M03 Programació bàsica

UF3 CAS6 Fonaments de gestió de fitxers

CFGS: ASIX - DAW

Mòdul Professional: M03 Programació Bàsica

UF: 3

Professor/a assignat: Maria Merino

Membres: Alberto Dos Santos i Isaac Brull

Acabament del treball: Terres de l'Ebre 29/05/2020

ÍNDEX

Introducció

En aquest treball, treballarem els fonaments de gestió de fitxers en Java. Farem server noves eines de treball com per exemple el “try” i “catch” entre altres. El nostre objectiu principal sera crear un programa que sera un joc per a practicar anglès i on podrem afegir tantes paraules com nosaltres vulguem. També tindrem en compte la configuració del programa que es pugui realitzar pel propi usuari per a que pugui arxivar el resultat de la partida anterior o crear registres.

Teoría

Explicarem les coses noves que hem utilitzat en aquest treball.

- Try Catch

El try s'utilitza quan es desitgen preveure excepcions en el transcurs de la execució d'un programa. És necessari ja que allí es on es col·loquen les instruccions que es desitjen realitzar, posteriorment s'utilitza un catch on s'epecifica la excepció que es sap que pot passar o donar-se a terme.

Exemple de try catch:

```
class Main {
  public static void main(String[] args) {
    int[] a = new int[10];
    try{
      a[-1] = 3;
    }
    catch(Exception e){
      System.out.println("Missatge d'error personalitzat");
    }
    System.out.println("Este missatge sí que l'imprimirà");

    a[-1] = 3;
    System.out.println("Però este no. Aquí no arriba l'execució");
  }
}
```

Com podem veure, en aquest exemple podem crear un missatge d'error personalitzat mentre que segons les condicions establertes, arribarà un punt que saltara l'error i que l'última línia no s'executarà.

- Split
- Random
- File

Per fer aquest programa ha sigut crucial utilitzar File ja que hens ha servit per a obrir els fitxers i conté mètodes molt útils

- ArrayList

En aquest cas hem treballat amb fitxers, i aquests poden tenir dades “il·limitades” per aixo al programa no podiem marcar un màxim i per aquesta rao hem utilitzat ArrayList per a crear arrays “indefinits”.
També aquesta classe te diferents mètodes com el .size() o el .get() que canvie pero són prou fàcil d’utilitzar.

- Llegir i escriure a fitxers

Per llegir i escriure a fitxers hem utilzat PrintStream per a escriure als fitxers i Scanner per a llegir als fitxers.

Classe Joc

En aquesta classe utilizarem OOP, utilitzarem objectes que cadascun representara una partida, també contindran diferents atributs:

```
final String idiomaPrincipal;

String[] parellaActual;

ArrayList<Integer> paraulesFetes;
ArrayList<String> parelles;
int codiIdioma;
```

IdiomaPrincipal = marcara el idioma que és mostrara al usuari
parella actual = l'utilizarem per a saber quina parella esta traduint l'usuari
paraulesfetes = guardarem l'index de les parelles anteriorment traduides
parelles = guardarem totes les parelles del fitxer parelles.txt

Primer de tot el constructor:

```
public Joc(String idioma) {
    this.parelles = new ArrayList<String>();
    this.idiomaPrincipal = idioma;
    this.paraulesFetes = new ArrayList<Integer>();
    this.loadParelles();
    this.start();
}
```

El contructor crea un arraylist a parelles, aplica el parametre idioma a idiomaPrincipal i crea l'array de paraules fetes.
Seguidament crida loadparelles() que agafa les parelles del arxiu i les guarda al arraylist

```java
public void loadParelles() {    // Aquest mètode guarda totes les parell
    try {
        File fitxer = new File("parelles.txt");
        Scanner arxiu = new Scanner(fitxer);

        do{
            String linea = arxiu.nextLine();

            String[] parellesLinea = linea.split(",");
            for (int cada = 0; cada < parellesLinea.length; cada++) {
                this.parelles.add(parellesLinea[cada]);
            }
        }while(arxiu.hasNextLine());
        System.out.println("debug- paraules carregades correctament.");
        arxiu.close();
    } catch (Exception e) {
        // Excepció!

        e.printStackTrace(System.out);
    }
}
```

despres start() comença el joc.

```java
public void start() {
    // Aquest if que tenim a la part inferio
    // aixo influeix en la manera que guarde
    // les paraules en catala van primer i d
    // sapigue quin idioma te que inprimir.
    String idiomaSecundari;

    if (this.idiomaPrincipal == "catala") {
        this.codiIdioma = 0;
        idiomaSecundari = "angles";
    } else {
        idiomaSecundari = "catala";
        this.codiIdioma = 1;
    }
```

Avans de començar a mostrar paraules per pantalla utilizarem una variable codiIdioma per a gestionar parella actual, ja que 0 sera la primera posició i la primera és catala, i la segona anglès, Exemple:

Parella	[paraula_catala,paraula_anglès]	
Codildioma	0	1

```
System.out.println("Començem:");
while (this.paraulesFetes.size() != this.parelles.size()) {
    System.out.println("Tradueix la paraula : " + this.displayWord());
    if (this.checkPair(sc.nextLine())) {
        Start.registreActual.registrar("Traducció correcta del " + this.idiomaPrincipal
          Arrays.toString(this.parellaActual));

        // Registrar paraula correcta.Registrar Traduccio correcta
    } else {
        Start.registreActual.registrar("Traducció incorrecta del " + this.idiomaPrincipa
          Arrays.toString(this.parellaActual));
        // Mostra la ratxa que tenia l'usuari.
        // Seguent paraula
        //
        // Traduccio incorrecta
    }
}
System.out.println("S'han acabat les parelles, afegeix més per seguir jugant.");
Start.registreActual.guardar();
```

Aquest és tot el codi de start()
Al while com a condició mentres les parelles fetes sigui inferior a les parelles totals.
Després utilitzem displayWord per enseyar una paraula al usuari

```
public String displayWord() {
    int numRandom;

    do{
        numRandom = (int)(Math.random() * (this.parelles.size()));
    }while(this.paraulesFetes.contains(numRandom));
    this.paraulesFetes.add(numRandom);
    this.parellaActual = parelles.get(numRandom).split("-");
    return (this.parellaActual[this.codiIdioma]);
    // Agafar una paraula de la llista de paraules i guardar-la a al
    // Podriem guardar el index de la parella per saber si l'em fet
}
```

una vegad hem ensenyat la paraula al usuari tenim que comprovar que l'entrada al usuari correspongui a la traduccio de la paraula, per aixo hem creat una funcio anomenada checkPair

```java
public boolean checkPair(String traduccio) {
    if (this.codiIdioma == 1) {
        if (this.parellaActual[0].equals(traduccio)) {
            System.out.println("Correcte!");
            return (true);
        } else {
            System.out.println("Erroni!");
            return (false);
        }
    } else {
        if (this.parellaActual[1].equals(traduccio)) {
            System.out.println("Correcte!");

            return (true);
        } else {
            System.out.println("Erroni!");

            return (false);
        }
    }
```

Una vegada sabem si la enceratat o ha fallat ho registrem al registre.
Cuan el while s'acabi veurem per pantalla que s'han acabat les parelles.

Classe Gestio

Aquesta classe la utilitzarem per a gestionar les parelles, s'encarregara de afegir modificar o esborrar parelles depenent de que vulgui fer l'usuari.

Primer de tot hem fet un menu i un switch per a accedir a diferents mètodes.

```
public static void configuracio() {
    System.out.println("Accions :" + "\"1\": Afegir parella al llistat." + "\"2\": Modifi
      "\"3\": Esborrar parella del llistat." + "\"4\": Arxivar el registre actual." + "\"
    switch (sc.nextLine()) {
    case "1":               // afegir
        afegir();
        break;
    case "2":               // modificar
        esborrar();
        System.out.println("Torna a introduir les dades correctes de la parella borrada a
        afegir();
        break;
    case "3":               // esborrar
        esborrar();
        break;
    case "4":
        Start.registreActual.arxivar();
        break;
    }
}
```

És pot apreciar que per a modificar una parella simplement hem utilitzat afegir i esborrar ja que és la manera més simple.

```java
public static void afegir() {
    // Afegir una parella de paraules al registre de paraules-partides. Ens demanarà la parau
    // i la paraula en anglès.
    System.out.println("Introduieix la paraula en català: (sense espais i en minuscula)");

    String catala = sc.nextLine();

    System.out.println("Introduieix la paraula en anglès: (sense espais i en minuscula)");

    String angles = sc.nextLine();
    String lineaCompleta = catala + "-" + angles + ",";

    try {
        PrintStream escriptor = new PrintStream(new FileOutputStream("", true));    // L'hem

        escriptor.write(lineaCompleta.getBytes());
        escriptor.close();
    } catch (Exception e) {
        // Excepció!

        e.printStackTrace(System.out);
    }
}
```

Afegir primer de tot guardem les dades de la parella i les juntem a un string conjunt anomenat lineaCompleta
Després accedim al arxiu en mode “afegir” aixo fa que al escriure a un fitxer les dades que havien anteriorment no s’esborren sino que és afegeix apart.
També per a escriure hem posat l’estring i .getBytes() ja que per a escriure amb PrintStream

Classe Registre

Crearem un registre de paraules/partides, que guardarem en un fitxer. De cada paraula guardarem la paraula en català, la seva traducció a l'anglès, si l'hem sabut traduir. Les parelles de paraules les introduirem nosaltres. Si l'hem sabut traduir o no ho anirà anotan l'aplicació quan acabem cada partida.

Per a fer la classe Registre, crearem un atribut anomenat registreComplet i el constructor adien a ell que de moment no li escriurem cap registre ja que utilitzarem el mètode registrar() per a guardar dades al registreComplet.

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.PrintStream;
import java.util.Scanner;

public class Registre {
        // De cada paraula guardarem la
// paraula en catala , la seva traducció a I
// sabut traduir o no de l'angles al catala
// "quan acabem cada partida.""
        String registreComplet;

        public Registre() {
                this.registreComplet = "";
        }
```

En la següent funció, farem un paràmetre d'entrada de tipus String i es guarda a un string principal.

```
/**
 * Funció registrar
 * metode que com a parametre entra String i es guarda a un string Principal
 * @param principal String
 */
public void registrar(String principal) {
        this.registreComplet += ",\n" + principal;
}
```

En aquest mètode, agafa el String principal i l'escriu al registre.TXT

```java
/**
 * Funció guardar
 *  metode que agafa el String principal i l'escriu al registre.TXT
 *
 */
public void guardar() {
        try {
                PrintStream escriptor = new PrintStream(new FileOutputStream("src/registre.txt", true));

                escriptor.write(this.registreComplet.getBytes());
                escriptor.close();
        } catch (Exception e) {
                // ExcepciÃ³!

                e.printStackTrace(System.out);
        }
}
```

En l'últim mètode de la classe, farem que pregunte a l'usuari un nom d'un arxiu, el crea i guarda el registre actual a l'arxiu específic.

La tècnica que hem utilitzat ha sigut que hem canviat el nom del arxiu registre.txt a el nom que l'usuari volia i hem creat un arxiu nou anomenat registre.txt

```java
/**
 * Funció arxivar
 *  metode arxivar que pregunta al usuari un nom d'un arxiu, el crea i guarda el registre actual a
 *
 */
public void arxivar() {
        // canviar de nom al registre i creac un registre nou
        Scanner sc = new Scanner(System.in);

        System.out.println("Introdueix el nom de l'arxiu per arxivar el registre actual.");

        String nomArxiu = sc.nextLine();

        try {
                File arxivat = new File(nomArxiu + ".txt");

                if (!arxivat.exists()) {
                        arxivat.createNewFile();
                }

                PrintStream escriptor = new PrintStream(new FileOutputStream(nomArxiu + ".txt"));
                File fitxer = new File("src/registre.txt");

                sc = new Scanner(fitxer);
                do{
                        escriptor.write(sc.nextLine().getBytes());
                }while(sc.hasNextLine());
                escriptor.close();
        } catch (Exception e) {
                // ExcepciÃ³!

                e.printStackTrace(System.out);
        }
}
```

Classe Start

Aquesta és la més senzilla de les classes ja que es el main i només caldrà cridar als metodes i funcions que ens faci falta.
Quan executem el programa ens anirà preguntan una serie de coses que volem fer però en aquest cas anem directament a jugar.
En aquest try el que fem es una serie de if que el que fan bàsicament és comprovar que els fitxers estiguin creats i en el cas que no estiguin creats els crea gràcies a “createNewFile”.

```java
import java.util.Scanner;
import java.io.File;

public class Start {
    public static void checkFiles() {
        File parelles = new File("parelles.txt");
        File registre = new File("registre.txt");
        try{
            if (!parelles.exists()) {
                parelles.createNewFile();
            }
            if (!registre.exists()) {
                registre.createNewFile();
            }
        }
        catch (Exception e) {
            // Excepció!

            e.printStackTrace(System.out);
        }
    }
```

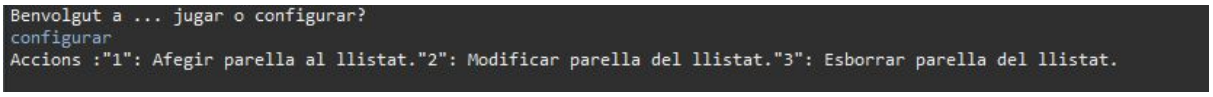
A l'hora de fer el main, farem un menu dintre d'un altre menu ja que l'opció de jugar, una vegada la triem, tenim que triar en quin idioma vole començar a jugar.

```
    public static Registre registreActual = new Registre();
    public static void main(String[] args) {
        checkFiles();

        Scanner sc = new Scanner(System.in);
        Joc partida;

        System.out.println("Benvolgut a ... jugar o configurar?");
        switch (sc.nextLine()) {
        case "jugar":
            System.out.println("Quin idioma vols traduir? catala/angles");
            switch (sc.nextLine()) {
            case "angles":
                partida = new Joc("catala");
                break;
            case "catala":
                partida = new Joc("angles");
                break;
            }
            break;
        case "configurar":
            Gestio.configuracio();
            break;
        }
    }
}
```

Com podem veure, una vegada triem l'opció de jugar, ens preguntara que si volem traduir del català a l'anglès o de l'anglès al català i una vegada seleccionat, comencem a jugar.

```
Benvolgut a ... jugar o configurar?
jugar
Quin idioma vols traduir? catala/angles
catala
debug- paraules carregades correctament.
Començem:
Tradueix la paraula : egg,
```

Per l'altra part, si triem l'opció de configurar, podrem triar entre les 3 següents:



i depèn el que triem farem el que ens demana en cada cas.

Conclusió

En aquest treball hem après a manipular fitxers, a llegir-los i escriure'ls informació, també hem après a utilitzar altres eines i altres Paquets molt útils. El confinament del covid19 ha fet que no poguessim fer aquest programa com tocava per falta de temps. Ens haguessa agradat profundir més amb el tema i expandir el codi però no s'ha pogut fer.