IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

PA 2021/22

Programação Avançada

Enunciado de Laboratório

5

Laboratório – ADT Graph E utilização API SmarGraph

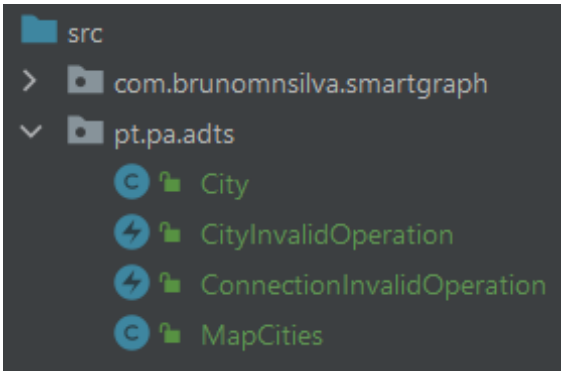
Objetivos:

- Utilização do ADT Graph;
- Desenvolvimento de testes unitários
- Utilizar a API *SmartGraph* para visualizar e interagir com um grafo.

Introdução:

Comece por clonar o projeto existente no GitHub no seguinte endereço:
https://github.com/estsetubal-pa-geral/Lab5Template.git

Verifique que já existem criadas as classes presentes na figura:



O objetivo do laboratório é fazer a implementação da classe **MapCities** que utiliza o ADT Graph<V, E> para implementar um mapa de distâncias entre cidades (City).

NOTA: Todos os métodos seguintes deverão ser implementados invocando métodos já implementados na classe base utilizada (não é necessário fazer qualquer alteração na implementação da classe).

Nível 1 – Implementação dos métodos que fornecem informação sobre a classe

Crie os seguintes métodos auxiliares **privados** na classe MapCities:

- **Vertex<City> findCity(String city)**
  Devolve o vértice onde a cidade *city* se encontra ou null, c.c.

- **boolean existCity(String city)**
  Verifica se a cidade *city* existe (como vértice).

- **Edge<Integer, City> findConnection(String sourceCity, String destinationCity)**
  `throws ConnectionInvalidOperation`
  Devolve a aresta que liga os vértices *sourceCity* e *destinationCity* ou *null*. Caso alguma das cidades não exista é lançada uma exceção.

- **boolean existConnection(String sourceCity, String destinationCity)**
  Verifica se existe a ligação (aresta) entre *sourceCity* e *destinationCity*.

Nível 2 – Implementar os métodos para adicionar e remover elementos no grafo

Crie os seguintes métodos públicos:

- **`void addCity(String city)`**
  `throws CityInvalidOperation`
  Insere uma nova cidade no mapa (um novo vértice). Se a cidade já existir é lançada uma exceção.

- **`Vertex<City> removeCity(String city)`**
  `throws CityInvalidOperation`
  Remove e devolve a cidade enviada ao método. Se a cidade não existir é lançada a exceção.

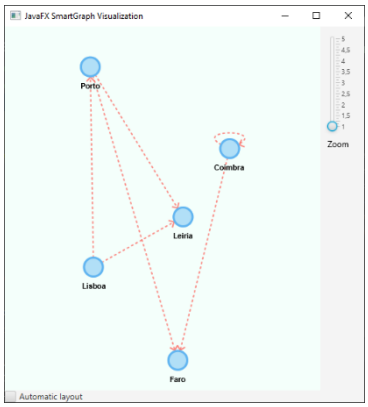
- **`void addConnection(String sourceCity, String destinationCity, int distance)`**
  `throws ConnectionInvalidOperation`
  Adiciona ao grafo a ligação entre duas cidades (aresta entre dois vértices). Se a ligação entre as cidades já existir é lançada a exceção.

- **`void removeConnection(String sourceCity, String destinationCity)`**
  `throws ConnectionInvalidOperation`
  Remove do grafo a ligação entre duas cidades. Se a ligação não existir é lançada a exceção.

Nível 3 – Visualização do grafo usando SmartGraph

Altere o método **build_sample_map** e adicione as cidades e as distâncias indicadas como comentário no método.

```
Porto Leiria 185
Lisboa Porto 300
Coimbra Coimbra 10
Lisboa Leiria 130
Coimbra Faro 255
Porto Faro 550
```



Altere a propriedade edge.label no ficheiro smartgraph.properties para ver no mapa as distâncias entre as cidades.

Nível 4 – Implementar métodos adicionais e testes unitários relacionados

Crie os seguintes métodos públicos:

- **`boolean isIsolated(String city) throws CityInvalidOperation`**
  Verifica se uma cidade está isolada, isto é, se existe no grafo, mas não tem ligações a qualquer outra cidade (nem a ela própria).

- **`Collection<City> neighbors(String city) throws CityInvalidOperation`**
  Devolve a lista de cidades "vizinhas" de uma determinada cidade.

Comece por criar o método **setup** de modo a disponibilizar uma configuração inicial a cada um dos testes unitários.

De seguida implemente os seguintes testes unitários:

- **`isIsolated_isCorrect_AfterInsertRemoveCitiesAndConnections`**
  Verifica se:
  - Um vértice depois de criado é considerado isolado;
  - Um vértice depois de ligado a outro não é considerado isolado;
  - Um vértice ligado a outro volta a ser considerado isolado se for removida a única ligação entre eles;
  - Um vértice com >=2 ligações a outros vértices não é isolado depois de remover uma das ligações.

- **`addConnection_isCorrect_whenSourceIsEqualToDestination`**
  Verifica se se pode fazer uma ligação de uma cidade a ela própria, isto é, se não é lançada qualquer a exceção.

- **`neighbors_isCorrect_afterInsertAndRemoveVerticesAndEdges`**
  Verifica se a lista de cidades vizinhas é gerada corretamente, isto é:
  - Se X for uma cidade sem ligações, então não tem vizinhos;
  - Se X->Y, então Y faz parte da lista de vizinhos de X
  - Se X->Y e se removermos a ligação entre X e Y, então Y não faz parte da lista de vizinhos de X;
  - Se X->Y, se removermos Y e voltarmos a adicionar a cidade Y (mas não a ligação), então Y não faz parte da lista de vizinhos de X.

  Na implementação da UT utilize a função auxiliar **exists**.

```
private boolean exists(Iterable<City> list, String city) {
    for(City item: list)
        if (item.element().equals(city))
            return true;
    return false;
}
```

5 Implementação do método *load*

Complete a implementação do método **load** da classe que permite fazer o carregamento das cidades e respetivas ligações a partir de um ficheiro de dados (texto).

As cidades e as ligações devem ser criadas dinamicamente através dos métodos existentes na classe. A criação dos elementos deverá ser feita apenas se o elemento não existir, seja ele cidade ou ligação.

A estrutura do ficheiro é a seguinte:

Linha de dados (Exemplo)	Descrição
`Abrantes`	Uma palavra apenas – Nome da **cidade** (vértice apenas)
`Faro 2`	Duas palavras apenas - Uma **cidade** e a **distância** para si própria (A ligação a implementar será **origem**: Faro | **destino**: Faro | **distância**: 2)
`Viseu Coimbra 95`	Três palavras – Cidade de **origem,** cidade de **destino** e a **distância** entre elas**.** (A ligação a implementar será **origem**: Viseu | **destino**: Coimbra | **distância**: 95)
Nota: Existe apenas um espaço entre cada elemento numa linha de dados.	

Aplique o método *load* ao ficheiro **data/demo-graph.txt** disponibilizado no projeto para fazer o carregamento automático de dados de um mapa e reescreva o carregamento dos dados a apresentar no *SmartGraph* a partir desse ficheiro.