Aula 03 – JGraphT

Notas de Aula de Teoria dos Grafos

Profª.: Patrícia D. L. Machado

UFCG – Unidade Acadêmica de Sistemas e Computação

Sumário

Introdução

JGraphT é uma biblioteca em Java que dá suporte a implementação de diferentes tipos de grafos e inclui a implementação de vários algoritmos para solução de problemas usando grafos. Está disponível em código aberto no **GitHub**[1]. É organizada em pacotes que agrupam conceitos de teoria dos grafos ou utilitários. A documentação está disponível no formato Javadocs[2].

Trata-se de uma implementação flexível, no sentido de que qualquer objeto pode ser usado para representar vértices e diferentes classes podem ser definidas para representar arestas. Além disso, provê a segurança de tipos através do conceito de ***generics***. Utiliza diferentes ***iterators*** para caminhar sobre o grafo e adaptadores para visualização gráfica. Foi projetada para apresentar bom desempenho com velocidade comparável a aplicações nativas em muitos casos. Tem sido amplamente utilizada por uma extensa comunidade ativa.

[1] https://github.com/jgrapht/jgrapht
[2] https://jgrapht.org/javadoc/

Repositório GraphTheory-JGraphT

O repositório **GraphTheory-JGraphT**, hospedado no GitHub, foi criado para auxiliar em atividades de programação desta disciplina usando a JGraphT. Possui classes básicas que exemplificam a implementação de conceitos vistos nas aulas e auxiliam na realização dos exercícios práticos da disciplina. Consiste em 4 pacotes básicos:

- **classexamples** - Código com exemplos de uso dos recursos para explorar conceitos vistos em cada aula da disciplina. Arquivos são nomeados Aula## para indicar a aula e o conceito explorado no código.
- **datasets** - bases de dados utilizadas em exercícios avançados da disciplina.
- **graphs** - diversos grafos que podem ser utilizados para testar os códigos do pacote classexamples.
- **util** - classes utilitárias que implementam funções auxiliares para a construção de programas usando a JGraphT sem se preocupar com detalhes mais complexos.

O pacote **util** possui as classes descritas na tabela abaixo (o uso de cada um destes pacotes e suas classes será explicado neste e em outros roteiros de laboratório para a JGraphT).

Classe	Descrição
DefaultVertex	Implementa um vértice com identificador e um conjunto de atributos, onde um deles pode ser o *label* do vértice.
DrawUtil	Fornece métodos para visualização gráfica de grafos.
ExportUtil	Fornece métodos para exportação de grafos no formato GML.
ImportUtil	Fornece métodos para importação de grafos em diferentes formatos.
MeasureUtil	Fornece métodos que implementam métricas avançadas para análise de grafos.
PrintUtil	Fornece métodos para imprimir grafos em formato textual no console.
RelationshipDirectedEdge	Extende a classe **DefaultEdge** da JGraphT para dar suporte a arcos com múltiplos atributos em grafos direcionados.
RelationshipDirectedWeightedEdge	Extende a classe **DefaultWeightedEdge** da JGraphT para dar suporte a arcos com múltiplos atributos e pesos em grafos direcionados.
RelationshipEdge	Extende a classe **DefaultEdge** da JGraphT para dar suporte a arestas com múltiplos atributos em grafos não-direcionados.
RelationshipWeightedEdge	Extende a classe **DefaultWeightedEdge** da JGraphT para dar suporte a arestas com múltiplos atributos e pesos em grafos não-direcionados.
TreeUtil	Fornece métodos adicionais para Árvores.
VertexEdgeUtil	Fornece métodos auxiliares para acesso de atributos em Vértices e Arestas e *Suppliers* para Vértices e Arestas usados na importação de grafos.

Instalação em IDE

A forma mais simples de instalar a JGraphT é importando o repositório **GraphTheory-JGraphT**[3]. O repositório é um projeto **Eclipse-Maven** que já apresenta as configurações Maven necessárias para importar a JGraphT. É só seguir os passos disponíveis no arquivo README.md do Repositório.

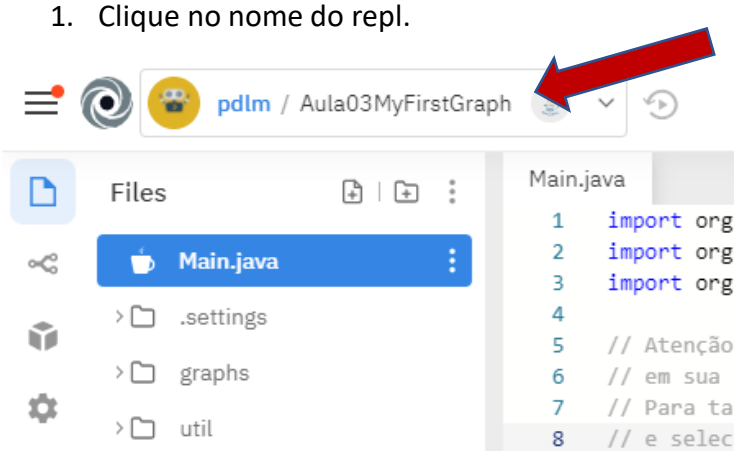
De forma geral, o site da JGraphT (jgrapht.org) apresenta orientações gerais para instalação em diferentes IDEs. Caso não use o Maven, deve baixar a versão mais atual, descompactar em um folder e importar os arquivos .jar como biblioteca externa de seu projeto em IDEs como o Eclipse. Caso use o MAVEN, bastará adicionar uma dependência ao seu projeto. Passos detalhados para instalar, usando ou não o MAVEN, em diferentes IDEs estão disponíveis neste link[4].

Usando a JGraphT no repl.it

Alternativamente, é possível usar a JGraphT sem precisar instalá-la em uma IDE local. Faremos isto usando o site **repl.it**.

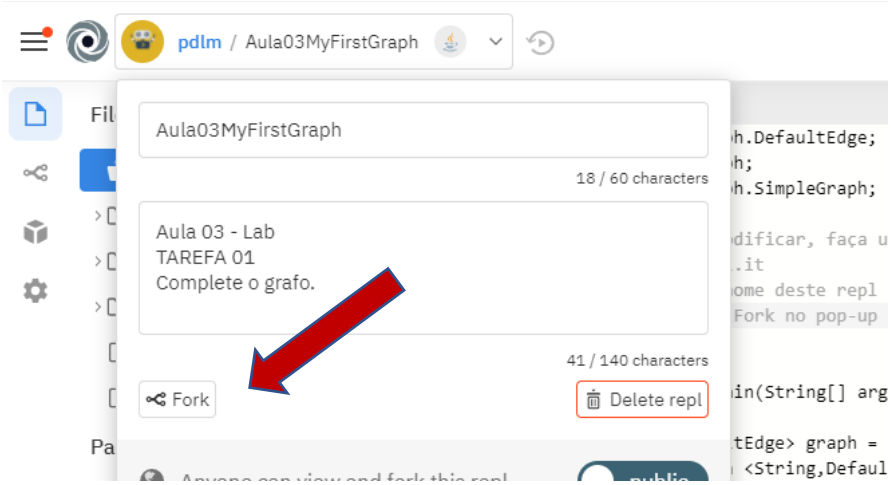
Para todos os exercícios deste e dos próximos laboratórios, será apresentado um **repl** onde o aluno poderá construir o código proposto, já com todos os pacotes necessários importados. Mas, antes de codificar, será necessário *criar uma cópia do repl em sua conta*. Para tal siga este procedimento:

1. Clique no nome do repl.



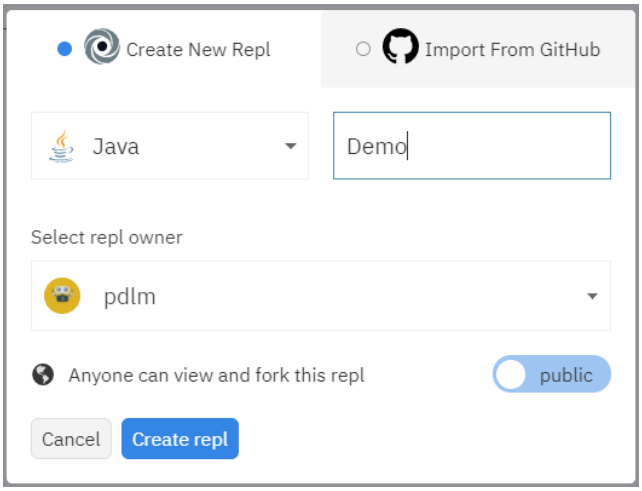
[3] https://github.com/pdlmachado/GraphTheory-JGraphT
[4] https://github.com/jgrapht/jgrapht/wiki/Users:-How-to-use-JGraphT-as-a-dependency-in-your-projects

2. Clique no botão Fork que aparece no pop-up.

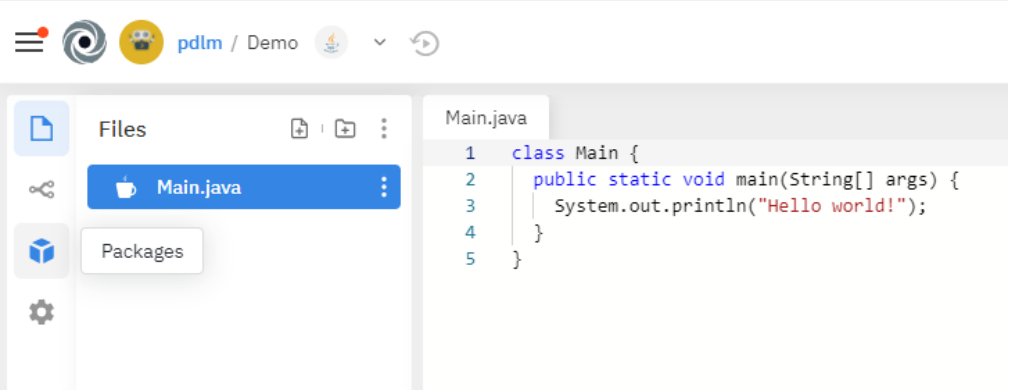


Configurando um repl para usar a JGraphT

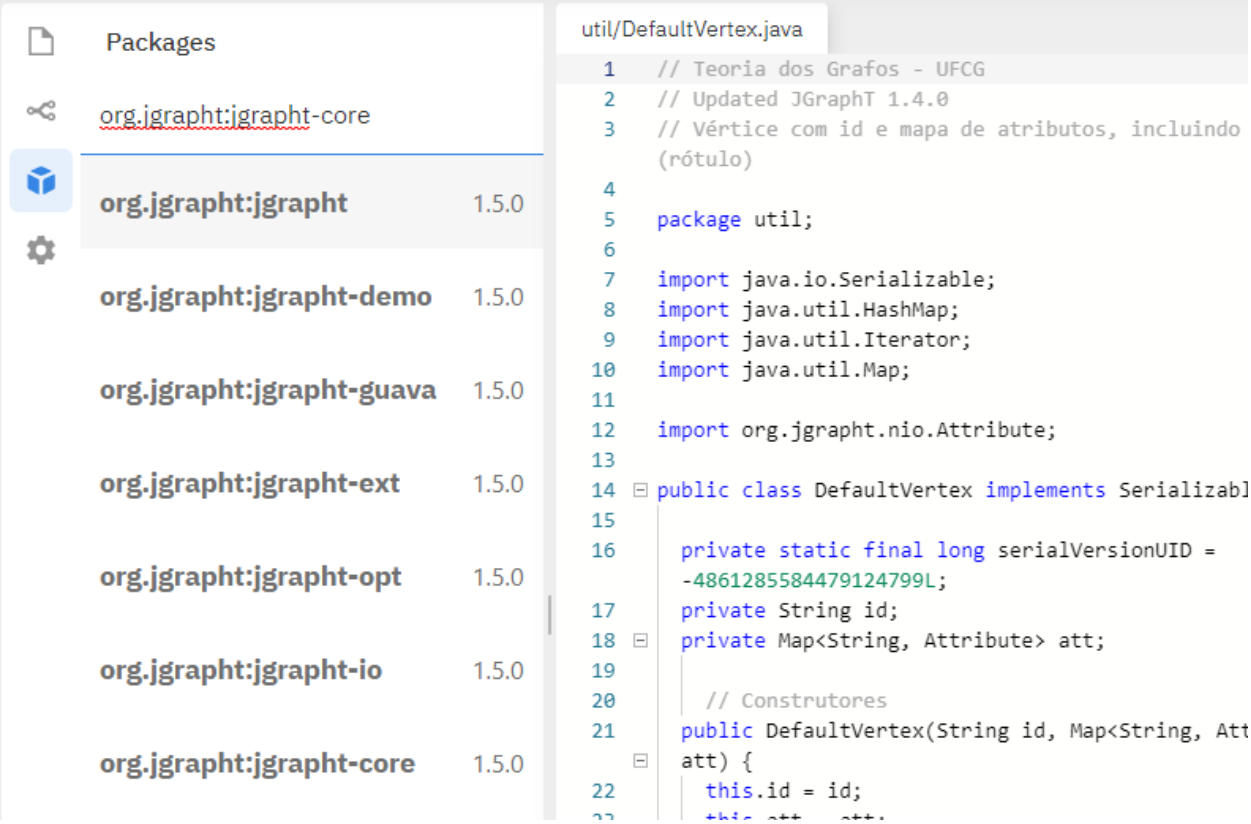
Para criar e configurar seu próprio repl usando a JGraphT. Crie um repl com a linguagem Java.



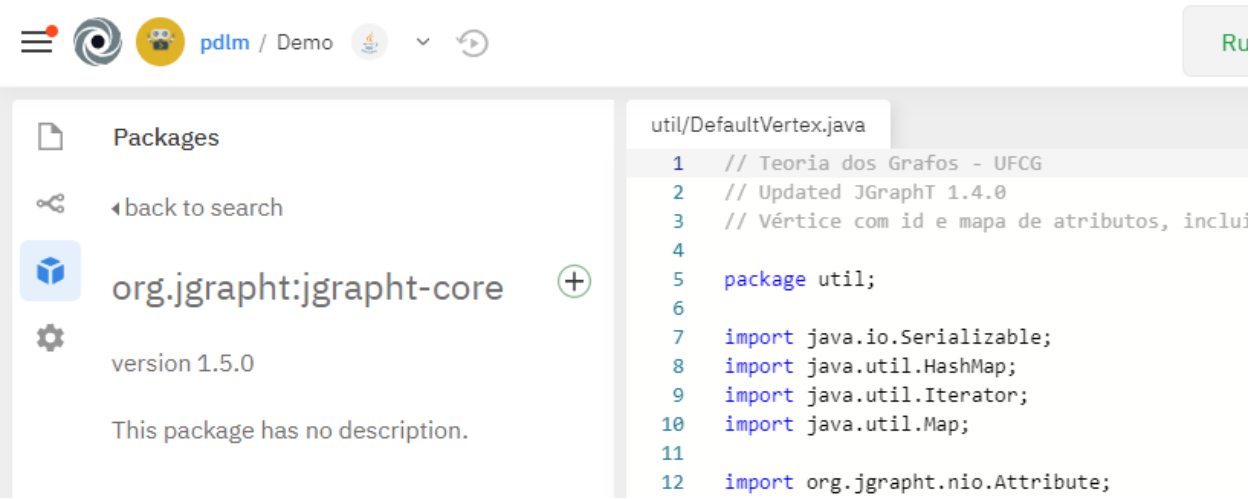
No repl criado, clique no ícone para adicionar pacotes.



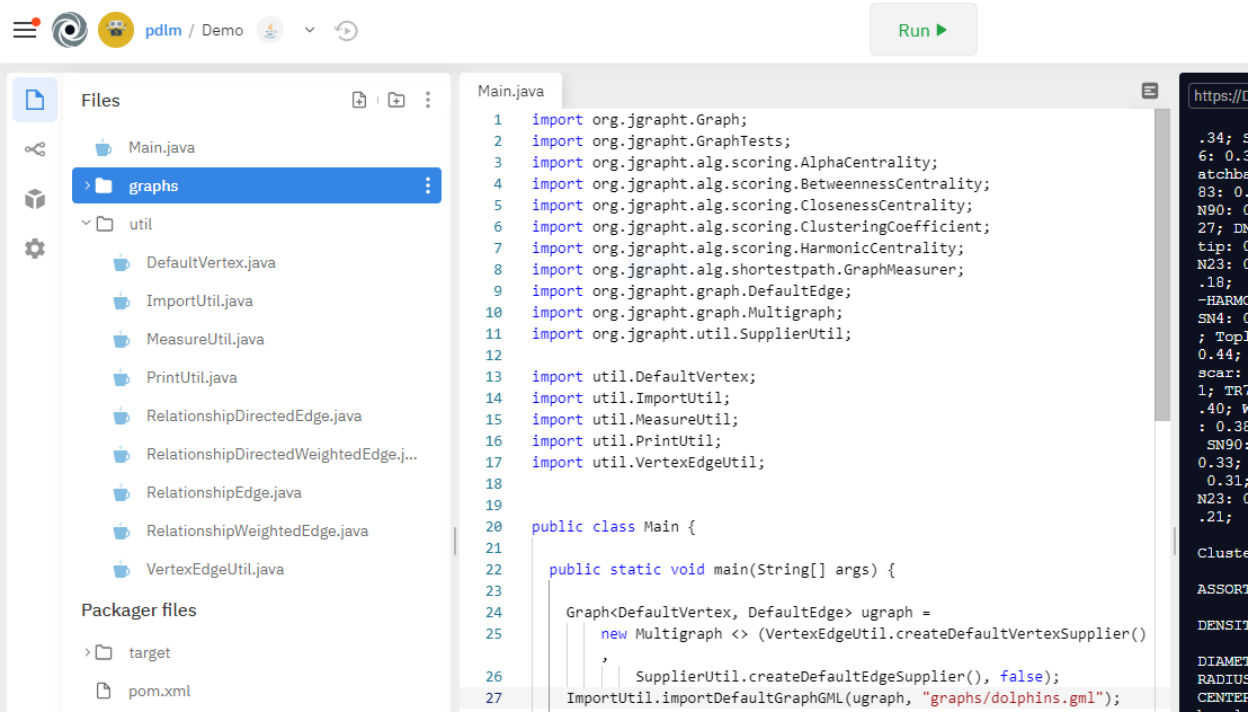
Faça uma busca e selecione o pacote org.jgraph:jgrapht-core, versão 1.5.0.



Adicione o pacote, clicando em +. E depois realize este mesmo procedimento para os pacotes org.jgrapht:jgrapht-io e org.jgrapht:jgraph-ext.



Por fim, realize o upload dos arquivos do repositório GraphTheory-JGraphT, em particular, as classes do pacote util.



Generics na JGraphT

Em Java, uma interface ou classe que pode ser declarada para receber um ou mais parâmetros de tipo escritos entre os símbolos <>. Estes parâmetros devem ser definidos quando uma nova

instância da classe é criada. Por exemplo, no framework de Coleções de Java, a classe ArrayList<*E*> implementa a interface List<*E*>, onde *E* é um parâmetro de tipo. No fragmento de código abaixo, podemos observar que o objeto *words* é uma lista de *strings* que é criada como uma instância da classe ArrayList. O tipo dos elementos da lista – String – é instanciado no momento da criação do objeto.

Interface List<E>
Class ArrayList<E>

```
List<String> words = new ArrayList<String>();
words.add("Hello ");
words.add("world!");
String s = words.get(0)+words.get(1);
assert s.equals("Hello world!");
```

Na JGraphT, classes são parametrizadas pelo tipo dos vértices e o tipo das arestas**.** Em particular**,** **Interface Graph<V,E>** que é implementada por todas as classes básicas de grafo da biblioteca é um *generics* parametrizado pelo tipo associado aos vértices do grafo, tipo **V**, e o tipo associado as arestas do grafo, tipo **E**. Ou seja, para criar um grafo, deveremos indicar que classe implementa vértices e arestas. Dentre as classes que implementam a interface **Graph**, temos **Multigraph**, que dá suporte a criação de multigrafos, a classe **SimpleGraph** que dá suporte a criação de grafos simples e a classe **Pseudograph** que dá suporte a criação de pseudografos.