Politecnico di Torino

e-Lite

The jGraphT library

Tecniche di Programmazione – A.A. 2021/2022

CC BY NC SA

Summary

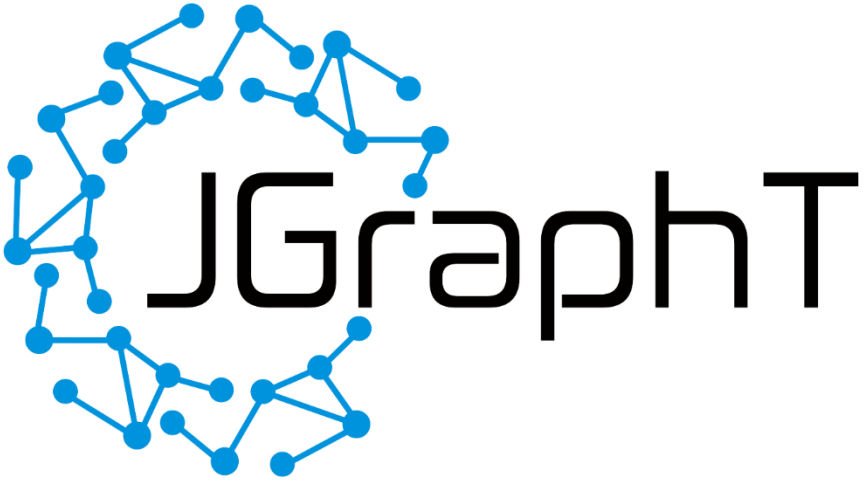
- The JGraphT library
- Creating graphs

Introduction to jGraphT

The jGraphT library

JGraphT

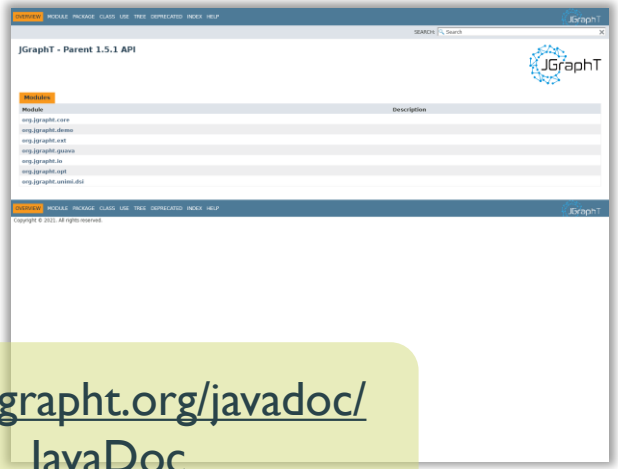
- http://jgrapht.org
  - (do not confuse with jgraph.com)
- Free Java graph library that provides graph objects and algorithms
- Easy, type-safe and extensible thanks to <generics>
- Just add **`jGrapht 1.5.1`** to your project
  - Already in our Maven Archetype



jGraphT Pointers

https://jgrapht.org
Main website

https://github.com/jgrapht/jgrapht
GitHub repository



https://jgrapht.org/javadoc/
JavaDoc

https://jgrapht.org/guide/UserOverview
User Guide

JGraphT structure

https://jgrapht.org/javadoc/

Packages in the module org.jgrapht.core	
org.jgrapht	The front-end API's interfaces and classes, including Graph.
org.jgrapht.alg	Algorithms provided with **JGraphT**.
org.jgrapht.event	Event classes and listener interfaces, used to provide a change notification mechanism on graph modification events.
org.jgrapht.generate	Generators for graphs of various topologies.
org.jgrapht.graph	Implementations of various graphs.
org.jgrapht.graph.builder	Various builder for graphs.
org.jgrapht.graph.specifics	Implementations of specifics for various graph types.
org.jgrapht.traverse	Graph traversal means.
org.jgrapht.util	Non-graph-specific data structures, algorithms, and utilities used by **JGraphT**.

JGraphT algorithms

https://jgrapht.org/javadoc/

Packages **org.jgrapht.alg.**	
.clique	Clique related algorithms.
.clustering	Graph clustering algorithms.
.color	Graph coloring algorithms.
.connectivity	Algorithms dealing with various connectivity aspects.
.cycle	Algorithms related to graph cycles.
.decomposition	Algorithms for computing decompositions.
.flow	Flow related algorithms.
.mincost	Algorithms for minimum cost flow
.independentset	Algorithms for Independent Set in a graph.
.interfaces	Algorithm related interfaces.
.isomorphism	Algorithms for (sub)graph isomorphism.
.lca	Algorithms for computing lowest common ancestors.
.matching	Algorithms for the computation of matchings.
.partition	Algorithm for computing partitions.
.scoring	Vertex and/or edge scoring algorithms.
.shortestpath	Shortest-path related algorithms.
.spanning	Spanning tree and spanner algorithms.
.tour	Graph tours related algorithms.
.transform	Package for graph transformers
.vertexcover	Vertex cover algorithms.

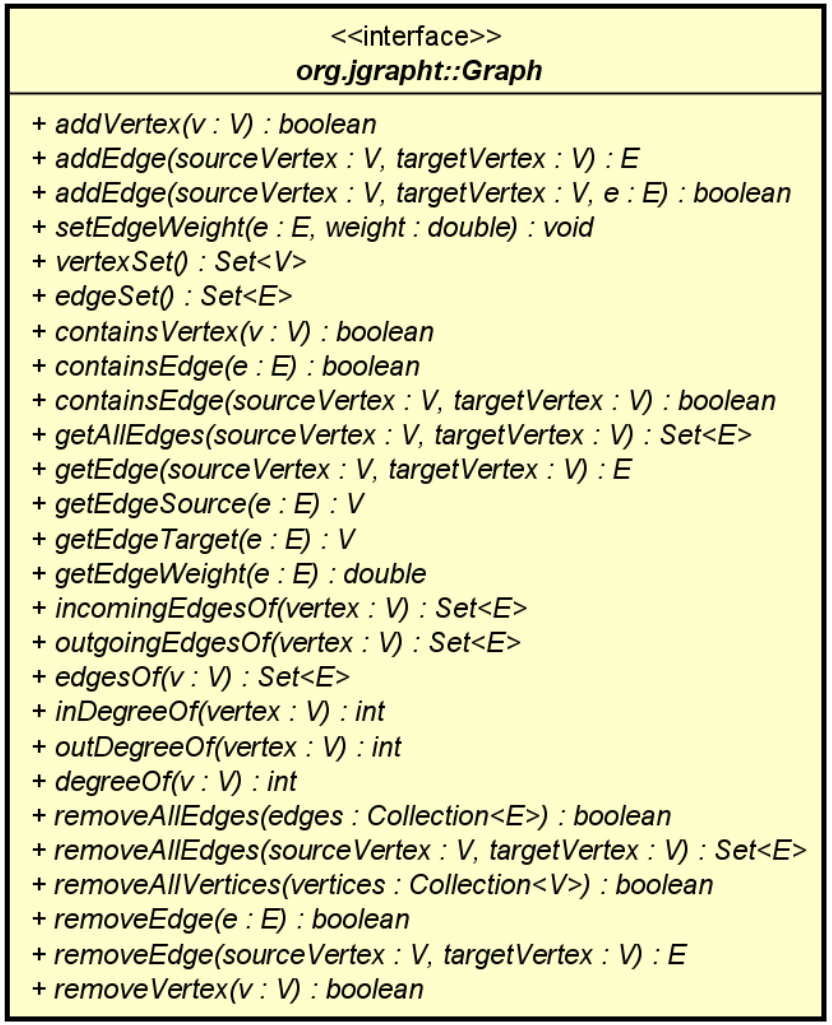
Graph objects

- All graphs derive from:
  - Interface *org.jgrapht.***Graph**<V, E>
- V = type of vertices
  - Any class
- E = type of edges
  - *org.jgrapht.graph.***DefaultEdge**
  - *org.jgrapht.graph.***DefaultWeightedEdge**
  - Your own custom subclass

<V, E>

- User-defined objects, depending on the problem
- Must properly define hashCode and equals
  - The Graph implementation and many graph algorithms use HashSet and HashMap internally!
- Vertex type V
  - Your own object
  - Define hashCode and equals
- Edge type E
  - Subclass of DefaultEdge or DefaultWeightedEdge
  - Do not redefine (override) the provided hashCode and equals

What is a Graph?



Graph classes

org.jgrapht

Graph (I)

org.jgrapht.graph

DefaultDirectedGraph

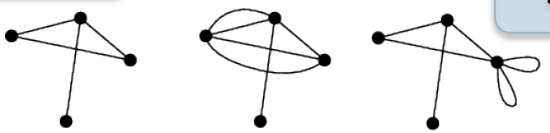
DefaultDirectedWeightedGraph

SimpleGraph

SimpleWeightedGraph

SimpleDirectedGraph

SimpleDirectedWeightedGraph



DirectedMultigraph

Multigraph

DirectedWeightedMultigraph

WeightedMultigraph

DirectedPseudograph

DirectedWeightedPseudograph

Pseudograph

WeightedPseudograph

Graph Implementation Classes

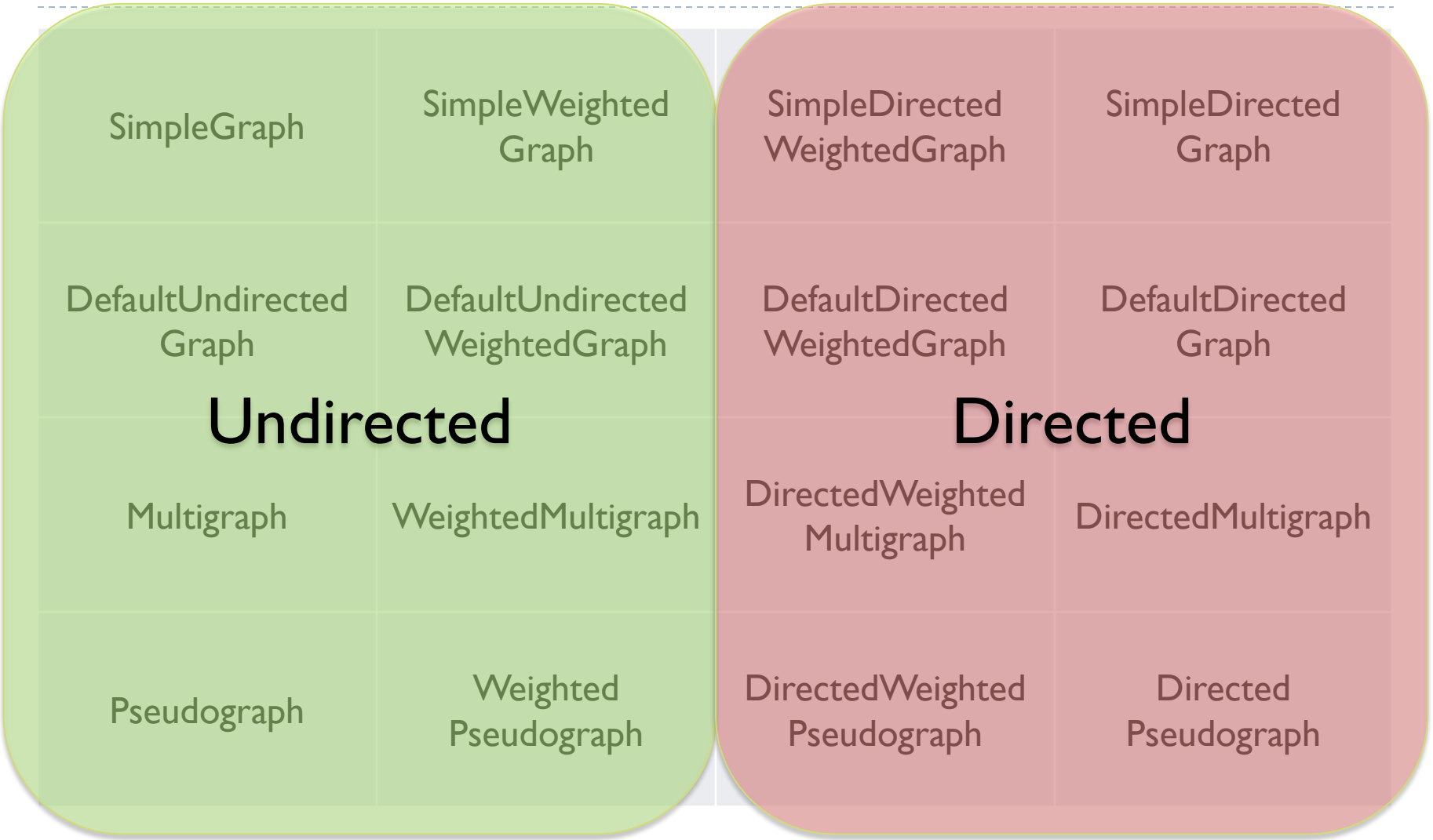
Class Name	Edges	Self-loops	Multiple edges	Weighted
DefaultUndirectedWeightedGraph	undirected	yes	no	yes
SimpleGraph	undirected	no	no	no
Multigraph	undirected	no	yes	no
Pseudograph	undirected	yes	yes	no
DefaultUndirectedGraph	undirected	yes	no	no
SimpleWeightedGraph	undirected	no	no	yes
WeightedMultigraph	undirected	no	yes	yes
WeightedPseudograph	undirected	yes	yes	yes
DefaultUndirectedWeightedGraph	undirected	yes	no	yes
SimpleDirectedGraph	directed	no	no	no
DirectedMultigraph	directed	no	yes	no
DirectedPseudograph	directed	yes	yes	no
DefaultDirectedGraph	directed	yes	no	no
SimpleDirectedWeightedGraph	directed	no	no	yes
DirectedWeightedMultigraph	directed	no	yes	yes
DirectedWeightedPseudograph	directed	yes	yes	yes
DefaultDirectedWeightedGraph	directed	yes	no	yes

https://jgrapht.org/guide/UserOverview#graph-structures

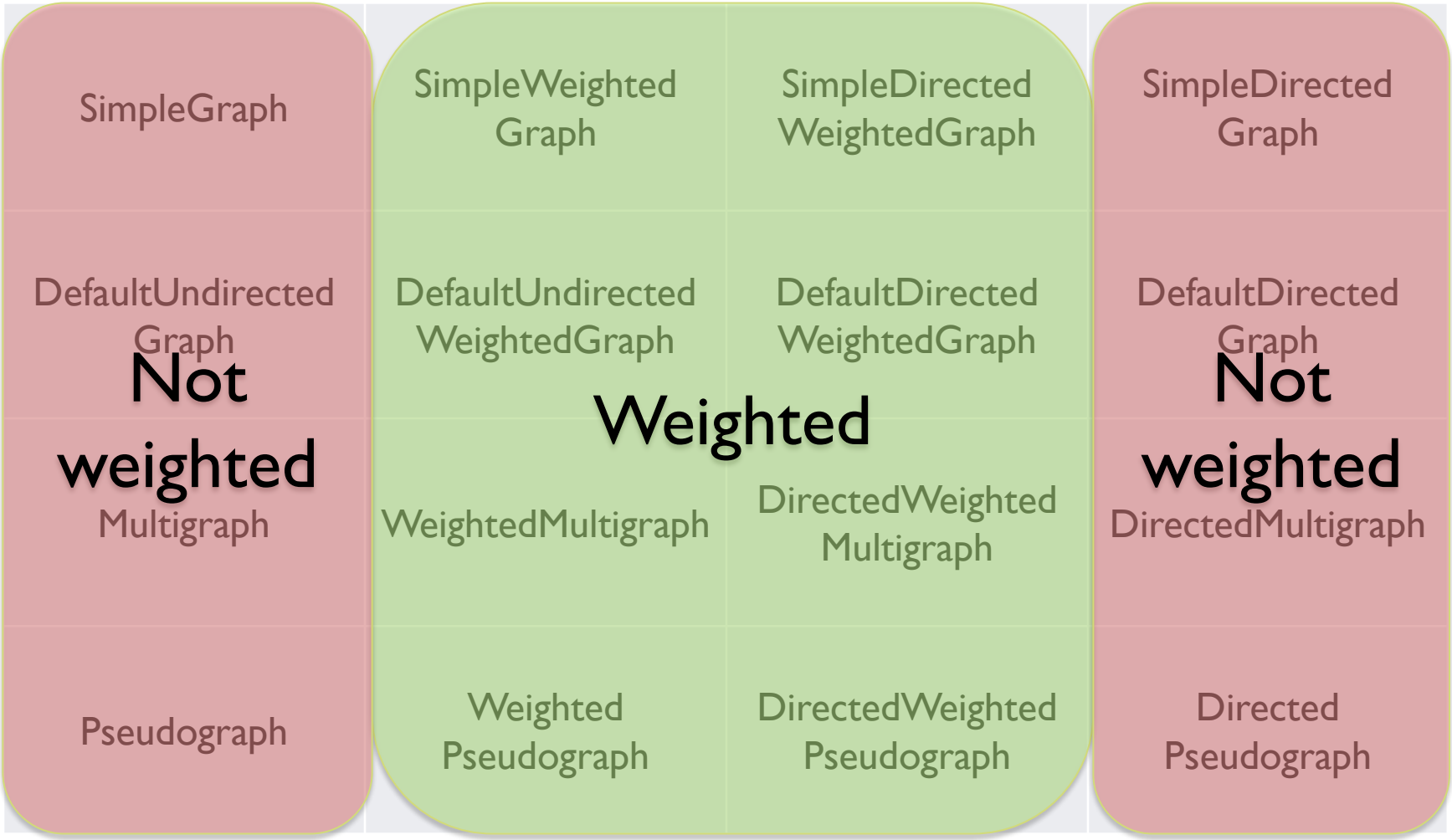
Graph Implementation Classes

SimpleGraph	SimpleWeighted Graph	SimpleDirected WeightedGraph	SimpleDirected Graph
DefaultUndirected Graph	DefaultUndirected WeightedGraph	DefaultDirected WeightedGraph	DefaultDirected Graph
Multigraph	WeightedMultigraph	DirectedWeighted Multigraph	DirectedMultigraph
Pseudograph	Weighted Pseudograph	DirectedWeighted Pseudograph	Directed Pseudograph

Graph Implementation Classes



Graph Implementation Classes



Graph Implementation Classes

Creating graphs

The jGraphT library

Creating graphs (1/2)

Class Name	Edges	Self-loops	Multiple edges	Weighted
DefaultUndirectedWeightedGraph	undirected	yes	no	yes
SimpleGraph	undirected	no	no	no
Multigraph	undirected	no	yes	no
Pseudograph	undirected	yes	yes	no
DefaultUndirectedGraph	undirected	yes	no	no
SimpleWeightedGraph	undirected	no	no	yes
WeightedMultigraph	undirected	no	yes	yes
WeightedPseudograph	undirected	yes	yes	yes
DefaultUndirectedWeightedGraph	undirected	yes	no	yes
SimpleDirectedGraph	directed	no	no	no
DirectedMultigraph	directed	no	yes	no
DirectedPseudograph	directed	yes	yes	no
DefaultDirectedGraph	directed	yes	no	no
SimpleDirectedWeightedGraph	directed	no	no	yes
DirectedWeightedMultigraph	directed	no	yes	yes
DirectedWeightedPseudograph	directed	yes	yes	yes
DefaultDirectedWeightedGraph	directed	yes	no	yes

- Decide what is the vertex class `V`
- Decide which graph class suits your needs
  - For unweighted graphs, use DefaultEdge as E
  - For weighted graphs, use DefaultWeightedEdge as E
- Create the graph object
  - `Graph<V,E> graph = new SimpleGraph<V,E>(E.class) ;`

Alternative: GraphTypeBuilder

- Use the GraphTypeBuilder class, and specify the variants of the graph you need:
  - ```
    Graph<Integer, DefaultEdge> g =
      GraphTypeBuilder
        .<Integer, DefaultEdge> undirected()
        .allowingMultipleEdges(false)
        .allowingSelfLoops(false)
        .edgeClass(DefaultEdge.class)
        .weighted(false)
        .buildGraph();
    ```
  - See: https://jgrapht.org/javadoc/org/jgrapht/graph/builder/GraphTypeBuilder.html
    ```

Creating graphs (2/2)

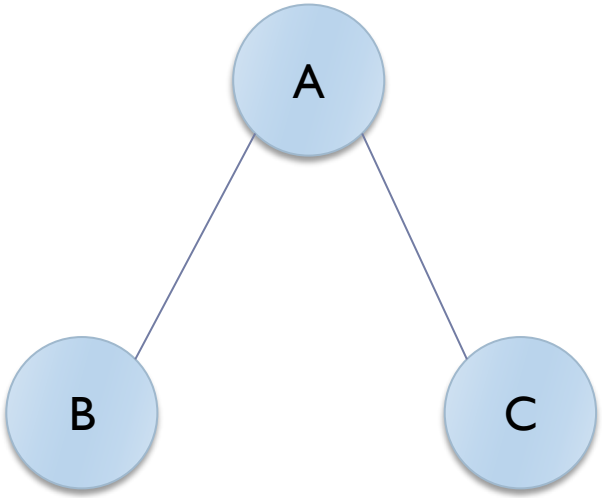
- Add vertices
  - boolean **addVertex**(V v)
- Add edges
  - E **addEdge**(V sourceVertex, V targetVertex)
  - boolean **addEdge**(V sourceVertex, V targetVertex, E e)
  - void **setEdgeWeight**(E e, double weight)
- Print graph (for debugging)
  - toString()
- Remember: E and V should correctly implement .equals() and .hashCode()

Example

```
Graph<String, DefaultEdge> graph = new
SimpleGraph<>(DefaultEdge.class) ;

graph.addVertex("A") ;
graph.addVertex("B") ;
graph.addVertex("C") ;

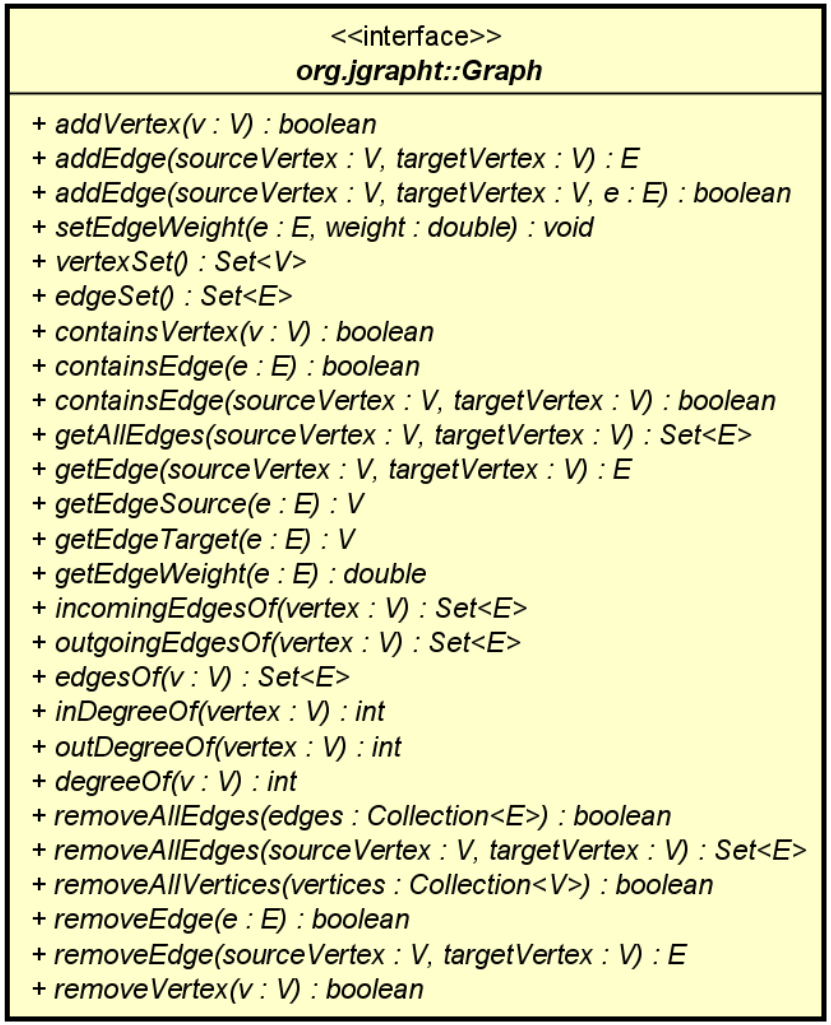
graph.addEdge("A", "B") ;
graph.addEdge("A", "C") ;
```



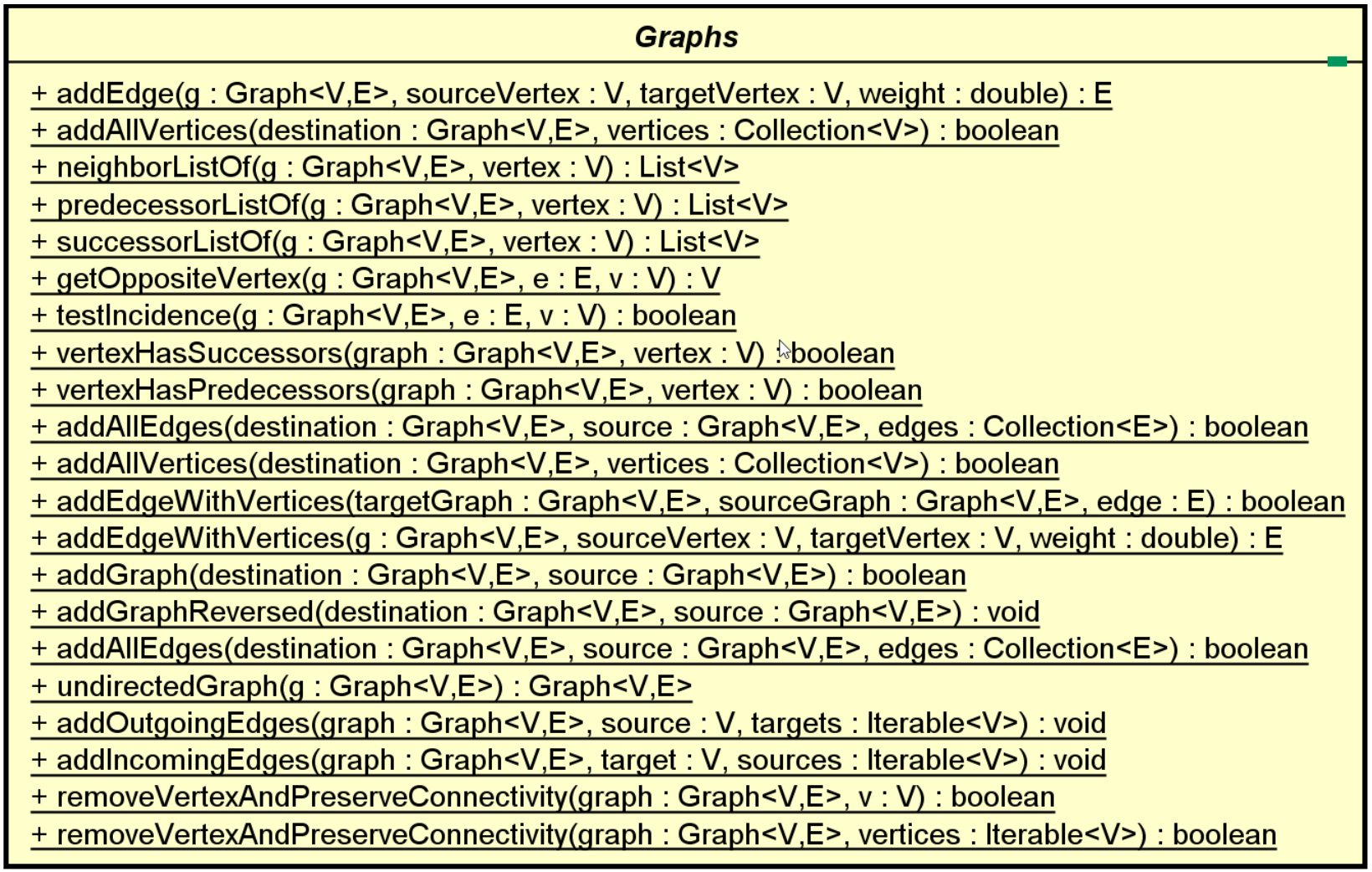
Querying graph structure

- Navigate structure
  - java.util.Set<V> **vertexSet**()
  - boolean **containsVertex**(V v)
  - boolean **containsEdge**(V sourceVertex,V targetVertex)
  - java.util.Set<E> **edgesOf**(V vertex)
  - java.util.Set<E> **getAllEdges**(V sourceVertex,V targetVertex)
- Query Edges
  - V **getEdgeSource**(E e)
  - V **getEdgeTarget**(E e)
  - double **getEdgeWeight**(E e)

Graph manipulation functions



The Graphs utility class



Utility functions

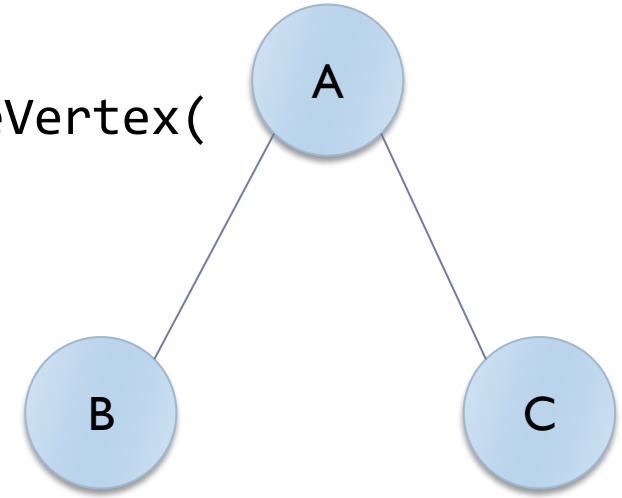
- Static class **org.jgrapht.Graphs**
- Easier creation
  - public static <V,E> E **addEdge**(Graph<V,E> g,V sourceVertex, V targetVertex, double weight)
  - public static <V,E> E **addEdgeWithVertices**(Graph<V,E> g, V sourceVertex,V targetVertex)

Utility functions

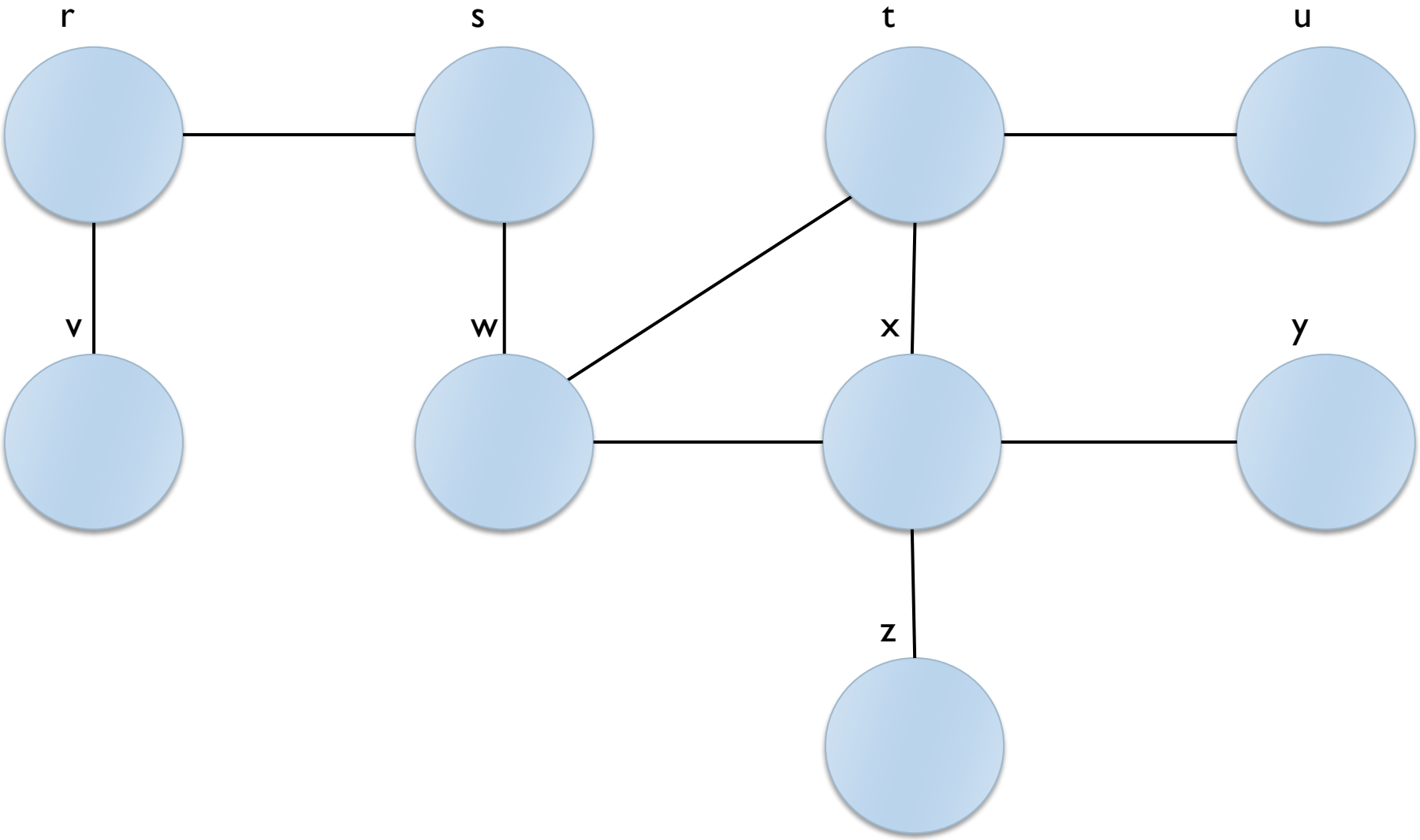
- Static class **org.jgrapht.Graphs**
- Easier navigation
  - public static <V,E> java.util.List<V> **neighborListOf**(Graph<V,E> g,V vertex)
  - public static String **getOppositeVertex**(Graph<String, DefaultEdge> g, DefaultEdge e, String v)
  - public static <V,E> java.util.List<V> **predecessorListOf**(DirectedGraph<V,E> g,V vertex)
  - public static <V,E> java.util.List<V> **successorListOf**(DirectedGraph<V,E> g,V vertex)

Example

```
for( String s: graph.vertexSet() ) {
        System.out.println("Vertex "+s) ;
        for( DefaultEdge e: graph.edgesOf(s) ) {
                System.out.println("Degree: “
                        +graph.degreeOf(s)) ;
                System.out.println(
                        Graphs.getOppositeVertex(
                        graph, e, s)) ;
        }
}
```



Example



Licenza d'uso

- Queste diapositive sono distribuite con licenza Creative Commons "Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)"
- Sei libero:
  - di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
  - di modificare quest'opera
- Alle seguenti condizioni:
  - Attribuzione — Devi attribuire la paternità dell'opera agli autori origina e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
  - Non commerciale — Non puoi usare quest'opera per fini commerciali.
  - Condividi allo stesso modo — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.
- http://creativecommons.org/licenses/by-nc-sa/3.0/