Que

Modifica el programa anterior para no haver de carregar totes les dades a memòria

Per a que

Per aprendre com treballar en gestionar la memòria i fer el repte un poc més complicat

Com

En primer lloc, les meues classes Colors i ReadClient no s'han modificat, per tant passaré a explicar les altres classes. Començaré explicant la classe Mantenible. Aquesta classe serveix com a pare per a les classes Mòdul i Alumne, només té un constructor amb els paràmetres nom i id, i les funcions són iguals que les funcions de la classe Alumne de la pràctica anterior.

Les classes Alumne i Mòdul hereten de Mantenible i només tenen un constructor que crida al constructor de Mantenible.

La classe Matrícula és igual que a la pràctica anterior.

La classe Mantenibles és la classe pare d'Alumnes, Mòduls i Matrícules. Té unes 5 funcions: El mètode llegirFitxer llig un fitxer demanat amb el seu path com a paràmetre d'entrada i retorna un ArrayList de tipus Mantenible.

El mètode escriure demana el text a escriure i el path per apuntar al fitxer. buscarLlista crida a llegirFitxer i guarda l'ArrayList per a buscar el text demanat com a paràmetre i retorna la posició.

mostrarLlista mostra una llista dels objectes Mantenibles d'un fitxer.

I l'últim mètode és el fromString que demana com a paràmetre un ArrayList de Mantenible i el retorna com a cadena amb un format especial per a ser guardat en fitxers.

La classe Alumnes té el menú similar al de la pràctica anterior amb unes lleugeres modificacions per adaptar-se a les funcions de la seua classe pare. També té les funcions alta() que té com a paràmetres d'entrada un ArrayList on s'han de fer les comprovacions i afegir els alumnes, el nom i l'id de l'alumne i el path on escriure després les modificacions. En acabar la funció crida a escriure de la classe pare per pujar els canvis al fitxer.

La funció baixa() fa el mateix que en alumnes, però també elimina el mòdul de tots els alumnes.

matricularAlumne i desmatricularAlumnes són pràcticament iguals que en la pràctica anterior, però demanant un ArrayList a modificar.

La classe Mòduls, passa el mateix que en la classe Alumnes, alta() pot pujar diversos mòduls a la vegada a més de cridar la funció generarId() que genera un id per al mòdul.

La funció baixa és la mateixa que en alumne, però també elimina el mòdul de tots els alumnes. matricularAlumne i desmatricularAlumnes són pràcticament iguals que en la pràctica anterior, però demanant un ArrayList a modificar. La classe Matrícules també ha canviat una mica, en totes les funcions demana un ArrayList a modificar.

Les funcions dades(), enllaçarMatrícula(), qualificar(), modificar() i mostrar() són iguals, eccepte per el detalls de l'ArrayList. Les dues funcions afegides són llegirFitxer() llig el fitxer de matrícules i retorna un ArrayList d'aquestes i paraString() és la funció que retorna en cadena els objectes matrícula per a poder ser emmagatzemats. La meua classe principal és igual que en la pràctica anterior, però només amb el mètode menú.

Pseudocòdic

Clase Mantenible:

```
    Atributos:
        - nom (String)
        - id (String)
        - rc (Instancia de ReadClient, declarada como estática)

    Constructor Mantenible(nom: String, id: String):
        Asignar nom a this.nom
        Asignar id a this.id

    Método estático pedirNom(): // Solicita un nombre
        nom = ""
        Hacer:
            Leer el nombre desde el usuario y asignarlo a 'nom'
        Mientras comprabarDatos(nom, Verdadero, "El nom no puede contener números")
sea Falso
        Devolver nom

    Método estático pedirId(): // Solicita una identificación (ID)
        id = ""
        Hacer:
            Leer la identificación (ID) desde el usuario y asignarla a 'id'
        Mientras comprabarDatos(id, Falso, "El ID debe contener exactamente 8
números") sea Falso
        Devolver id

    Método estático comprabarDatos(str: String, option: Boolean, msgErr: String): //
Comprueba si una cadena cumple con ciertos criterios
        correcto = Falso
        matches = ""
        Si option es Verdadero:
            matches = "\\D*" // Acepta cualquier cosa que no sea un número
        Sino:
            matches = "\\d{8}" // Debe contener exactamente 8 dígitos numéricos

        Si str coincide con matches:
            correcto = Verdadero
        Sino:
            Mostrar mensaje de advertencia 'msgErr'
            correcto = Falso
        Devolver correcto

    Método toString():
        Devolver la representación de la instancia como una cadena en el formato
"nombre - ID"

    Método fromString():
        Crear una cadena 'obj' que contiene el nombre y la ID separados por una coma
        Devolver 'obj'
```

Clase Alumne extiende Mantenible:

```
    Atributos:
        - nom (String)
        - id (String)

    Constructor Alumne(nom: String, id: String):
        Llamar al constructor de Mantenible con los argumentos nom y id
```

Clase Modul extiende Mantenible:

```
    Atributos:
        - nom (String)
        - id (String)

    Constructor Modul(nom: String, id: String):
        Llamar al constructor de Mantenible con los argumentos nom y id
```

Clase Matricula:

```
    Atributos:
        - notes (ArrayList de Double)
        - mitjana (Double)
        - idMdoul (String)
        - nia (String)

    Constructor Matricula(nia: String, idMdoul: String):
        Asignar idMdoul a this.idMdoul
        Asignar nia a this.nia
        Inicializar notes como un nuevo ArrayList de Double

    Método addNota(nota: Double...):
        Para cada nota en nota:
            Agregar nota a la lista notes
        Llamar a updateMitjana para actualizar la mitjana

    Método addNota(nota: Double):
        Agregar nota a la lista notes
        Llamar a updateMitjana para actualizar la mitjana

    Método setNota(nota: Double, pos: Entero):
        Establecer la nota en la posición pos en la lista notes como nota
        Mostrar mensaje de éxito
        Llamar a updateMitjana para actualizar la mitjana

    Método delNota(pos: Entero):
        Eliminar la nota en la posición pos de la lista notes
        Llamar a updateMitjana para actualizar la mitjana

    Método toString():
        Obtener la posición del módulo en la lista de módulos usando idMdoul
        Obtener el nombre del módulo
        Crear una cadena 'matricula' que incluye el nombre del módulo y las notas
        Devolver 'matricula' con la mitjana destacada en púrpura

    Método mostrarNotes(lista: Boolean):
        Si la lista de notas no está vacía:
            Si lista es verdadero:
                Para cada nota en notes:
                    Mostrar el índice y la nota
            Si no:
                Para cada nota en notes:
                    Mostrar la nota seguida de un guión y espacio
                Mostrar la mitjana en púrpura
            Devolver 0
        De lo contrario:
            Mostrar un mensaje de advertencia
            Devolver -1

    Método updateMitjana():
        Calcular la suma de todas las notas en la lista notes y dividirla por la
cantidad de notas para obtener la mitjana
        Mostrar un mensaje de éxito

    Método fromString():
        Crear una cadena 'obj' que contiene nia, idMdoul y las notas
```

```
Método fromString():
    obj = nia,el id del modulo,
    i = 0
    Mientras i sea menor que el tamaño de la lista 'notes':
        obj += notes.get(i) + " "
        Incrementar i en 1
    Devolver 'obj' como una cadena de texto
        Devolver 'obj'
```

Clase Mantenibles:

```
Clase abstracta Mantenibles:
    Métodos:
        - leerFicher(path: String): ArrayList de Mantenible
            list = Lista vacía de Mantenible
            fl = Archivo
            sc = Scanner
            Intentar:
                fl = Crear un objeto Archivo con la ruta 'path'
                sc = Crear un objeto Scanner para leer el archivo
                Mientras sc tiene líneas:
                    Leer una línea 'ln' del archivo
                    Dividir 'ln' en 'entidades' utilizando ';'
                    Para cada 'entidadString' en 'entidades':
                        Dividir 'entidadString' en 'entidad' utilizando ','
                        Crear un nuevo objeto Mantenible con entidad[0] y entidad[1]
y agregarlo a 'list'
                Fin Mientras
                Cerrar sc
            Capturar FileNotFoundException:
                Mostrar mensaje de error "Fallo en el archivo"
            Capturar Exception:
                Mostrar mensaje de error "Algo ha salido mal"
            Finalmente:
                Limpiar fl y sc
            Devolver 'list'

        - escribir(str: String, path: String)
            fl = Archivo
            fw = FileWriter
            Intentar:
                fl = Crear un objeto Archivo con la ruta 'path'
                fw = Crear un objeto FileWriter con el archivo, modo de escritura y
sin apéndice
                Escribir 'str' en el archivo
                Cerrar fw
            Capturar IOException:
                Mostrar mensaje de error "Algo ha salido mal en el archivo"
            Capturar Exception:
                Mostrar mensaje de error "Algo ha salido mal"
            Finalmente:
                Limpiar fl y fw

        - buscarList(str: String, path: String, modo: Boolean): Entero
            list = leerFicher(path)
            retorno = -1
            Para cada índice 'i' y objeto 'm' en 'list':
                Si 'modo' es Verdadero:
                    Asignar 'buscar' como 'm.id'
                Sino:
                    Asignar 'buscar' como 'm.nom'
                Si 'buscar' es igual a 'str':
                    Asignar 'i' a 'retorno'
            Devolver 'retorno'

        - mostrarLista(path: String)
            list = leerFicher(path)
            Para cada índice 'i' y objeto 'm' en 'list':
```

```
                Mostrar 'i + 1' y 'm.toString()'

        - fromString(list: ArrayList de Mantenible): String
            objs = ""
            Para cada índice 'i' en 'list':
                Concatenar 'list.get(i).fromString()' y ';' a 'objs'
            Devolver 'objs'

    Métodos abstractos (a ser implementados por las subclases):
        - fromString(): String
```

Clase Alumnes extiende Mantenibles:

```
    Atributos:
        - rc (Instancia de ReadClient)

    Método menu():
        menu = 0
        repetir = Verdadero
        Mientras repetir sea Verdadero:
            Mostrar el menú:
                - (0) Salir
                - (1) Alta
                - (2) Baixa
                - (3) Llista
            Leer la selección del usuario en 'menu'
            Según 'menu':
                Caso 0:
                    Mostrar "Has salido del menú Alumnes"
                    repetir = Falso
                Caso 1:
                    Pedir el nombre y la ID del alumno
                    Llamar al método 'alta' con el nombre, ID, lista y ruta de
almacenamiento
                Caso 2:
                    Llamar al método 'baixa' con la ruta de almacenamiento de alumnos
                Caso 3:
                    Llamar a 'mostrarLista' con la ruta de almacenamiento de alumnos
                Predeterminado:
                    Mostrar un mensaje de advertencia

    Método alta(nom: String, id: String, list: ArrayList de Mantenible, path:
String):
        Si buscarList(id, path, Verdadero) devuelve -1:
            Agregar un nuevo objeto 'Alumne' con el nombre y la ID a la lista 'list'
            Mostrar un mensaje de éxito con el nombre y la ID
        De lo contrario:
            Mostrar un mensaje de advertencia
        Escribir 'list' en el archivo en la ruta 'path'
        Devolver el tamaño de la lista menos 1

    Método baixa(path: String):
        Obtener la lista de alumnos desde el archivo en la ruta 'path'
        Buscar la posición de la ID ingresada en la lista
        Si la posición es -1:
            Mostrar un mensaje de error
        De lo contrario:
            Eliminar el alumno de la lista en la posición encontrada
            Mostrar un mensaje de éxito
        Escribir la lista actualizada en el archivo en la ruta 'path'
```

Clase Moduls extiende Mantenibles:

```
    Atributos:
        - rc (Instancia de ReadClient)

    Método menu():
```

```
        menu = 0
        repetir = Verdadero
        Mientras repetir sea Verdadero:
            Mostrar el menú:
                - (0) Salir
                - (1) Alta
                - (2) Baixa
                - (3) Llista
                - (4) Matricular Alumne
            Leer la selección del usuario en 'menu'
            Según 'menu':
                Caso 0:
                    repetir = Falso
                    Mostrar "Has salido del menú Mòdul"
                Caso 1:
                    nom = rc.pedirString("Nom del mòdul: ")
                    alta(nom)
                Caso 2:
                    baixa(MODULS_PATH)
                Caso 3:
                    super.mostrarLista(MODULS_PATH)
                Caso 4:
                    nia = Mantenible.pedirId()
                    idModul = Mantenible.pedirId()
                    matricularAlumne(nia, idModul, Matriculas.leerFicher())
                Predeterminado:
                    Mostrar "Debes introducir un valor válido"

    Método alta(modul: Arreglo de Strings)
        list = super.leerFicher(MODULS_PATH)
        Para cada nombre en modul:
            Si super.buscarList(nombre, MODULS_PATH, Falso) es -1:
                idAleatorio = generarId()
                list.add(new Modul(nombre, idAleatorio))
                Mostrar nombre + " se ha dado de alta"
            Sino:
                Mostrar "El módulo ya existe"
        super.escribir(super.fromString(list), MODULS_PATH)

    Método generarId(): String
        min = 10000000
        max = 99999999
        random = Generar un número aleatorio en el rango (min, max)
        Devolver el número aleatorio como una cadena de texto

    Método baixa(path: String)
        list = leerFicher(path)
        pos = buscarList(Mantenible.pedirId(), path, Verdadero)
        Si pos es -1:
            Mostrar "El módulo no existe"
        Sino:
            desmatricularAlumnes(list.get(pos).id, Matriculas.leerFicher())
            list.remove(pos)
            Mostrar "El módulo se ha dado de baja correctamente"
        Escribir fromString(list) en 'path'

    Método matricularAlumne(nia: String, idModul: String, list: ArrayList de
Matricula): Matricula
        matrs = list
        matrs.add(new Matricula(nia, idModul))
        super.escribir(Matriculas.paraString(list), MATRICULES_PATH)
        Devolver la última matrícula agregada en 'matrs'

Método desmatricularAlumnes(idModul: String, list: ArrayList de Matricula)
    matrList = list
    matrSize = Tamaño de matrList
```

```
    i = 0
    Mientras i sea menor que matrSize:
        matr = matrList.get(i)
        Si matr.idMdoul es igual a idModul:
            matrList.remove(i)
            Decrementar matrSize en 1
        Sino:
            Incrementar i en 1
```

Clase Matriculas extiende Mantenibles:

```
    Atributos:
        - rc (Instancia de ReadClient)

    Método menu():
        menu = 0
        repetir = Verdadero
        Mientras repetir sea Verdadero:
            Matricula matr
            ArrayList<Matricula> list
            Mostrar el menú:
                - (0) Salir
                - (1) Qualificar
                - (2) Modificar
                - (3) Traure bolletí de notes
            Leer la selección del usuario en 'menu'
            Según 'menu':
                Caso 0:
                    repetir = Falso
                    Mostrar "Has salido del menú Avaluar"
                Caso 1:
                    list = leerFicher()
                    matr = dades(list)
                    Si matr no es nulo:
                        qualificar(matr)
                    Escribir paraString(list) en MATRICULES_PATH
                Caso 2:
                    list = leerFicher()
                    matr = dades(list)
                    Si matr no es nulo:
                        modificar(matr)
                    Escribir paraString(list) en MATRICULES_PATH
                Caso 3:
                    mostrar()
                Predeterminado:
                    Mostrar "Debes introducir un valor válido"

    Método dades(list: ArrayList de Matricula): Matricula
        matr = nulo
        Si la lista no está vacía:
            ok = Falso
            Hacer:
                nia = Mantenible.pedirId()
                idModul = Mantenible.pedirId()
                matr = enlazarMatricula(nia, idModul, list)
                Si matr no es nulo:
                    ok = Verdadero
                    Mostrar "Datos correctos"
                Sino:
                    Mostrar "Datos incorrectos"
            Mientras ok sea Falso
        Sino:
            Mostrar "No hay matrículas registradas"
        Devolver matr

    Método enlazarMatricula(nia: String, idModul: String, list: ArrayList de
Matricula): Matricula
```

```
        matr = nulo
        Para cada matrícula en la lista:
            Si matrícula.nia es igual a nia y matrícula.idMdoul es igual a idModul:
                matr = matrícula
        Devolver matr

    Método qualificar(matr: Matricula)
        Si matr no es nulo:
            cant = rc.pedirInteger("Cuántas notas quieres agregar: ")
            Para i en rango(0, cant):
                nota = rc.pedirDouble("Nota a agregar: ", 0.0, 10.0)
                matr.addNota(nota)
                Mostrar "La nota se ha agregado"

    Método qualificar(matr: Matricula, notes: Arreglo de Double)
        matr.addNota(notes)

    Método modificar(matr: Matricula)
        Si matr.mostrarNotes(Verdadero) no es -1:
            pos = rc.pedirInteger("Cuál nota quieres modificar: ") - 1
            nota = rc.pedirDouble("Dime la nota que quieres poner: ", 0.0, 10.0)
            matr.setNota(nota, pos)

    Método mostrar()
        list = leerFicher()
        mostrar = "\n"
        alumnesList = Mantenibles.leerFicher(ALUMNES_PATH)
        Para cada índice 'i' y objeto 'alm' en 'alumnesList':
            mostrar += alm.toString() + ": \n"
            nia = alm.id
            Para cada índice 'j' y objeto 'matr' en 'list':
                Si matr.nia es igual a nia:
                    mostrar += "\t" + matr.toString() + "\n"
        Mostrar mostrar

Método leerFicher(): ArrayList de Matricula
    matrs = Nueva ArrayList de Matricula
    Intentar:
        Crear un objeto File 'fl' con la ruta MATRICULES_PATH
        Crear un objeto Scanner 'list' para leer el archivo
        Mientras 'list' tenga una siguiente línea:
            ln = Leer la siguiente línea de 'list'
            entidades = Dividir 'ln' en partes usando ";"
            Para cada 'entidadString' en 'entidades':
                entidad = Dividir 'entidadString' en partes usando ","
                nia = Obtener la primera parte de 'entidad'
                idModul = Obtener la segunda parte de 'entidad'
                niaExist = Buscar si 'nia' existe en ALUMNES_PATH
                modulExist = Buscar si 'idModul' existe en MODULS_PATH
                Si 'niaExist' y 'modulExist':
                    notas = Obtener la tercera parte de 'entidad'
                    notesArray = Dividir 'notas' en partes usando " "
                    Crear un arreglo 'notes' de dobles con el tamaño
'notesArray.length'
                    Para 'i' desde 0 hasta el tamaño de 'notesArray':
                        Intentar:
                            Convertir 'notesArray[i]' a un número decimal y
almacenarlo en 'n'
                            Asignar 'n' a 'notes[i]'
                        Capturar excepciones de conversión de tipo y mostrar mensajes
de error
                    Crear un objeto 'matr' de tipo Matricula con 'nia' e 'idModul'
                    Añadir las notas al objeto 'matr'
                    Añadir 'matr' a la lista 'matrs'
                Sino:
                    Mostrar un mensaje de error indicando que el alumno o el módulo
```

```
no existe
        Limpiar objetos 'fl' y 'list'
    Capturar excepción de FileNotFoundException y mostrar un mensaje de error
    Capturar otras excepciones generales y mostrar un mensaje de error
    Devolver 'matrs'

Método paraString(matriculas: ArrayList de Matricula): String
    list = matriculas
    objs = ""
    Para 'i' desde 0 hasta el tamaño de 'list':
        objs += Obtener la representación en cadena de 'list.get(i)' usando
'fromString()' + ";"
    Devolver 'objs' como una cadena de texto
```

Clase practica_04:

```
    Atributos:
        - alumnesList (Instancia de Alumnes)
        - ALUMNES_PATH (Ruta del archivo de lista de alumnos)
        - modulsList (Instancia de Moduls)
        - MODULS_PATH (Ruta del archivo de lista de módulos)
        - matriculasList (Instancia de Matriculas)
        - MATRICULES_PATH (Ruta del archivo de lista de matrículas)

    Método main(args: Arreglo de Strings):
        rc = Nueva instancia de ReadClient
        repit = Verdadero
        Mientras repit sea Verdadero:
            Mostrar el menú principal:
                - (0) Salir
                - (1) Menu Alumnes
                - (2) Menu Mòdul
                - (3) Avaluar
            Leer la selección del usuario en 'menu'
            Según 'menu':
                Caso 0:
                    repit = Falso
                Caso 1:
                    Llamar al método menu() de 'alumnesList'
                Caso 2:
                    Llamar al método menu() de 'modulsList'
                Caso 3:
                    Llamar al método menu() de 'matriculasList'
                Predeterminado:
                    Mostrar "Debes introducir un valor válido"
```

Conclusió

En conclusió, aquesta pràctica m'ha sigut una mica complicada, ja que he pres la decisió de crear dues classes més genèriques per a aquelles funcions que són iguals, però també m'ha resultat força cansada, ja que fer aquests canvis m'ha endarrerit molt.