Memoria Practica (II)

Accés a Dades

Andreu Sanz Sanz

Tabla de continguts

Que

La feina consisteix a crear una estructura bàsica que en permetrà, una volta finalitzat el curs, tindre una aplicació que ens permetera gestionar un centre de formació. Aquesta primera practica consistira en crear un sistema bàsic que permetrà donar de alta i matricular alumnes en mòduls, crear eixos mòduls i, una volta fet això, de poder avaluar, ficar notes a eixos mòduls (almenys 3 notes)
De moment treballarem a memòria, no escriurem a disc.

Per a que

Per a tindre la base del programa, ja que anem a treballar en aquest projecte durant tot el curs, a més a més per a veure realment quant sabem de Java, ja que és un projecte llarg i amb un cert nivell de complicació. També és útil per a aprendre a organitzar-nos per a altres projectes realment grans i aprendre a gestionar l'ús de diferents tipus de classes més genèriques.

Com

En primer lloc, començaré parlant de les meues classes generals:

Sobre la meua classe **`Colors`**, simplement es tracta d'una classe on es troben les variables per definir un color en la consola. A més, tinc tres funcions per escriure per la consola segons el tipus de missatge que siga: un error (roig), un warning (cian) o un ok (verd). El groc es representa com cian ja que en una consola amb fons blanc és més complicat llegir el warning.

La meua classe **`ReadClient`** és una classe que serveix per llegir el teclat del client i gestionar els possibles errors. En el constructor de la classe, assigna a **`sc`**, de tipus **`Scanner`**, un objecte **`Scanner`**. En general, les funcions d'aquesta classe sempre demanen una cadena, que és el missatge que li dóna a l'usuari per saber quin tipus de dada ha d'introduir. Totes elles gestionen les seues pròpies excepcions.

- **`pedirInteger`** demana un número enter i no deixa de demanar-lo fins que siga un valor vàlid, després retorna aqueix enter.
- **`pedirDouble`** demana un número, però amb la peculiaritat que té dues variables d'entrada addicionals: el màxim i el mínim. Això significa que si l'usuari introdueix un número fora d'aquest rang, se li tornarà a demanar un número. Si aquests dos valors són **`null`**, llavors no existeix aqueix rang.
- **`pedirString`** li demana una cadena a l'usuari.
- **`bufferClear`** neteja el buffer amb un 'sc.nextLine'.

Tinc una classe anomenada **Matricula**. Aquesta classe té un ArrayList de números decimals anomenat **notes** i **mitjana**, que és de tipus decimal. Aquesta classe té diferents mètodes:

- **updateMitjana** suma totes les notes de **notes** i calcula la mitjana per assignar-la a la variable **mitjana**.
- **addNota** afig una nota (número decimal que es proporciona com a paràmetre) a **notes** i crida a **updateMitjana**.
- **setNota** canvia una nota per una altra que es proporciona com a paràmetre d'entrada i crida a **updateMitjana**.
- **delNota** elimina una nota de **notes** que es proporciona com a paràmetre d'entrada i crida a **updateMitjana**.
- **mostrarNotes** mostra tot el contingut de **notes** i retorna -1 si **notes** està buit. Aquest mètode també pot rebre un booleà com a paràmetre per determinar si es vol mostrar la llista en format vertical o horitzontal.

La classe **Modul** té una variable **nom** que es sol·licita en crear l'objecte i una variable **matricula** de tipus **Matricula** que s'instancia en el constructor.

La classe **Alumne** té tres variables: **nom**, **nia** (totes dues cadenes de text) i **modulsList** (un **ArrayList** de tipus **Modul**). El seu constructor rep **nom** i **nia** com a paràmetres i assigna **modulsList** com un ArrayList. El mètode **mostrar** mostra el nom i el nia de l'alumne i les notes de cada mòdul de forma horitzontal.

La classe **Alumnes** té com a variables un ArrayList de tipus **Alumne** anomenat **llista** i un objecte de tipus **ReadClient** anomenat **rc**. Aquesta classe té diverses funcions: La funció **menú** serveix per guiar a l'usuari sobre les possibilitats que pot fer. Aquesta funció és un menú amb 4 opcions: eixir, donar d'alta, donar de baixa o veure la llista d'alumnes. L'opció un crida al mètode **alta()**, que sol·licita el nia i el nom de l'alumne i l'afegeix a l'ArrayList. La segona opció crida al mètode **baixa()**, que sol·licita el nia de l'alumne i l'elimina de l'ArrayList. La tercera opció crida a 'mostrarLlista', que mostra tots els alumnes. També hi ha una altra funció **buscarNia** que serveix per buscar el nia d'un alumne a l'ArrayList i retorna la posició de l'ArrayList o -1 si no es troba.

La classe **Mòduls** és molt similar a la classe **Alumnes**, amb la diferència que aquesta classe també pot matricular un alumne en un mòdul amb la funció **matricularAlumne**. Aquesta funció sol·licita el nia de l'alumne i el nom del mòdul per assignar un objecte de tipus mòdul a l'ArrayList de mòduls de l'alumne. També té una funció anomenada **desmatricularAlumnes()** que s'executa quan es dona de baixa un mòdul. Aquesta funció elimina el mòdul especificat pels paràmetres d'entrada en cadascun dels alumnes existents. En la funció **buscarMòdul**, hi ha dues variants: una que només sol·licita el nom del mòdul a buscar i una altra que sol·licita el mòdul i la llista en la qual buscar. S'utilitza per buscar a la llista d'un alumne en particular.

La classe **Matrícules** té només una variable, un objecte del tipus **ReadClient**. Aquesta classe també disposa d'una funció **menú()**, però en aquesta funció primer es sol·liciten les dades de l'alumne i del mòdul per a crear un objecte de tipus **Matrícula**. Això és degut a que en les 3 opcions del menú, 2 d'elles sol·liciten els mateixos valors, i en la tercera opció no se sol·licita cap valor. Això pot canviar en el futur a causa de les modificacions que vull fer. Dins del menú, tens l'opció de **qualificar**, que crida a la funció **qualificar()**. Aquesta funció sol·licita la quantitat de notes a afegir i després demana les notes per afegir-les a la matrícula de l'alumne. La segona opció és **modificar**, que modifica una nota. La tercera opció mostra les matricules de tots els alumnes.

La meua classe principal es diu **Pràctica_02** i té dues variables estàtiques: **alumnesLlista** i **mòdulsLlista**, que són objectes de les seues respectives classes. Després tenim la funció principal on instanciem un objecte de tipus **ReadClient**, un objecte de tipus **Matrícules**, i una variable boleana **repetir** per a repetir el menú. He creat un bucle **while** amb la condició de la variable **repetir**. Dins d'aquest bucle, es mostra el menú principal, es llig l'opció de l'usuari i s'assigna a la seua respectiva opció en el **switch**. Excepte en el cas que l'opció siga 0 i el programa acabe, el **switch** dirigeix l'usuari al menú d'alumnes, mòduls o matrícules. Si no es tracta de cap d'aquests casos, es mostra un missatge d'advertència i es repeteix el bucle **while**. Aquest menú és molt semblant en totes les classes.

Pseudocòdic

Clase Colors:

```
        - WHITE_ANSI = "\u001B[37m"
        - RESET_ANSI = "\u001B[0m"
        - BLACK_ANSI = "\u001B[30m"
        - RED_ANSI = "\u001B[31m"
        - GREEN_ANSI = "\u001B[32m"
        - CYAN_ANSI = "\u001B[36m"
        - YELLOW_ANSI = "\u001B[33m"
        - BLUE_ANSI = "\u001B[34m"
        - PURPLE_ANSI = "\u001B[35m"
        - BLACK_BACKGROUND_ANSI = "\u001B[40m"
        - RED_BACKGROUND_ANSI = "\u001B[41m"
        - GREEN_BACKGROUND_ANSI = "\u001B[42m"
        - YELLOW_BACKGROUND_ANSI = "\u001B[43m"
        - BLUE_BACKGROUND_ANSI = "\u001B[44m"
        - PURPLE_BACKGROUND_ANSI = "\u001B[45m"
        - CYAN_BACKGROUND_ANSI = "\u001B[46m"
        - WHITE_BACKGROUND_ANSI = "\u001B[47m"

    Método errMsg(msg):
        Imprimir "Error: " seguido de msg con el color RED_ANSI y luego RESET_ANSI
    Método warMsg(msg):
        Imprimir "War: " seguido de msg con el color CYAN_ANSI y luego RESET_ANSI
    Método okMsg(msg):
        Imprimir "OK: " seguido de msg con el color GREEN_ANSI y luego RESET_ANSI
```

Clase ReadClient:

```
    Atributos:
        - sc (objeto de tipo Scanner para leer desde la consola)

    Constructor ReadClient():
        Inicializar el atributo sc con un nuevo objeto de tipo Scanner(System.in)

    Método bufferClear():
        Limpia el buffr

    Método pedirInteger(msg):
        Repetir mientras repit sea verdadero:
            Establecer repit como falso
            Mostrar el mensaje msg
            Intentar leer un número entero desde la entrada estándar y almacenarlo en
la variable num
            Si se produce una excepción InputMismatchException:
                Mostrar un mensaje de advertencia indicando que se debe introducir un
número entero
                Descartar la entrada incorrecta llamando a sc.next()
                Establecer repit como verdadero
        Devolver num (el número entero leído)

    Método pedirDouble(msg, min, max):
        Repetir mientras repit sea verdadero:
            Establecer repit como falso
            Mostrar el mensaje msg
            Intentar leer un número decimal desde la entrada estándar y almacenarlo
en la variable num
            Si se produce una excepción InputMismatchException:
                Mostrar un mensaje de advertencia indicando que se debe introducir un
número decimal
                Descartar la entrada incorrecta llamando a sc.next()
                Establecer repit como verdadero
            Si min y max no son nulos y num está fuera del rango:
                Mostrar un mensaje de advertencia indicando que se debe introducir un
número dentro del rango [min, max]
```

```
                Establecer repit como verdadero
        Devolver num (el número decimal leído)

    Método pedirString(msg):
        Repetir mientras repit sea verdadero:
            Mostrar el mensaje msg
            Establecer repit como falso
            Intentar leer una línea de texto desde la entrada estándar y almacenarla
en la variable str
            Si str es nulo o una cadena vacía:
                Mostrar un mensaje de advertencia indicando que no se pueden dejar
campos vacíos
                Establecer repit como verdadero
        Devolver str (la cadena de texto leída)
```

Clase Matricula:

```
    Atributos:
        - notes (lista de números decimales para almacenar las notas)
        - mitjana (número decimal para almacenar la media de las notas)

    Constructor Matricula():
        Inicializar la lista de notas (notes) como una lista vacía

    Método addNota(n):
        Agregar la nota (n) a la lista de notas
        Llamar al método updateMitjana() para actualizar la media

    Método setNota(nota, pos):
        Establecer la nota en la posición (pos) de la lista de notas como (nota)
        Mostrar un mensaje de éxito
        Llamar al método updateMitjana() para actualizar la media

    Método delNota(pos):
        Eliminar la nota en la posición (pos) de la lista de notas
        Llamar al método updateMitjana() para actualizar la media

    Método mostrarNotes(lista):
        Si la lista de notas no está vacía:
            Si lista es verdadero:
                Para cada nota en la lista de notas:
                    Mostrar el índice + 1 y la nota con formato
            De lo contrario:
                Para cada nota en la lista de notas (excepto la última):
                    Mostrar la nota con formato + " - "
                Mostrar la media (mitjana) con formato en color púrpura y luego en
color blanco
            Devolver 0
        De lo contrario:
            Mostrar un mensaje de advertencia indicando que el alumno no tiene notas
            Devolver -1

    Método updateMitjana():
        Inicializar sumatorio como 0
        Para cada nota en la lista de notas:
            Sumar la nota a sumatorio
        Calcular la media (mitjana) como el cociente de sumatorio entre el tamaño de
la lista de notas
        Mostrar un mensaje de éxito indicando que la media se ha actualizado
```

Clase Modul:

```
    Atributos:
        - nom (cadena de texto para almacenar el nombre del módulo)
        - matr (objeto de tipo Matricula para gestionar las calificaciones)
```

```
    Constructor Modul(nom):
        Inicializar el atributo nom con el valor de nom
        Inicializar el atributo matr como un nuevo objeto de tipo Matricula()
```

Clase Alumne:

```
    Atributos:
        - nom (cadena de texto)
        - nia (cadena de texto)
        - modulsList (lista de objetos de tipo Modul)

    Constructor Alumne(nom, nia):
        Inicializar el atributo nom con el valor de nom
        Inicializar el atributo nia con el valor de nia
        Inicializar la lista modulsList como una lista vacía

    Método buscarModul(nomModul):
        retorno = -1
        Para cada elemento i en modulsList:
            Si modulsList[i].nom es igual a nomModul:
                Asignar i a retorno
                Romper el bucle
        Devolver retorno

    Método mostrar():
        Imprimir "Nombre: " seguido de nom
        Imprimir "NIA: " seguido de nia
        Para cada módulo m en modulsList:
            Imprimir el nombre del módulo m.nom seguido de ": "
            Llamar al método mostrarNotes(false) del objeto m.matr (asumiendo que
matr es un atributo de tipo Matricula)
```

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
```

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(modul):
        Para cada alumno en la lista de alumnos:
            Obtener la posición del módulo en la lista de módulos del alumno usando
buscarModul(modul, alumno.modulsList)
            Si la posición es diferente de -1:
                Eliminar el módulo de la lista de módulos del alumno
```

Clase Matriculas:

```
    Atributos:
        - rc (objeto de tipo ReadClient para lectura de entrada)

    Método menu():
        Leer el NIA del alumno desde la entrada del usuario y almacenarlo en nia
        Obtener la posición del alumno en la lista de alumnos usando buscarNia(nia)
        Si posNia es diferente de -1:
            Obtener el objeto Alumne correspondiente al alumno
            Leer el nombre del módulo desde la entrada del usuario y almacenarlo en
modul
            Obtener la posición del módulo en la lista de módulos del alumno usando
buscarModul(modul)
            Si posModul es diferente de -1:
                Obtener el objeto Matricula correspondiente al módulo del alumno
                Repetir mientras repetir sea verdadero:
```

```
                    Mostrar el menú de Avaluar:
                        (0) Salir
                        (1) Qualificar
                        (2) Modificar
                        (3) Mostrar boletín de notas
                    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 Avaluar"
                        Caso 1:
                            Llamar al método qualificar(matr)
                        Caso 2:
                            Llamar al método modificar(matr)
                        Caso 3:
                            Llamar al método mostrar()
                        De lo contrario:
                            Mostrar un mensaje de advertencia indicando que se debe
introducir un valor válido
            De lo contrario:
                Mostrar un mensaje de error indicando que el alumno no está inscrito
en ese módulo
        De lo contrario:
            Mostrar un mensaje de error indicando que el alumno no existe

    Método qualificar(matr):
        Leer la cantidad de notas a agregar desde la entrada del usuario y
almacenarla en cant
        Para cada nota en el rango de 1 hasta cant (inclusive):
            Leer la nota desde la entrada del usuario y almacenarla en nota
            Llamar al método addNota(nota) de matr para agregar la nota
            Mostrar un mensaje de éxito

    Método modificar(matr):
        Si matr.mostrarNotes(true) no devuelve -1:
            Leer la posición de la nota a modificar desde la entrada del usuario y
restar 1 para obtener la posición real en la lista
            Leer la nueva nota desde la entrada del usuario y almacenarla en nota
            Llamar al método setNota(nota, pos) de matr para modificar la nota
```

Clase Practica_02:

```
    Atributos:
        - alumnesList (objeto de tipo Alumnes)
        - modulsList (objeto de tipo Moduls)

    Método main(args):
        Crear un objeto de tipo ReadClient llamado rc
        Crear un objeto de tipo Matriculas llamado matriculas
        Inicializar la variable repit como verdadera

        Repetir mientras repit sea verdadero:
            Mostrar el menú principal:
                (0) Salir
                (1) Menu Alumnes
                (2) Menu Modul
                (3) Avaluar
            Leer la opción del usuario y almacenarla en menu

            Según el valor de menu:
                Caso 0:
                    Establecer repit como falso
                    Mostrar "Has salido de la aplicación"
                Caso 1:
                    Llamar al método menu() de alumnesList (Menú de Alumnes)
                Caso 2:
```

```
                    Llamar al método menu() de modulsList (Menú de Moduls)
                Caso 3:
                    Llamar al método menu() de matriculas (Menú de Avaluar)
                De lo contrario:
                    Mostrar un mensaje de advertencia indicando que se debe
introducir un valor válido
```

Conclusió

En conclusió, aquesta pràctica m'ha sigut de gran utilitat per repassar àmpliament la programació en Java. A més, la creació de les classes 'ReadClient' i 'Colors' em serà de molta ajuda en futures pràctiques. Espere no haver de corregir molts errors en aquest codi en particular. També confie en què les pròximes pràctiques no siguen tan extenses com aquesta i que, donat que en les següents només s'han de modificar parts del codi, no haja de fer canvis significatius ni dedicar-l'hi moltes hores.