Para importar um Grafo (ponderado ou não) a partir de múltiplos arquivos CSV, utiliza-se o método *importGraphMultipleCSV* da classe *ImportUtil* do pacote *util*. Para tal, deve-se dispor dos parâmetros detalhados na implementação genérica abaixo:

```
ImportUtil.importGraphMultipleCSV
(Graph<DefaultVertex, RelationshipWeightedEdge> graph,
String Vfilename, String idAtt, String labelAtt, String Efilename,
String sourceAtt, String targetAtt, String weightAtt, boolean simplegraph,
boolean weighted)
```

onde graph é uma instância da interface Graph com *generics* declarado como a seguir: Graph<DefaultVertex,RelationshipWeightedEdge>. *Vfilename* é a localização (*FilePath*) do sistema de arquivos no qual se encontra o arquivo CSV que especifica os atributos que descrevem os vértices, *idAtt* é o nome do atributo em *Vfilename* que representa o *id* de cada vértice, *labelAtt* é o nome do atributo em *Vfilename* que representa o *label* de cada vértice, *Efilename* é a localização (*FilePath*) do sistema de arquivos no qual se encontra o arquivo CSV que especifica os atributos que descrevem as arestas, *sourceAtt* é o nome do atributo em *Efilename* que representa um terminal de uma aresta, *targetAtt* é o nome do atributo em *Efilename* que representa o outro terminal de uma aresta, *weightAtt* é nome do atributo em *Efilename* que especifíca o peso (caso exista) de cada aresta, *simplegraph* é um valor booleano que determina se o grafo é simples, e *weighted* é um valor booleano que determina se o grafo é ponderado.
Por exemplo, dentre as classes que implementam a interface *Graph*, pode-se utilizar a classe *SimpleGraph* para instanciar o grafo simples abaixo:

```
Graph<DefaultVertex, RelationshipWeightedEdge> simpleGraph =
        new SimpleGraph <>
                (VertexEdgeUtil.createDefaultVertexSupplier(),
                VertexEdgeUtil.createRelationshipWeightedEdgeSupplier(),false);
```

Em seguida, pode-se utilizar os dados das Tabelas 1 e 2 (representativas de relações de amizades em uma rede social hipotética), para criar os vértices e arestas de um grafo não ponderando mediante a importação detalhada abaixo:

```
ImportUtil.importGraphMultipleCSV(unweightedgraph,
        graphpathname + "pessoas.csv","Login","Nome",
         graphpathname + "amizades.csv", "Pessoa1", "Pessoa2", null, true,
false);
```

Adicionalmente, pode-se utilizar os dados das Tabelas 1 e 2, para criar os vértices e arestas de um grafo ponderado mediante a importação detalhada abaixo (onde o atributo Tempo na Tabela 2 especifica o tempo de amizade em anos entre duas pessoas):

```
ImportUtil.importGraphMultipleCSV(weightedgraph,
        graphpathname + "pessoas.csv","Login","Nome",
        graphpathname + "amizades.csv", "Pessoa1", "Pessoa2", "Tempo", true,
true);
```

Tabela 1 - Arquivo CSV que detalha os vértices representativos de pessoas em uma rede social hipotética.

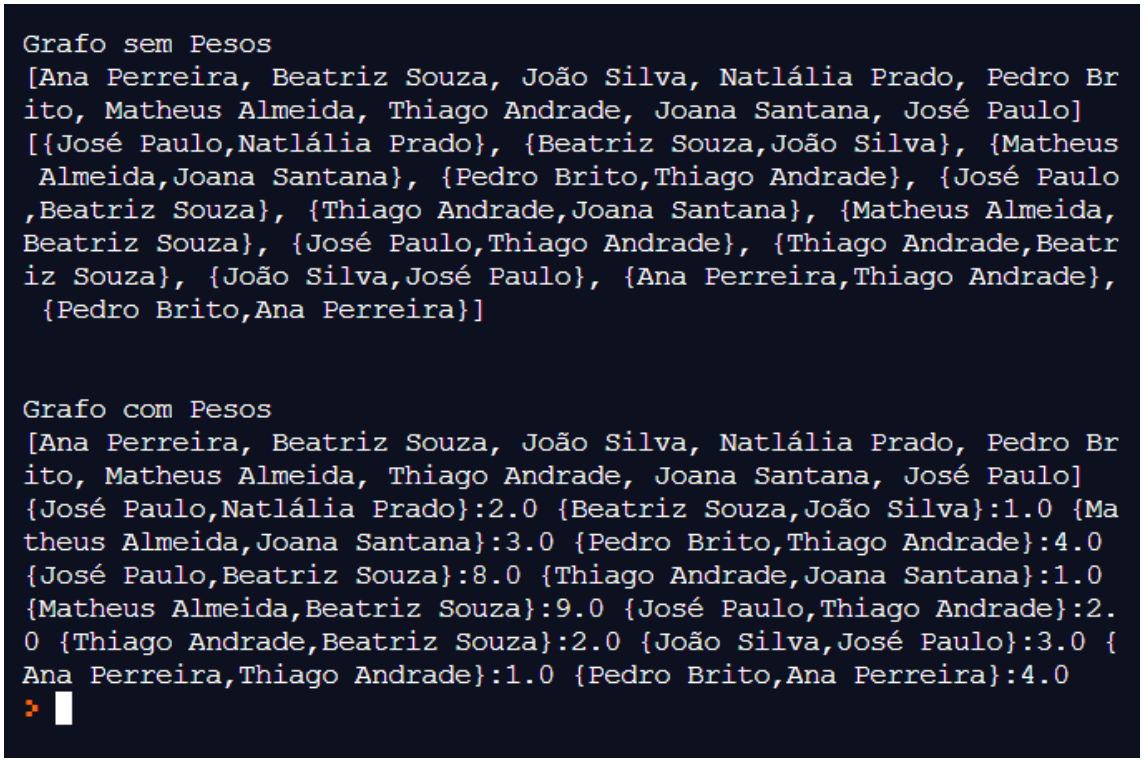
pessoas.csv
Login,Nome,Cidade,Estado
AninhaP,Ana Perreira,Aracaju,Sergipe
BiaSouza,Beatriz Souza,Santos,São Paulo
JoaoS,João Silva,Salvador,Bahia
NatPd,Natlália Prado,Belo Horizonte,Minas Gerais
PedBr,Pedro Brito,Fortaleza,Ceará
MatheusAl,Matheus Almeida,Porto Seguro,Bahia
ThiagoAd,Thiago Andrade,Sousa,Paraíba
JoanaS,Joana Santana,Crato,Ceará
JoseP,José Paulo,Campina Grande,Paraíba

Tabela 2 - Arquivo CSV que detalha as arestas representativas de relações de amizades, em uma rede social hipotética, entre as pessoas da Tabela 1.

amizades.csv
ID,Pessoa1,Pessoa2,Tempo
1,JoseP,NatPd,2
2,BiaSouza,JoaoS,1
3,MatheusAl,JoanaS,3
4,PedBr,ThiagoAd,4
5,JoseP,BiaSouza,8
6,ThiagoAd,JoanaS,1
7,MatheusAl,BiaSouza,9
8,JoseP,ThiagoAd,2
9,ThiagoAd,BiaSouza,2
10,JoaoS,JoseP,3
11,AninhaP,ThiagoAd,1
12,PedBr,AninhaP,4

Caso seja necessário acessar os demais atributos de vértices e arestas, pode-se utilizar, respectivamente, o método getAtt da classe DefaultVertex, e o método getAtt da classe RelationshipWeightedEdge. Por exemplo, para um vértice v, representativo de uma pessoa, pode-se obter o valor do atributo Estado com v.getAtt("Estado"). Para uma aresta e, representativa de uma relação de amizade, pode-se obter o valor do atributo Tempo com e.getAtt("Tempo").
Por fim, ao implementar a codificação descrita acima na JgraphT, pode-se utilizar o método *printGraph* da classe *util.PrintUtil* para obter a impressão disposta na Figura 1 a seguir:

Figura 1 – Impressão de grafos gerados a partir dos dados das Tabelas 1 e 2.



Atividade realizada por mim, Adísio Pereira Fialho Júnior - 120111047