clase Practica\_03:**

Atributos estáticos:

- alumnesList (instancia de Alumnes)

- ALUMNES\_PATH (cadena de texto que contiene la ruta al archivo de alumnos)

- modulsList (instancia de Moduls)

- MODULS\_PATH (cadena de texto que contiene la ruta al archivo de módulos)

- matriculasList (instancia de Matriculas)

- MATRICULES\_PATH (cadena de texto que contiene la ruta al archivo de matrículas)

Método principal main(args)

- Cargar datos de alumnos y módulos desde archivos llamando al método carregar

- Crear una instancia de ReadClient llamada rc

- Inicializar repit como verdadero

- Mientras repit sea verdadero

- Mostrar el menú principal con opciones (0) Salir, (1) Menu Alumnes, (2) Menu Mòdul, (3) Avaluar

- Leer la opción del menú y asignarla a la variable menu

- Según el valor de menu:

- Caso 0:

- Asignar falso a repit

- Mostrar "Has salido del menú Mòdul"

- Caso 1:

- Llamar al método menu de Alumnes

- Caso 2:

- Llamar al método menu de Moduls

- Caso 3:

- Llamar al método menu de Matriculas

- Por defecto:

- Mostrar un mensaje de advertencia "Debes introducir un valor válido"

- Escribir datos de alumnos, módulos y matrículas en archivos llamando al método escribir

Método carregar(path, modo)

- Leer datos desde un archivo de texto en la ruta path

- Para cada línea en el archivo

- Dividir la línea en entidades separadas por punto y coma (;)

- Para cada entidad en entidades

- Dividir la entidad en campos separados por coma (,)

- Si modo es verdadero (carga de alumnos)

- Llamar al método alta de Alumnes con el nombre y el NIA del alumno

- Sino (carga de módulos)

- Llamar al método alta de Moduls con el nombre y el ID del módulo

Método carregarMatricules()

- Leer datos desde un archivo de texto de matrículas

- Para cada línea en el archivo

- Dividir la línea en entidades separadas por punto y coma (;)

- Para cada entidad en entidades

- Dividir la entidad en campos separados por coma (,)

- Obtener el NIA, el ID del módulo y las notas

- Verificar si el NIA y el ID del módulo existen en Alumnes y Moduls

- Si existen

- Llamar al método matricularAlumne de Moduls

- Dividir las notas en valores individuales

- Para cada valor en notas

- Intentar convertir el valor a un número decimal y agregarlo a la matrícula

- Mostrar un mensaje de éxito

- Sino

- Mostrar un mensaje de error

Método escribir(list, path)

- Crear un archivo en la ruta path

- Crear un escritor FileWriter

- Escribir la lista list en el archivo

# Conclusió

En conclusió, aquesta pràctica m'ha sigut de gran utilitat per repassar àmpliament la programació de fitxers en Java. Aquesta pràctica m'ha costat molt de temps de fer degut a que he hagut de canviar les relacions de les meues Matricules, Mòduls i Alumnes.