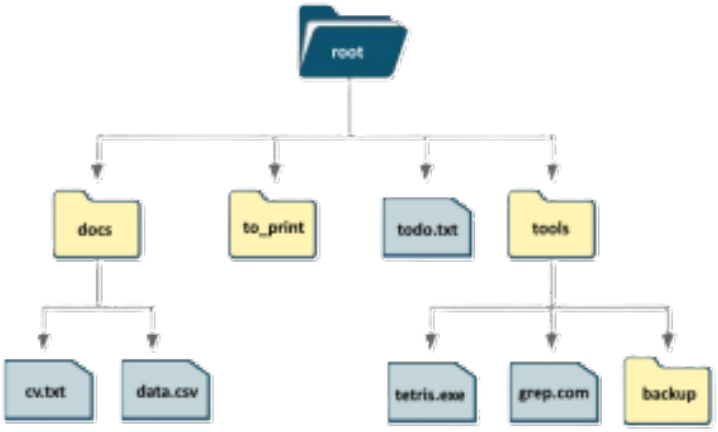
POLITECNICO SETUBAL

Programação avançada (PA)

Relatório de projeto “Personal File System Management”



Docente: André Sanguinetti

Projeto realizado por:

Bernardo Vaz 202200278

Alessandro Aguiar 202200272

Diogo Braizinha 202200223

João Oliveira 202200191

Índice

Apresentação da aplicação desenvolvida

Objetivos

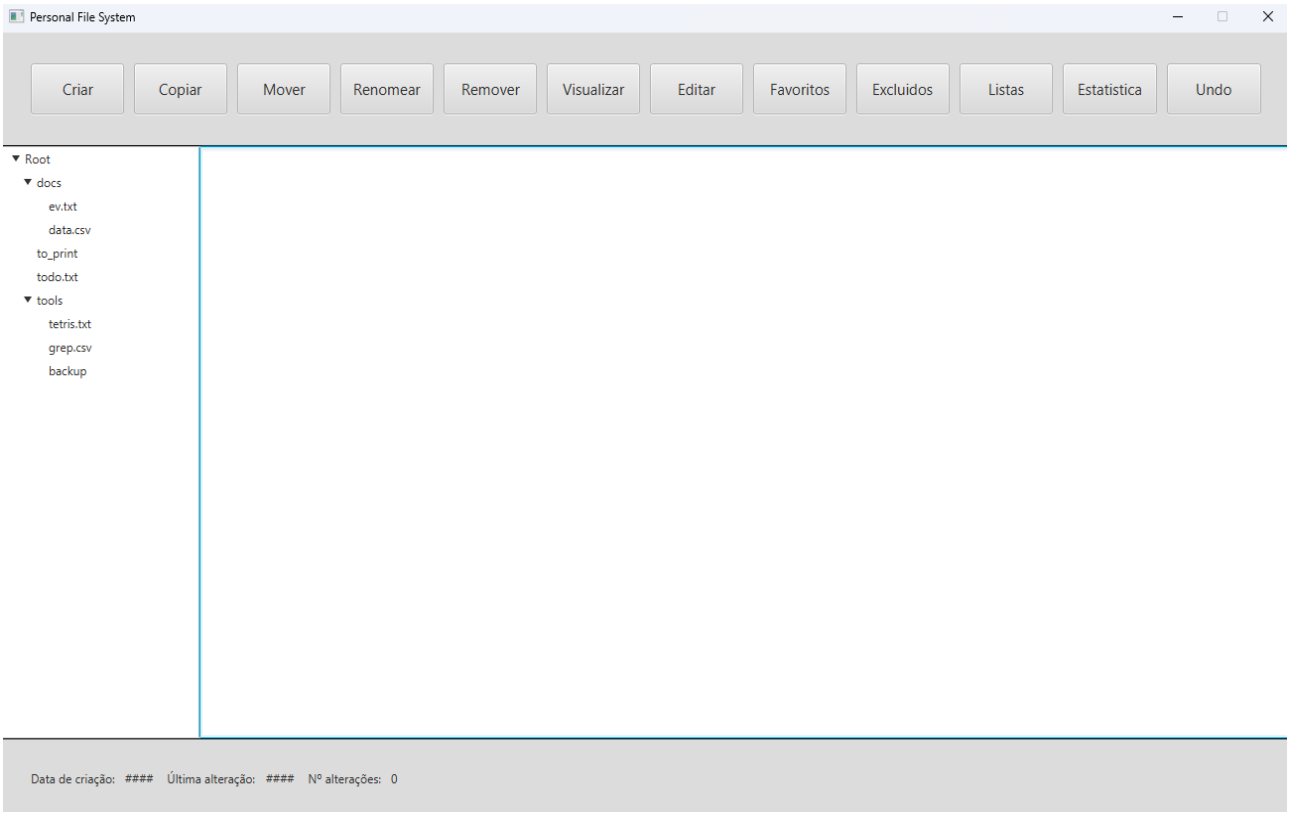
Este projeto desenvolvido pelo nosso grupo, tinha como objetivo simular uma gestão de sistemas de ficheiros chamado “PersonalFileSystem”, através do desenvolvimento de uma aplicação gráfica em Java FX tendo por base o ADT Tree.

Principais funcionalidades

Este sistema desenvolvido será capaz de algumas operações básicas, como a criação de um novo elemento seja ele ficheiro ou diretoria, a operação de copiar em que o que acontece e criar um novo elemento com todo o conteúdo do elemento de origem, a funcionalidade de mover um elemento de uma diretoria para outra, conseguir alterar o nome de um elemento através da funcionalidade de renomear, a operação de remover um elemento e por fim também será capaz de se visualizar o conteúdo de um elemento e editar o mesmo caso o estado esteja “unlocked” através das operações de visualizar e editar respetivamente.

O nosso grupo tem também 2 funcionalidades extras. Uma que permite adicionar um certo ficheiro ou diretoria a uma lista de favoritos, e outra que permite a visualização de uma lista de removidos, simulando a “reciclagem” dos computadores. Por último implementamos a funcionalidade do “undo”.

Modo de utilização



Muito sucintamente, vamos agora descrever como utilizar a aplicação sendo que quando corremos este é o ecrã inicial. Ao clicar no botão criar irá abrir uma janela onde se seleciona se queremos criar um ficheiro ou diretoria e conforme a escolha abrirá ainda uma segunda janela para inserir os dados necessários para a criação do dado elemento escolhido.

O botão copiar abre uma janela onde se seleciona o elemento pretendido para copiar e o destino de onde se pretende colocar essa copia, sendo este modo de funcionamento muito similar ao botão mover que tem as mesmas características com a diferença que invés de apenas copiar, move definitivamente o elemento para o destino escolhido.

O botão renomear abre uma janela em que selecionamos o elemento a renomear e inserimos o novo nome. Já no remover ainda é mais simples e apenas selecionamos o elemento que pretendemos remover.

O visualizar a mesma coisa, abre uma janela, selecionamos o elemento a visualizar e o texto irá aparecer no "texto area" ao lado da árvore.

Agora para o editar, o processo é o mesmo que renomear a diferença é que invés de alterar o nome, alteramos o conteúdo.

Para os favoritos, selecionamos na árvore o elemento que pretendemos colocar na lista de favoritos e clicamos no botão de favorito, o que fará com que o elemento seja marcado como favorito e visualizado na consola a lista atual dos favoritos, também é possível desmarcar selecionando o mesmo elemento e voltando a clicar no favorito.

Nos excluídos o que acontece é que aparece uma lista com os elementos eliminados numa nova janela existindo a possibilidade de restaurar o elemento selecionado.

Nas listas, ao clicar abre uma nova janela com 4 botões sendo eles os descendentes diretos que fornece essa informação depois de selecionar o elemento de partida para esse botão. Os ficheiros alterados, que mostra na text área a lista com os ficheiros alterados e a quantidade de vezes que o mesmo foi alterado. O botão último ficheiros que mostra os últimos ficheiros criados e por fim os últimos ficheiros alterados, que tal como o próprio nome indica mostra os últimos ficheiros modificados.

Depois nas estatísticas irá abrir uma nova janela com 4 botões, começando pelo espaço ocupado em que irá mostrar na text área a percentagem de espaço que cada elemento ocupado de uma dada diretoria. Teremos também um botão chamado Diretorias e ficheiros totais, que mostrará o número de ficheiros e diretorias existentes numa dada diretoria selecionada. O botão seguinte, mostrará a profundidade de um dado elemento selecionado em relação a root. Por fim o último botão desta janela mostrará o top 5 das diretorias com mais descendentes.

O botão undo como dito na introdução as funcionalidades, fará com que se volte um passo atrás na aplicação, bastando clicar nele.

Implementação da aplicação

Arquitetura lógica

Nesta aplicação que desenvolvemos foram criadas 1 interface, 1 enumerado e 6 classes normais. Para começar decidimos criar a interface "FileSystemElement" que iria servir de apoio ás duas classes que a iriam implementar (são elas o "File" e a "Directory") contendo os métodos comuns a ambas. O enumerado "FileType" apenas serve para que o "File" na sua construção tenha um dos tipos presentes no enumerado.

A classe "PersonalFileSystem" é a classe que fica responsável basicamente pela implementação das operações todas que vai ser possível realizar na aplicação, que vai estar desta forma dependente das classes do "File" e da "Directory" devido a necessitar dos construtores/métodos de ambas já que vai realizar operações sobre elementos que possam ser de estes dois tipos.

Passando agora a classe "PFSView", esta é a responsável pela parte gráfica da aplicação sendo que está diretamente ligada a "PersonalFileSystem" para conseguir mostrar as funcionalidades como já mencionado, implementadas na mesma.

Agora falando da classe "DateNameFileSystemElement", esta apenas tem o propósito de ajudar o "PersonalFileSystem" a comparar os atributos das datas dos ficheiros.

Por fim temos uma classe de testes "TestPersonalFileSystem", sendo que obviamente serve para realizar os testes.

Tipos abstratos de dados

Como tipos abstratos de dados usamos o ADT Tree já que nos foi fornecido, já que é com base nela que funciona a aplicação, pois cada ficheiro/diretoria irá ser um elemento que terá um pai, e poderá ter filhos, tal e qual uma árvore.

Padrões de software

Memento- O padrão memento é um padrão que serve para guardar "estados" anteriores de um objeto, ou seja significa isto que podemos utilizar este padrão de modo a que quando fosse realizada alguma operação sobre o nosso PersonalFileSystem pudéssemos guardar o estado da nossa árvore para que desse modo fosse possível realizar a funcionalidade de undo, já que a mesma vai acumulando numa stack os estados sucessivos da árvore permitindo mais tarde se necessário, regressar aos tais estados anteriores. Sendo que o originator é a classe "PersonalFileSystem", o caretaker é a nossa classe "Caretaker", e o memento é a nossa inner class "Memento".

Observer- O observer é um padrão que é utilizado para quando existe alguma alteração num dado objeto, sendo que no nosso caso o "PFSView" notifica o "PersonalFileSystem" para que este execute os métodos desejados.

Refactoring

Tipo “Bad-smell”	Ocorrências	Técnica refactoring
Long Method	2	Extract method
Duplicated code	3	Extract method
Primitive Obsession	1	Extract class

- Long method (Antes)

```
360         private void copyElemButton() {
361             copyElem.setOnAction(event -> {
362                 Stage stageCopy = new Stage();
363                 stageCopy.setTitle("Copiar Diretório/Ficheiro");
364                 Label fileSystem = new Label( text: "Selecione o Diretório/Ficheiro:");
365                 ComboBox<FileSystemElement> elementChoose = new ComboBox<>(this.elements);
366                 Label dirChoose = new Label( text: "Selecione a diretoria:");
367                 ComboBox<Directory> parentsChoose = new ComboBox<>(directories);
368 
369                 Button copy = new Button( text: "Copiar");
370 
371                 copy.setOnAction(ev -> {
372                     caretaker.saveMemento();
373 
374                     FileSystemElement copied = elementChoose.getValue();
375                     Directory parent = parentsChoose.getValue();
376 
377                     if(copied != null && parent != null) {
378                         pfs.copyElement(copied, parent);
379 
380                         changeDetailsPanel();
381 
382                         pfsView.getRoot().getChildren().clear();
383                         buildTreeView(pfsView.getRoot(), pfs.root());
384 
385                         // Fechar a janela de copia de arquivo
386                         ((Stage) stageCopy.getScene().getWindow()).close();
387                     } else {
388                         alertWarnings( text: "Nenhum ficheiro/diretoria foi selecionado! \nDestino não foi selecionado!");
389                     }
390                 });
391                 VBox vbox = new VBox( spacing: 20);
392                 vbox.setPadding(new Insets( topRightBottomLeft: 20));
393                 vbox.getChildren().addAll(fileSystem, elementChoose, dirChoose, parentsChoose, copy);
394                 Scene scene = new Scene(vbox,  width: 300,  height: 250);
395                 stageCopy.setScene(scene);
396                 stageCopy.showAndWait();
397                 refreshComboBox();
398             });
399         }
```

- Long method (Depois)

```
412     private void copyElemButton() {
413         copyElem.setOnAction(event -> handleCopyElement());
414     }
        1 usage  AlessandroA70 +1 *
415     private void handleCopyElement() {
416         Stage stageCopy = createCopyStage();
417         ComboBox<FileSystemElement> elementChoose = createElementComboBox();
418         ComboBox<Directory> parentsChoose = createDirectoryComboBox();
419         Button copy = createCopyButton(stageCopy, elementChoose, parentsChoose);
420         copy.setOnAction(ev -> handleCopyAction(stageCopy, elementChoose, parentsChoose));
421         VBox vbox = createCopyVBox(elementChoose, parentsChoose, copy);
422         Scene scene = new Scene(vbox,  width: 300,  height: 250);
423         stageCopy.setScene(scene);
424         stageCopy.showAndWait();
425         refreshComboBox();
426     }
        1 usage  AlessandroA70 +1
427 @   private Stage createCopyStage() {
428         Stage stageCopy = new Stage();
429         stageCopy.setTitle("Copiar Diretório/Ficheiro");
430         return stageCopy;
431     }
        4 usages  AlessandroA70 +1
432 @   private ComboBox<FileSystemElement> createElementComboBox() {
433         ComboBox<FileSystemElement> elementChoose = new ComboBox<>(this.elements);
434         return elementChoose;
435     }
        2 usages  AlessandroA70 +1
436 @   private ComboBox<Directory> createDirectoryComboBox() {
437         ComboBox<Directory> parentsChoose = new ComboBox<>(directories);
438         return parentsChoose;
439     }
        1 usage  AlessandroA70 +1
440 @   private Button createCopyButton(Stage stageCopy, ComboBox<FileSystemElement> elementChoose, ComboBox<Directory> parentsChoose) {
441         Button copy = new Button( text: "Copiar");
442         return copy;
443     }
        1 usage  DiogoBLourenco +3 *
444 @   private void handleCopyAction(Stage stageCopy, ComboBox<FileSystemElement> elementChoose, ComboBox<Directory> parentsChoose) {
445         caretaker.saveMemento();
446         FileSystemElement copied = elementChoose.getValue();
447         Directory parent = parentsChoose.getValue();
448         if (copied != null && parent != null) {
449             pfs.copyElement(copied, parent);
450             changeDetailsPanel();
451             pfsView.getRoot().getChildren().clear();
452             buildTreeView(pfsView.getRoot(), pfs.root());
453             ((Stage) stageCopy.getScene().getWindow()).close();
454         } else {
455             alertWarnings( text: "Nenhum ficheiro/diretoria foi selecionado! \nDestino não foi selecionado!");
456         }
457     }
        1 usage  AlessandroA70 +2 *
458 @   private VBox createCopyVBox(ComboBox<FileSystemElement> elementChoose, ComboBox<Directory> parentsChoose, Button copy) {
459         Label fileSystem = new Label( text: "Selecione o Diretório/Ficheiro:");
460         Label dirChoose = new Label( text: "Selecione a diretoria:");
461         VBox vbox = new VBox( spacing: 20);
462         vbox.setPadding(new Insets( topRightBottomLeft: 20));
463         vbox.getChildren().addAll(fileSystem, elementChoose, dirChoose, parentsChoose, copy);
464         return vbox;
465     }
```

- Primitive obsession (Antes)

  Antes tínhamos utilizado um atributo do tipo String na classe File que indicava se era “txt” ou “csv”, e para solucionar alteramos o tipo do atributo para FileType que sendo ele um enumerado, só permite de extensão txt, ou csv.

  Nota: Não tiramos foto ao antes e então não nos foi possível mostrar, daí termos explicado por palavras.

- Primitive obsession (Depois)

- Duplicated code (Antes)

  Antes tínhamos dois métodos iguais em que apenas diferiam no sentido em que um servia para o File, e outro para Directory.

- Duplicated code (Depois)

Gestão do projeto

Avaliação critica

Era pretendido desenvolver um programa que simulasse um sistema de gestão de ficheiros que realizasse várias funcionalidades e que ainda tivesse persistência de dados. Após o esforço de todos conseguimos desenvolver uma aplicação que achamos resolver os problemas propostos com exceção da persistência de dados, que infelizmente não conseguimos desenvolver, de resto achamos que realizamos um trabalho que cumpre o proposto.

Enumeração do trabalho realizado

Na enumeração do trabalho realizado por cada um não nos achamos capazes de o dividir pois felizmente todos desenvolvemos em conjunto o projeto todo, ou seja, ao longo do tempo fomos sempre fazendo tudo em conjunto contribuindo quase que por igual em todos os aspetos.