PROJECTE UF5

Implementació dels continguts de la UF5
al projecte realitzat a la UF3

Aleix Melich Also
1r DAM
Tortosa, 10 de Maig de 2024

Índex de continguts

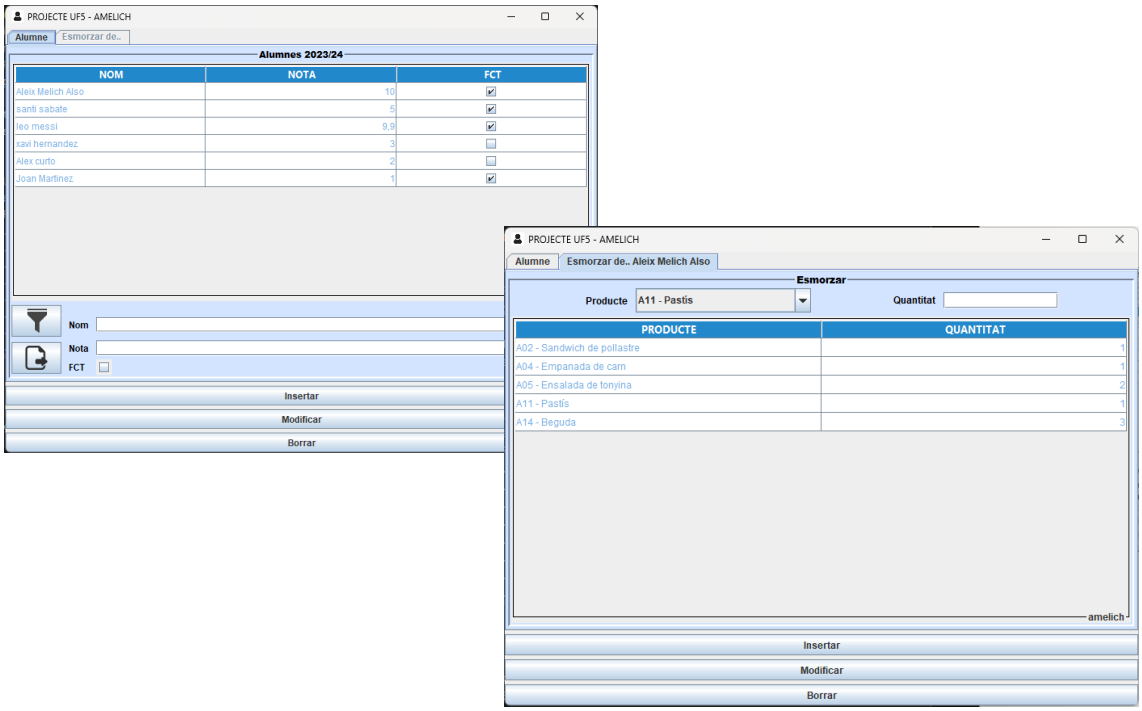
Introducció

El present projecte integral té com a objectiu principal la consolidació dels continguts, resultats d'aprenentatge i criteris d'avaluació de la Unitat Formativa (UF) en qüestió.

El projecte es desenvoluparà en diverses fases clarament definides, amb una durada estimada d'unes 40 hores, equivalent a la durada real de la UF5. En cada fase, es requerirà l'entrega de diversa documentació i materials relacionats amb els objectius específics establerts. Aquesta documentació està presentada en format PDF i s'emmagatzemarà a l'arrel del repositori de GitHub, dins la carpeta designada com a "documentacio".

El contingut del projecte inclourà el codi font del mateix projecte, així com altres elements requerits a cada fase particular. A la documentació es trobarà el contingut i explicació de les cinc fases juntament amb les funcionalitats extres que el reforcen i la conclusió del projecte.

Cal afegir també que tots els mètodes i classes estan documentades amb el seu respectiu JavaDOC i també hi han alguns comentaris pel codi cosa que el fan mes entenedor per una possible ampliació o per la lectura o interès del !creador.

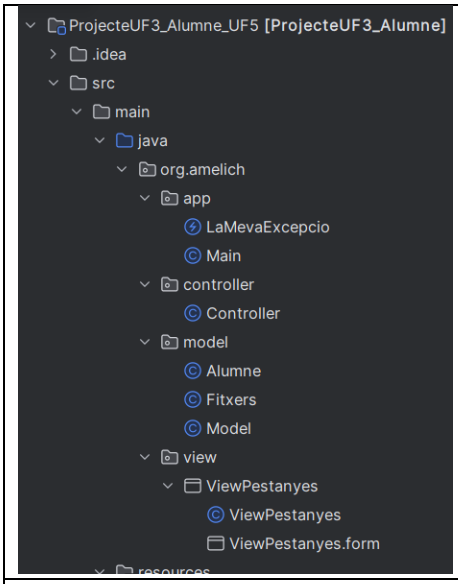


Fase 1

La primera fase del projecte es basava mes o menys en una adaptació d'un projecte CRUD a un paradigma MVC (Model-Vista-Controlador). Teníem el projecte de la UF3 i el que he optat per fer ha estat usar-lo per dur a terme el projecte d'aquesta UF.

Al principi no sabia per on començar ja que aplicar la paradigma MVC canvia per complet l'estructura del projecte però agafant com a guia les explicacions del projecte que anàvem fent a classe ho he pogut solventar força be.

Aquesta ha estat l'estructura final:



app

- LaMevaExcepcio: Aquesta classe d'excepció es fa servir per a gestionar errors específics de l'aplicació.
- Main: Aquest fitxer conté la funció main, que és la primera funció que s'executa quan s'inicia l'aplicació.

controller

- Controller: Aquesta classe és responsable de gestionar la lògica de l'aplicació, incloent la interacció entre la vista i el model. Defineix mètodes per a assignar comportaments als botons i a la taula de la vista, així com per a gestionar canvis de propietats i excepcions.

view

- ViewPestanyes: Aquesta es la nova classe que incorpora la creació i configuració de l'aplicació.
- ViewPestanyes.view: Representa la GUI de ViewPestanyes.

model

- Alumne: És un POJO (Plain Old Java Object) que representa un alumne.
- Fitxers: És responsable de la lectura i escriptura de dades en un fitxer.
- Model: Aquesta clase seria la responsable de la gestió de les dades de la taula que despres es mostra a l'usuari.

Un altre punt a tractar sobre la Fase 1 ha estat que havíem de crear un tipus d'excepció personalitzada i configurar-la per tots aquells missatges o excepcions que teníem configurades per al projecte anterior. Havíem d'implementar-ho usant la interfície Map<K,v> (TreeMap, HashMap, …).

Fitxer LaMevaExcepcio.java

LaMevaExcepcio.java

```java
package org.amelich.app;

import java.util.HashMap;
import java.util.Map;

public class LaMevaExcepcio extends Exception {
    private static final Map<Integer, String> errorMessages = new HashMap<>();

    static {
        errorMessages.put(1, "Per modificar o borrar una fila l'has de seleccionar a la taula");
        errorMessages.put(2, "No pots introduir cap numero en aquest camp, nomes caracters.");
        errorMessages.put(3, "Aquest nom ja esta inscrit a la taula, canvial.");
        errorMessages.put(4, "Has d'introduir una nota correcta (0-10) i si te decimals separar-la per comes.");
        errorMessages.put(5, "Falta omplir alguna dada, revisa-ho.");
        errorMessages.put(6, "No pots introduir cap caràcter en aquest camp, només números.");
        errorMessages.put(7, "No hi han dades per exportar.");
        errorMessages.put(8, "Error al crear el fitxer de sortida.");
        // Afegeix més missatges d'error segons sigui necessari
    }

    private int errorCode;

    public LaMevaExcepcio(int errorCode) {
        super(errorMessages.getOrDefault(errorCode, defaultValue: "Error desconegut"));
        this.errorCode = errorCode;
    }

    public String retornaMissatge() { return super.getMessage(); }

    public int retornaNumero() { return this.errorCode; }

}
```

També havíem d'usar un PropertyChangeListener per tractar aquestes excepcions, això ha estat realitzat al document Controller.java.

```java
public class Controller implements PropertyChangeListener { //1. Implementació de interfície PropertyChangeListener

    //2. Propietat lligada per controlar quan genero una excepció
    public static final String PROP_EXCEPCIO="excepcio";
    private LaMevaExcepcio excepcio;

    public LaMevaExcepcio getExcepcio() { return excepcio; }

    public void setExcepcio(LaMevaExcepcio excepcio) {
        LaMevaExcepcio valorVell=this.excepcio;
        this.excepcio = excepcio;
        canvis.firePropertyChange(PROP_EXCEPCIO, valorVell,excepcio);
    }

    //3. Propietat PropertyChangesupport necessària per poder controlar les propietats lligades
    PropertyChangeSupport canvis=new PropertyChangeSupport( sourceBean: this);

    //4. Mètode on posarem el codi de tractament de les excepcions --> generat per la interfície PropertyChangeListener
    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        LaMevaExcepcio rebuda=(LaMevaExcepcio)evt.getNewValue();

        try {
            throw rebuda;
        } catch (LaMevaExcepcio e) {
            //Aquí farem ele tractament de les excepcions de l'aplicació
            switch (evt.getPropertyName()) {
                case PROP_EXCEPCIO:

                    switch (rebuda.retornaNumero()) {
                        case 1:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge(), title: "Avís", JOptionPane.WARNING_MESSAGE);
                            break;
                        case 2, 3:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge(), title: "Error", JOptionPane.ERROR_MESSAGE);
                            this.view.getCampNom().setSelectionStart(0);
                            this.view.getCampNom().setSelectionEnd(this.view.getCampNom().getText().length());
                            this.view.getCampNom().requestFocus();
                            break;
                        case 4:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge(), title: "Error", JOptionPane.ERROR_MESSAGE);
                            this.view.getCampNota().setSelectionStart(0);
                            this.view.getCampNota().setSelectionEnd(this.view.getCampNota().getText().length());
                            this.view.getCampNota().requestFocus();
                            break;
                        case 5:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge());
                            if (this.view.getCampNom().getText().isBlank()) {
                                this.view.getCampNom().requestFocus();
                            } else if (this.view.getCampNota().getText().isBlank()) {
                                this.view.getCampNota().requestFocus();
                            } else {
                                this.view.getCampQuantitat().requestFocus();
                            }
                            break;
                        case 6:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge(), title: "Error", JOptionPane.ERROR_MESSAGE);
                            this.view.getCampQuantitat().setSelectionStart(0);
                            this.view.getCampQuantitat().setSelectionEnd(this.view.getCampQuantitat().getText().length());
                            this.view.getCampQuantitat().requestFocus();
                            break;
                        case 7, 8:
                            JOptionPane.showMessageDialog( parentComponent: null, rebuda.retornaMissatge(), title: "Error", JOptionPane.ERROR_MESSAGE);
                            break;
                    }
```

També es necessari afegir aquesta línia per poder implementar el PropertyChangeListener dins el codi.

```java
    public Controller(Model model, Vista view) {
        this.model = model;
        this.view = view;

        //Mètode per lligar la vista i el model
        lligarVistaModel();

        //Assigno el codi dels listeners
        assignarCodiListeners();

        //5. Necessari perquè Controller reaccioni davant de canvis a les propietats lligades
        canvis.addPropertyChangeListener(this);
    }
```

Aquí tenim un exemple d'una excepció aplicada a una ordre del projecte, en aquest cas tenim un FocusListener que determina el nom, si al campNom se li insereix un dígit retorna un error de codi 3 fent us del controlador creat anteriorment setExcepcio.

```
// EN AQUEST CAS DETERMINEM QUE S'HA D'INTRODUIR UNA NOM VALID ON NOMES PERMETRÀ CARACTERS
campNom.addFocusListener((FocusAdapter) focusLost(e) -> {
        super.focusLost(e);

        String nom=(campNom.getText());
        if (nom.matches( regex: ".*\\d.*")){
            setExcepcio(new LaMevaExcepcio( errorCode: 2));
        }
});
```

L'error de codi 2 esta configurat a la pestanya *LaMevaExcepcio.java* i te com a resultat la següent línia:

LaMevaExcepcio.java

```
1   package org.amelich.app;
2
3   import java.util.HashMap;
4   import java.util.Map;
5
6   public class LaMevaExcepcio extends Exception {
7       private static final Map<Integer, String> errorMessages = new HashMap<>();
8
9       static {
10          errorMessages.put(1, "Per modificar o borrar una fila l'has de seleccionar a la taula");
11          errorMessages.put(2, "No pots introduir cap numero en aquest camp, nomes caracters.");
12          errorMessages.put(3, "Aquest nom ja esta inscrit a la taula, canvial.");
```

Fase 2

En aquesta fase potser ha estat la que li he invertit mes temps, es basava crear una nova classe al paquet model anomenada SuperCollection que sigui capaç d'actuar com diferents tipus de col·leccions segons el valor d'alguna de les seues propietats. Aquestes col·leccions les tenies configurades al projecte i després de que triïs la que prefereixis ell ja t'obria la segona pestanya i configurava la llista de (en el meu cas) productes amb la col·lecció establerta anteriorment.

Jo ho he fet directe indicant que la col·lecció amb la qual van els productes es de tipus TreeSet, la configuració per establir aquesta opció ha estat implementada al fitxer *Alumne.java* i ha estat la següent:

Collection s'utilitza com a tipus de dades per a la variable esmorzars en la classe Alumne.

```java
public class Alumne implements Serializable {
    private String nomCognom;
    private double nota;
    private boolean fct;

    private Collection<Esmorzar> esmorzars;

    /**
     * @param nomCognom
     * @param nota
     * @param fct
     * @param esmorzars
     */
    public Alumne(String nomCognom, double nota, boolean fct, Collection<Esmorzar> esmorzars) {
        this.nomCognom = nomCognom;
        this.nota = nota;
        this.fct = fct;
        this.esmorzars = esmorzars;
    }
```

Collection és una interfície en Java que representa un grup d'objectes, coneguts com a elements. En aquest cas, els elements són objectes de la classe Esmorzar.

```java
public Collection<Alumne.Esmorzar> getEsmorzars() { return esmorzars; }

private void setEsmorzars(Collection<Esmorzar> esmorzars) { this.esmorzars = esmorzars; }
```

La variable esmorzars és una col·lecció d'objectes Esmorzar, que representa els esmorzars que un alumne pot tenir.

```java
public static class Esmorzar implements Comparable<Esmorzar>, Serializable {
    public int compareTo(Esmorzar o) { return this.producte.compareTo(o.getProducte()); }

    public static enum Producte {
        A01( nom: "Entrepa de pernil i formatge"), A02( nom: "Sandwich de pollastre"),
        A03( nom: "Croissant de xocolata"), A04( nom: "Empanada de carn"),
        A05( nom: "Ensalada de tonyina"), A06( nom: "Poma"), A07( nom: "Iogurt natural"),
        A08( nom: "Barreta de cereals"), A09( nom: "Suc de taronja"),
        A10( nom: "Batut de fruites"), A11( nom: "Pastís"),
        A12( nom: "Truita de patates"), A13( nom: "Patates fregides"), A14( nom: "Beguda");
        private String nom;

        private Producte(String nom) { this.nom = nom; }

        public String getNom() { return nom; }

        @Override
        public String toString() { return this.name()+" - " +nom; }
    }
```

Cada Esmorzar conté un Producte i una quantitat.

```
private Esmorzar.Producte producte;
private int quantitat;

public Esmorzar(Esmorzar.Producte producte, int quantitat) {
    this.producte = producte;
    this.quantitat = quantitat;
}
```

Finalment guardem aquest Col·lecció de Esmorzar en un TreeSet, entenc que la implementació de SuperCollection es que aquí li indicàvem la col·lecció triada i afegia el Esmorzar.

```
} else{
    Alumne al = new Alumne(campNom.getText(), nota, SI_CheckBox.isSelected(), new TreeSet<Esmorzar>());
    model.addRow(new Object[]{campNom.getText(), nota, SI_CheckBox.isSelected(), al});
    JOptionPane.showMessageDialog( parentComponent: null,  message: "Has inscrit
```

Sincerament ara després de documentar-ho donat a que depenent del tipus de col·lecció que fiquéssim a la captura superior faria una cosa o una altra potser haguessa estat fàcil implementar un combobox indicant la col·lecció que prefereixes i modificant el TreeSet per una variable que guardi tots els tipus de Colleccions.. però ja no hi ha temps i prefereixo entregar el projecte dins de la data limit establerta.

Fase 3

Aquesta fase consistia en crear expressions regulars en aquells casos com per exemple en la gestió de valors incorrectes dins els camps del formulari.

Sobre aquesta gestió de valors nosaltres les tractàvem de la següent manera:

- Un nom vàlid on nomes permetrà caràcters (CampNom)
- Una nota valida que nomes permetrà números, tambe s'han implementat els decimals (CampNota)
- Una quantitat valida que nomes permetrà números (CampQuantitat)

Per establir aquests resultats m'ha ajudat el paràmetre *matches* el qual indica si la cadena amb la qual se l'ajunta coincideix o no amb la expressió regular introduïda.

Aquí tenim els dos casos.

- L'expressió regular *".*\\d.*"* s'utilitza per comprovar si una cadena de text conté qualsevol dígit numèric. En aquest cas, si el conte retorna l'error de codi 2.
  També cal afegir que es una expressió força senzilla però no m'he volgut complicar molt ja que després per ficar un insert, abans no fiques el nom amb la correcta escriptura estàs una bona estona i si estàs en un procés de disseny que no pares de fer proves doncs molesta un poc.

```
campNom.addFocusListener((FocusAdapter) focusLost(e) -> {
        super.focusLost(e);

        String nom=(campNom.getText());
        if (nom.matches( regex: ".*\\d.*")){
            setExcepcio(new LaMevaExcepcio( errorCode: 2));
        }
});
```

- L'expressió regular *"[0-9]+([.,][0-9]+)?"* en el codi seleccionat permet números que estan formats per una o més xifres, seguits opcionalment per un punt o una coma i una o més xifres. Això permet validar números enters i decimals.
  En cas de que no coincidir amb dígits numèrics o comes, retornarà el codi d'error 4.

```
campNota.addFocusListener((FocusAdapter) focusLost(e) → {
        super.focusLost(e);

        if (!campNota.getText().matches( regex: "[0-9]+([.,][0-9]+)?")) {
            setExcepcio(new LaMevaExcepcio( errorCode: 4));
        }

        try {
            NumberFormat num=NumberFormat.getNumberInstance(Locale.getDefault());
            double nota= num.parse(campNota.getText().trim()).doubleValue();
            if (nota<0 || nota>10) {
                setExcepcio(new LaMevaExcepcio( errorCode: 4));
            }
        } catch (ParseException ignored) {}
});
```

- L'expressió regular *".*[a-zA-Z].*"* coincideix amb qualsevol caràcter alfabètic, tant majúscules com minúscules, al ficar algun caràcter que sigui un d'aquests retornarà el codi d'error 6.

```
campQuantitat.addFocusListener((FocusAdapter) focusLost(e) → {
        super.focusLost(e);

        if (campQuantitat.getText().matches( regex: ".*[a-zA-Z].*")) {
            setExcepcio(new LaMevaExcepcio( errorCode: 6));
        }
});
```

Fase 4

En aquesta part he creat el Fitxers.java que sen ocupa de la lectura i escriptura de fitxers. No es ben-be el mateix ja que aquí el pojo es molt diferent però he aplicat el mateix format de codi que vaig realitzar a la UF3 amb els ObjectStreams.

Fitxer Fitxers.java

Fitxers.java

```java
package org.amelich.model;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.io.*;


public class Fitxers {
    //propietat constant per definir el nom del fitxer una sola vegada dins l'aplicació
    public static final String FITXER="./src/main/resources/dades2.dat";
    static File f = new File(FITXER);        //apuntador al fitxer, però no el crea

    public static void llegirDades(DefaultTableModel model) {



        //Mirem si el fitxer ja existix (o no)
        if (f.exists() && !f.isDirectory()) {
            ObjectInputStream entrada = null;
            try {
                entrada = new ObjectInputStream(new BufferedInputStream(new FileInputStream(f)));
                while (true) {
                    Alumne alumne = (Alumne) entrada.readObject();
                    model.addRow(new Object[]{alumne.getNomCognom(), alumne.getNota(), alumne.isFct(), alumne});
                }
            } catch (EOFException ex) {
            } catch (IOException | ClassNotFoundException ex) {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Error al llegir les dades");
            } finally {
                try {
                    if (entrada != null) entrada.close();
                } catch (IOException ex) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Error al tancar el fitxer de lectura");
                }
            }
        }
    }

    public static void escriureDades(DefaultTableModel model) {

        //Mirem si hi ha dades
        if (model.getRowCount() >= 0) {
            ObjectOutputStream sortida = null;
            try {
                sortida = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(f)));
                for (int i = 0; i < model.getRowCount(); i++) {
                    Alumne escriptura = (Alumne) model.getValueAt(i,  column: 3);
                    sortida.writeObject(escriptura);
                }
            } catch (IOException ex) {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Error al guardar les dades");
            } finally {
                try {
                    if (sortida != null) sortida.close();
                } catch (IOException ex) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Error al tancar el fitxer de sortida");
                }
            }
        }
    }
}
```

Després aquests mètodes s'instancien al Model.java o al Controller.java per omplir o guardar l'ultima actualització dels objectes creats.

Fase 5

I a l'ultima fase havíem de fer us dels streams de la classe Stream, el que he ideat per implementar-ho ha estat un botó de Filtrar el qual feia la funció de filtrar els objectes inserits a la primera pestanya pel caràcter que se li indicava al InputDialog que se li obria tant prompte clicava el botó.

Sobre el codi ha estat implementat primer convertint les files de la taula a una List i sobre aquesta list quedar-me amb els objectes que començaven amb els primers caràcters que tenia a la variable caracterFiltre. Actualitzar la llista i omplir la taulaFiltrat amb els objectes que havien quedat a la llista. Si els caracterFiltre no trobava cap alumne saltava el codi d'error 9. Tot seguit el codi (línia 389-428):

```
filtreButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String caracterFiltre = JOptionPane.showInputDialog( parentComponent: null,  message: "Introdueix el caràcter

        if (caracterFiltre != null && !caracterFiltre.isEmpty()) {
            DefaultTableModel model = (DefaultTableModel) taula.getModel();

            // Convertim les files de la taula a una llista d'alumnes
            List<Alumne> alumnes = new ArrayList<>();
            for (int i = 0; i < model.getRowCount(); i++) {
                alumnes.add((Alumne) model.getValueAt(i,  column: 3));
            }

            // Convertimos el caracterFiltre a minúsculas
            String caracterFiltreLowerCase = caracterFiltre.toLowerCase();

            // Utilitzem el mètode stream() per a filtrar els alumnes
            List<Alumne> alumnesFiltrats = alumnes.stream()
                    .filter(alumne -> alumne.getNomCognom().toLowerCase().startsWith(caracterFiltreLowerCase))
                    .collect(Collectors.toList());

            // Llimpiem la taula modelFiltrat
            modelFiltrat.setRowCount(0);

            if (!alumnesFiltrats.isEmpty()) {
                // Afegim a la taula modelFiltrat només l'alumne filtrat
                for (Alumne alumne : alumnesFiltrats) {
                    modelFiltrat.addRow(new Object[]{alumne.getNomCognom(), alumne.getNota(), alumne.isFct()});
                }

                // Creem una nova taula per a la finestra de filtratge
                JTable taulaFiltrat = new JTable(modelFiltrat);
                JScrollPane scrollPane = new JScrollPane(taulaFiltrat);
                configureTable(taulaFiltrat);
                // Creem la finestra de filtratge
                JFrame finestraFiltrat = view.createFilterWindow(caracterFiltre, scrollPane);
            } else {
                setExcepcio(new LaMevaExcepcio( errorCode: 9));
            }
        }
```

He intentar fer us del paràmetre Optional però per a la finalitat en la que jo ho vull no hem valia ja que he vist que amb el paràmetre Optional nomes et deixava acabar filtrant pel primer cas que trobi o algo així i jo vull que hem filtre tots els alumnes.

Algunes funcionalitats extres

- Els colors, ja sigui de la finestra en si o de les columnes
- La finestra te com a títol “PROJECTE UF5 - AMELCH”.
- La finestra te una mida i ubicació determinada just quan s’executa el programa.
- Nomes es permet seleccionar una fila.
- No es permet moure o redimensionar les columnes.
- No es permet editar les caselles de la taula.
- En cada inici el cursor estarà preparat per ja començar a escriure al camp nom.
- Per a cada acció de clicar un botó, al codi s’executarà un *sout* de “S’ha clicat el boto de ____”. (Primera línia de cada codi de polsador)
- Per als JOptionPane.showMessageDialog() s’ha determinat el parametre JOptionPane de manera que el missatge a mostrar sigui una “information”, “wearing” o “error”.
- Quan passem per damunt de cada fila de la taula el cursor canvia a una HAND_CURSOR.
- Despres de cada insert o eliminació d’Esmorzar el combobox es desplega automàticament preparat ja per a que li seleccionem la opció mes acordada en la nostra gana.
- Quan s’obre la finestra de filtratge no obliga a tancar-la abans de tocar la finestra d’alumnes però si que es posiciona sempre per damunt d’aquesta.
- El filtreButton i exportarButton ademes de tenir cada boto una icona personalitzada tenen un missatge d’ajuda per saber que fa aquell boto, el missatge l’observem quan passem el cursor per damunt.
- Si el producte ja esta inclòs a la taula esmorzars suma la quantitat introduïda al valor que ja tenia el producte. Tot seguit ho comunica amb un missatge.
- Al tancar la pestanya et pregunta si vols sortir sense guardar o vols guardar les dades.
- L’aplicació te una icona personalitzada la qual també podem observar des de la “barra de tareas”
- He implementat un boto d’exportacio de les dades el qual exporta totes les dades de la taula a un fixer .txt amb un format si llegible.

Solució de problemes

A diferencia de l’anterior projecte si m’he trobat força problemes que han comportat dedicació d’hores per solventar-ho. Alguns, degut al poc temps que manava quedant per entregar el projecte he hagut de no implementar aquella opció, com per exemple les col·leccions que no he fet una supercoleccio com a tal i he fet que l’objecte es convertixque directament en una unica i ja esta. Tampoc he implementat el RandomAccessFile al Fitxers.java però casibé ni ho he intentat ja que s’hem ficava el temps damunt.

També recordo que al començament hem va costar adaptar el projecte UF3 al patró MVC però amb l’ajuda del projecte del github ho he anat fent. Molts d’errors de que no trobava x classe perquè era publica o privada o era un mètode.. que es van acabar solucionant.

Conclusió

Penso que ha estat un projecte que en respecte al de la UF3 m'he vist una altra versió de mi mateix en quan a la programació, mes fluida i entenent molt millor els conceptes, segueixo pensant, igual que a la UF3, que hem faltaria algo de ordenació sobre els mètodes o classes ja que en moments no sabia en quin fitxer era el mes correcte per ubicar-los.

Sincerament al començament no ho hem veia capaç de poder-ho fer i potser aquelles setmanes d'incertesa i de no saber per on començar van ser el detonant per a que a dia d'entregar-lo necessiti alguns dies mes per fer allò que m'he deixat o realitzar ampliacions tal com vaig fer al de la UF3. Finalment hem vaig envalentonar i vaig anar al fang passant el meu propi projecte de la UF3 a MVC, un cop fet això la negativitat s'esvaïa i veia mes clar el final del túnel. A partir d'aqui anar fase x fase recolzant-me de la documentació del moodle o del que s'anava fent a classe i tot acabava sortint be.

Com a millores penso que haguera estat be donar un projecte verge per on començar, per veure també un codi diferent al que tu ja havies fet a l'anterior projecte i també per partir tots d'una base.
També penso que seria mes correcte el determinar les fases del projecte ja des d'un inici i estructurar-lo d'alguna millor manera ja que en el cas de les excepcions es una cosa que conforme vas ficant codi ten surten cada cop mes per implementar i no pot ser que les excepcions estiguin situades a la fase 3 penso que haurien d'estar a l'ultima.
De totes maneres això nomes ha estat per posar alguna pega no poden haver moltes millores a fer si el projecte ha estat realitzat i sense codis d'error.

Dono per finalitzat el projecte amb moltes implementacions o funcionalitats extres per fer però amb un nivell superior amb el que vaig començar.