

**Joanã Costa 180 221 109**

**Daniel Cordeiro 190 221 101**

**PROGRAMAÇÃO AVANÇADA**

**WebCrawler**



Índice

[Índice de Figuras 3](#_Toc30609631)

[1. Introdução 4](#_Toc30609632)

[2. Tipos Abstratos de Dados 4](#_Toc30609633)

[3. Diagrama de Classes 5](#_Toc30609634)

[4. Documentação de Classes 7](#_Toc30609635)

[CareTaker 7](#_Toc30609636)

[Factories 7](#_Toc30609637)

[HomeController 7](#_Toc30609638)

[Home View 7](#_Toc30609639)

[Home View.StopCriteria 8](#_Toc30609640)

[Link 8](#_Toc30609641)

[LoggerWriter 8](#_Toc30609642)

[Main 8](#_Toc30609643)

[MyDigraph 8](#_Toc30609644)

[MyEdge 8](#_Toc30609645)

[MyVertex 8](#_Toc30609646)

[SearchDepth 8](#_Toc30609647)

[SearchExpandedPages 8](#_Toc30609648)

[SearchIterative 8](#_Toc30609649)

[SearchPages 8](#_Toc30609650)

[WebCrawler 9](#_Toc30609651)

[WebCrawlerJson 9](#_Toc30609652)

[WebCrawlerFile 9](#_Toc30609653)

[WebPage 9](#_Toc30609654)

[5. Padrões de Software 10](#_Toc30609655)

[MVC e Observer 10](#_Toc30609656)

[DAO (Data Access Object) 11](#_Toc30609657)

[Strategy 11](#_Toc30609658)

[Simple Factory 11](#_Toc30609659)

[Singleton 12](#_Toc30609660)

[Memento 12](#_Toc30609661)

[6. Refactoring 13](#_Toc30609662)

[7. Conclusão 18](#_Toc30609663)

# Índice de Figuras

[Figura 1 - Diagrama de Classes 7](file:///D:\Engenharia\Ano%203\PA\PROJETO\Relatório.docx#_Toc30610151)

[Figura 2 - MVC 10](file:///D:\Engenharia\Ano%203\PA\PROJETO\Relatório.docx#_Toc30610152)

[Figura 3 - Classes dos tipos de ficheiros 11](file:///D:\Engenharia\Ano%203\PA\PROJETO\Relatório.docx#_Toc30610153)

[Figura 4 - Representação das Estratégias 11](#_Toc30610154)

[Figura 5 - Factory 11](#_Toc30610155)

[Figura 6 - Classe Singlenton 12](#_Toc30610156)

[Figura 7 - Implementação Memento 12](#_Toc30610157)

[Figura 8 - Implementação da inner class no modelo 12](#_Toc30610158)

# Introdução

Este projeto tem como objetivo a utilização da TAD ***Graph*** e ***Queue*** na resolução do problema de percorrer páginas *web* e extrair demais páginas associadas. Sendo assim será desenvolvido um ***WebCrawler***que fará essa tarefa através dos algoritmos ***breadth-fisrt*** e eventualmente o algoritmo **Djisktra**.

Dada um link de uma página web o pograma deverá ser capaz de fazer uma representação gráfica de todas as páginas incidentes a mesmas, possibilitando abrir estas mesmas páginas no browser e interação com o utilizador onde deverá possibilitar o utilizador carregar nessas respetivas páginas e mostrar todos os links incidentes desta.

# Tipos Abstratos de Dados

Para este projeto usamos alguns tipos para facilitar a manipulação dos dados.

Esses são os tipos de dados usados:

* ***HashMap*** – Usados para guardar os vértices e as arestas na classe ***MyDigraph***;
* ***ArrayLists*** – Usados para guardar vários tipos de dados ao longo do código;
* ***Queue*** – Usado para armazenar em memória os objetos das páginas Web para que sejam retiradas por ordem e processadas as mesmas;

Tipos abstratos de dados implementados:

* ***MyDigraph*** – Tipo de dados que representa um grafo, usado para representar as páginas web que tem ligações com outras.
* ***MyVertex*** – Tipo de dados que representa um vértice do grafo;
* ***MyEdge*** – Tipo de dados que representa uma aresta do grafo;

# Diagrama de Classes

Na figura que se segue estão representadas as classes utilizadas na 1º fase do desenvolvimento da aplicação do ***WebCrawler*** e os acoplamentos entre elas. (Ver Imagem).

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 1 - Diagrama de Classes

# Documentação de Classes

## CareTaker

Essa classe gerência as solicitações para salvar e restaurar os estados do objeto (WebCrawler).

## Factories

Essa classe é responsável por criar a estrutura MVC e os tipos de arquivo a serem exportados.

## HomeController

Esta classe é responsável por gerenciar toda a lógica de negócio entre o modelo e a vista.

## Home View

Esta classe contém toda a implementação sobre a interface do usuário. Implementa um comportamento da visualização e um observador para manter-se atualizado quando o modelo for alterado.

## Home View.StopCriteria

Enumerado que identifica os tipos de pesquisas durante o programa. Os valores são: **PAGES**, **DEPTH**, **ITERATIVE** e **EXPANDED**;

## Link

Classe que representa um link de uma página web onde este será guardado no grafo.

## LoggerWriter

Essa classe é responsável por gravar mensagens em um arquivo de texto

## Main

Está classe é ponto principal de entrada do programa.

## MyDigraph

Classe que implementa um grafo. Para além dos métodos normais de criação e manipulação de um grafo existem também métodos para retornar vértices aleatórios.

## MyEdge

Classe privada usada em MyDiGraph que implementa uma aresta do grafo. Contém um elemento e o vértice de início e fim.

## MyVertex

Classe privada usada em MyDiGraph que implementa um vértice do grafo. Contém um elemento.

## SearchDepth

Esta classe é uma estratégia para pesquisar páginas pelo modo interativo.

## SearchExpandedPages

Esta é outra classe de critérios de parada de estratégia para pesquisar nas páginas da Web.

## SearchIterative

Esta classe é uma estratégia para pesquisar páginas pelo modo interativo.

## SearchPages

Esta classe é uma estratégia para pesquisar todas as páginas em largura-primeiro.

## WebCrawler

Esta classe é responsável para criar o nosso modelo do WebCrawler baseado-se na classe MyDigraph implementado, onde os vértices são representados pelas páginas web em concreto (classe WebPage) e arestas por um link desta página (classe Link).

## WebCrawlerJson

Esta classe é para gerar um novo arquivo do tipo ***JSON*** pela serialização do java. Implementa uma interface que faz operações para salvar e carregar o arquivo pretendido

## WebCrawlerFile

Esta classe é para gerar um novo arquivo do tipo texto pela serialização do java. Implementa uma interface que faz operações para salvar e carregar o arquivo pretendido.

## WebPage

Classe que representa o vértice no gráfico. Está classe é representação de uma página web em concreto.

# Padrões de Software

Durante o decorrer da UC foram lecionados alguns padrões de software dos quais usados foram:

## MVC e Observer

A estrutura do nosso projeto está baseada neste padrão devido a sua robustez e distribuição de responsabilidade entre classes, cada classe é responsável pelo que lhe foi imputado. Sendo assim temos três packages no nosso projeto, **Model-Views-Controller**, está representado numa figura abaixo a estrutura no ***NetBeans***.

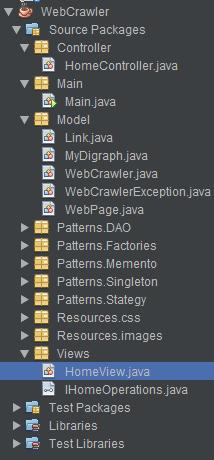


Figura 2 - MVC

## DAO (Data Access Object)

Como no nosso projeto precisaríamos exportar ficheiros de texto e em JSON, tivemos que implementar este padrão para esses dois tipos de ficheiro. Este está representado na classe ***WebCrawlerFile*** e ***WebCrawlerJson*** onde é usado o modelo (classe ***WebCrawler***).



Figura 3 - Classes dos tipos de ficheiros

## Strategy

O padrão strategy é usado para alternar entre os critérios de paragem no modo iterativo e nos modos automáticos.

No modo automático temos três critérios de paragem sendo eles:

* Páginas visitadas – Termina assim que n páginas forem visitadas;
* Profundidade – Termina assim que todas as páginas de que distam M links da página inicial;
* Uma página que tem mais do que n links, não será mais expandida;

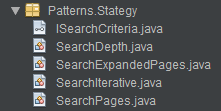


Figura 4 - Representação das Estratégias

## Simple Factory

Para este padrão usamos para criar objetos pertencente ao padrão MVC e os tipos de ficheiro juntamente com o DAO. As classes usadas são a ***Main, WebCrawler e WePage***.



Figura 5 - Factory

## 

## Singleton

Para este padrão, usamos para fazer os ***logs*** por ficheiro de texto do programa. Este é chamado em quase todos as classes do programa para notificar erros que ocorram durante a execução do programa e para escrever as páginas web que foram inseridos durante a pesquisa das mesmas.

Classes Usadas:

* WebCrawler
* WebPage
* WebCrawlerFile
* WebCrawlerJson
* SearchPages
* SearchIterative
* SearchDepth



Figura 6 - Classe Singlenton

## Memento

Apesar de não termos conseguido implementar de forma perfeita no nosso projeto, este também está presente na classe principal de modelo do ***WebCrawler***, onde o objetivo era ir guardando os estados do grafo quando a página era expandida e permitir fazer o undo dessa ação se o utilizador quisesse.

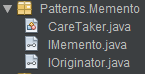


Figura 7 - Implementação Memento

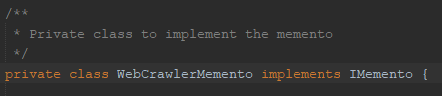


Figura 8 - Implementação da inner class no modelo

# Refactoring

Nesta parte, vamos começar a fazer toda busca por código que não esteja bem ao longo das classes do projeto (***bad smells***) por isso usamos técnicas que irão melhorar o aspeto do código fazendo com que outro programador ao ver o código seja mais fácil de ler o código dai chamado o ***Refactoring***.

|  |  |
| --- | --- |
| Bad Smell | Duplicated Code |
| Técnica de Refactoring | Extract Method |
| Código Antigo |  |
| Código Resultante |  |

|  |  |
| --- | --- |
| Bad Smell | Data Clump |
| Técnica de Refactoring | Extract Class + Hide delegate |
| Código Antigo |  |
| Código Resultante | Na classe WebCrawler:  Uma imagem com monitor, texto  Descrição gerada automaticamente |

|  |  |
| --- | --- |
| Bad Smell | Innappropriate Intimacy |
| Técnica de Refactoring | Hide Delegate |
| Código Antigo |  |
| Código Resultante |  |

|  |  |
| --- | --- |
| Bad Smell | Large Class |
| Técnica de Refactoring | Extract SubClass |
| Código Antigo |  |
| Código Resultante |  |

|  |  |
| --- | --- |
| Bad Smell | Makink Method Calls Simpler |
| Técnica de Refactoring | Rename method |
| Código Antigo |  |
| Código Resultante |  |

# Conclusão

Neste projeto conseguimos absorver todos os conhecimentos lecionados durante este semestre, desde os TAD’s, padrões de software a refactoring.

**Rafael Trindade - 160210033**

**Vítor Gomes - 160210050**

Com importância mais dos padrões de software onde o principal ponto é detetar padrões a partir de problemas encontrados durante o desenvolvimento do programa e aplicar o padrão mais indicado.

E por fim também temos o refactoring outro ponto importante no desenvolvimento de software dado que melhora a leitura do código para outro programador e também por consequência deixa o software mais robusto e estruturado e eventualmente melhoria do desempenho do programa.