IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Programação Avançada

2019/2020

Licenciatura em Engª. Informática

Relatório Final

Turma: SW-07
Docente: Professora Patrícia Macedo
nº180221017 Nome Afonso Cunha
nº180221019 Nome Marco Pereira

1. TADs implementados

O único TAD (Tipo Abstrato de Dados) implementado foi o Grafo mais propriamente um Grafo Orientado.

Em matemática e, mais especificamente, no contexto de Teoria dos Grafos, um grafo orientado é um grafo que consiste num conjunto de vértices, estes ligados por arestas que contêm uma direção associada.

No nosso projeto o grafo orientado vem como resposta ao problema de queremos representar sites e as suas hiperligações a outros sites ou páginas web, sendo que no contexto da nossa aplicação os sites serão os nossos vértices e as hiperligações com outras páginas serão as arestas.

2. Diagrama de Classes

O Diagrama de Classes foi criado usando um plugin do IDE Netbeans chamado easyUML. Tendo em conta a quantidade de classes existentes, removemos do diagrama as classes fornecidas pelos professores. Ainda assim, a qualidade da imagem não ficou a desejável, sendo recomendável aceder à imagem através do link em baixo e fazer zoom caso não seja legível o suficiente.

Link do diagrama: https://imgur.com/a/FSU0zAx

3. Documentação de classes

A documentação de classe como era baseada no javadoc decidimos não colocar no relatório para não o encher de informação já existente. Para ter acesso à documentação de classes (javadoc) basta aceder à pasta "**dist**" situada dentro da diretoria do projeto e aceder onde diz **index**

4. Padrões utilizados

- Observer
  - Conceito
    - O Observer é um padrão comportamental que permite que um objeto possa notificar outros de uma mudança de estado ou de eventos que tenham ocorrido no objeto
  - Uso
    - Após a leitura e análise do enunciado do projeto, deparámos com a necessidade da existência de um mecanismo que atualizasse a visualização do grafo (GraphView) quando houvesse alguma alteração no grafo
  - Mapeamento
    - Subject: WebScraper
    - Observer: GraphView
- Memento
  - Conceito
    - O padrão Memento é utilizado quando se quer guardar um estado de um objeto, sem violar o encapsulamento do mesmo, para que seja possível que o objeto possa voltar a esse estado em qualquer altura
  - Uso
    - Sendo a possibilidade de undo um dos requisitos do projeto, o padrão Memento surge como resposta a este problema. Assim, é possível guardar o estado de um objeto e restaurar um estado já guardado
  - Mapeamento
    - Originator: GraphView
    - Memento: Memento
    - Caretaker: GraphViewCareTaker

- DAO
  - Conceito
    - O padrão DAO é usado para abstrair e encapsular todos os acessos à fonte de dados independente do tipo que esta é (ex: base de dados, ficheiro json, ficheiro XML). O DAO funciona como um adaptador entre o componente e a fonte de dados

  - Uso
    - Um dos requisitos do projeto é a persistência de um grafo, assim podendo carregar um grafo que tenha sido previamente criado. Como solução a este problema surgiu o padrão DAO que nos permite fazer a implementação de diversas formas de guardar ficheiros

  - Mapeamento
    - AbstractDAO: GraphDao
    - Client: DirectGraph
    - ConcreteDAO: GraphDaoFile, GraphDaoJson
    - Data Source: Serializable, Gson

- Simple Factory
  - Conceito
    - O padrão Factory é utilizado quando se tem várias classes que implementam uma interface (ou existe hierarquia de classes), e se pretende delegar responsabilidades para criar um objeto num outro, encapsulando as diversas implementações da interface
  - Uso
    - Com a existência do padrão DAO no nosso projeto, o padrão Factory surge como auxílio na delegação de cada ConcreteProduct.
  - Mapeamento
    - Product: GraphDao
    - Factory: DaoFactory
    - ConcreteProduct: GraphDaoFile, GraphDaoJson

- Strategy
  - Conceito
    - O padrão Strategy é um padrão comportamental que permite a definição de vários algoritmos, colocá-los cada um em sua classe e tornar os seus objetos intercambiáveis
  - Uso
    - Na nossa aplicação, a necessidade da aplicação deste padrão surge com a possibilidade de diferentes comportamentos do WebScrapper
  - Mapeamento
    - Context: WebScrapperAutomatic
    - Strategy: IteractionStrategy
    - ConcreteStrategy: DephtCriteria, LinksCriteria, VisitedPagesCriteria

5. Bad Smells

Nome	Número de Ocorrências	Técnicas de refactoring utilizadas
Long Class	1	Extract Class
Long Method	2	Extract Method
Duplicated Code	2	Pull Up Method
Message Chain	3	Estes bad smells não foram corrigidos pois ou faziam parte das classes fornecidas pelos professores ou eram métodos de classes genéricas onde era necessário ser daquela forma
Data Class	3	Este bad smell não foi corrigido, em nenhuma das 3 ocasiões encontradas. Esta decisão deve-se ao facto de que estas classe não porem em risco a estrutura do projeto
Lazy Class	1	Inline Class

- Bad smells (Localizações)
  - Long Method
    - Métodos do WebScrapperAutomatic –PagesToVertex
    - Métodos do WebScrapperIterative - PagesToVertex
  - Long Class
    - GraphView
  - Duplicated Code
    - Métodos PagesToVertex das classes WebScrapperAutomatic e WebScrapperIterative
  - Message Chain
    - Todos os message chains foram encontrados na classe GraphView, esta que dá uso ao código fornecido pelos professores
  - Data Class
    - Connection
    - WebPage
    - GraphViewMemento
  - Lazy Class
    - Chart

6. Exemplos de soluções de bad smells

- Long Class (Extract Class)
  - Foi criada uma nova classe com um único atributo: uma listView

```
*/
public SideList() { listView = new ListView<>(); }
```

- Long Method (Extract method) e Duplicated Code (Pull Up Method)
  - Foi criado um novo método – createLinkVertex – e puxado para a classe pai, assim, resolvendo dois bad smells

```
public Vertex<Webpage> createLinkVertex(Graph<Webpage, Connection> graph, Webpage newWebpage){
    Vertex<Webpage> linkVertice = null;

    try {
        linkVertice = graph.insertVertex(newWebpage);
        observerMethods();
    } catch (InvalidVertexException ignored) {
        for (Vertex<Webpage> webpage : graph.vertices()){
            if (webpage.element().equals(newWebpage)){
                linkVertice = webpage;
            }
        }
    }

    return linkVertice;
}
```

- Lazy Class (Inline Class)
    - A classe Chart foi apagada porque só tinha um método e foi criado um método que servia o propósito da classe na classe Statistics

```
private BarChart<String, Number> createBarChart(DirectGraph<Webpage, Connection> g) {
    CategoryAxis xAxis = new CategoryAxis();
    NumberAxis yAxis = new NumberAxis();
    yAxis.setLabel("Number of references");
    xAxis.setLabel("Website title");

    BarChart<String,Number> bc =
            new BarChart<>(xAxis, yAxis);

    XYChart.Series<String, Number> series1 = new XYChart.Series<>();

    int i = 0;

    for (Map.Entry<Vertex<Webpage>, Integer> entry : g.returnTop5().entrySet()) {
        if (i != 5){
            i++;
            series1.getData().add(new XYChart.Data<>(entry.getKey().element().getTitle(), entry.getValue()));
        }
    }

    bc.getData().add(series1);

    return bc;
}
```