POLITECNICO DI TORINO

e-Lite

The jGraphT library

Tecniche di Programmazione – A.A. 2016/2017

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-core-0.9.0.jar`** to your project

Latest News

JGraphT source code has moved from the old svn repository to github. Please send all new contributions there in the form of pull requests!

Release 0.8.3 is now available. As of 0.8.0, JGraphT now requires Java 1.6 to build and run. If you are stuck on an earlier JVM/JDK version, you can keep using older releases of JGraphT.

If you'd like to help out with project maintenance, click the Like button for our "Adopt-a-Highway" initiative on Facebook!

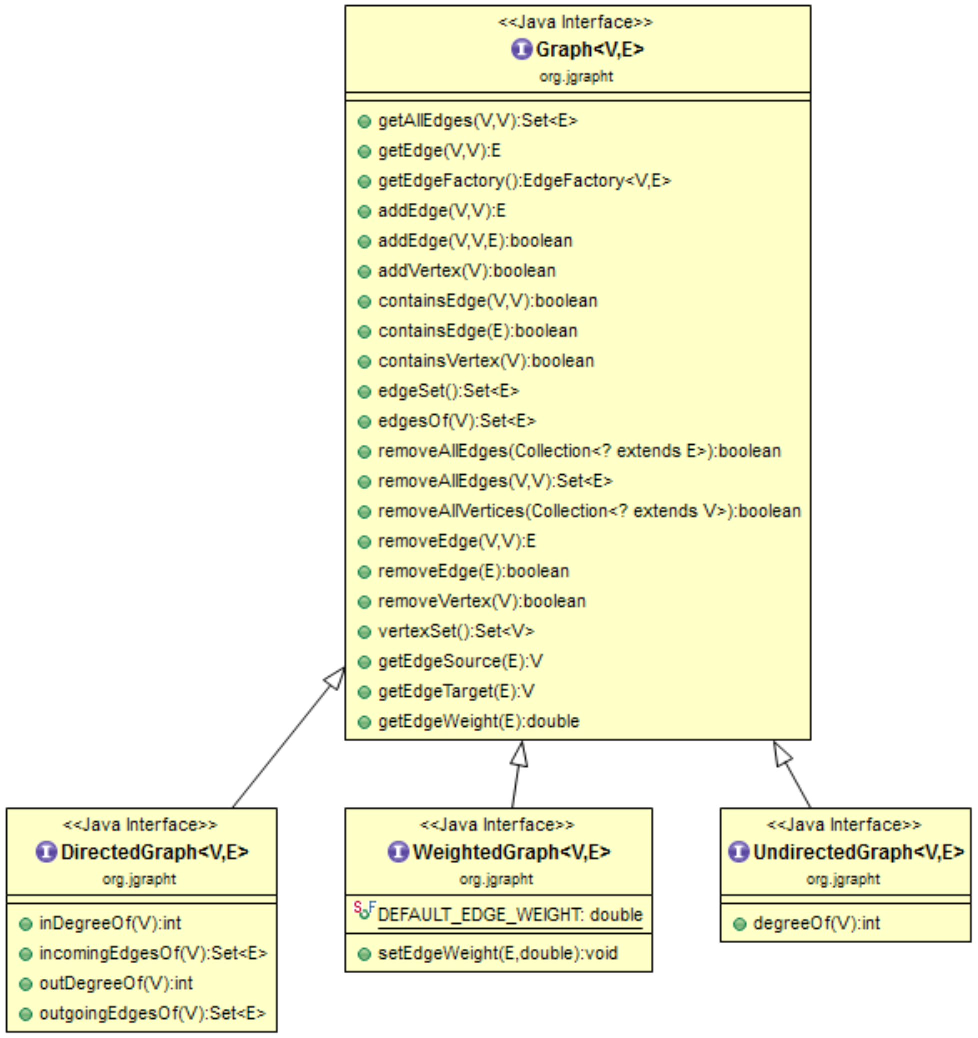
JGraphT structure

Packages	
org.jgrapht	The front-end API's interfaces and classes, including Graph, DirectedGraph and UndirectedGraph.
org.jgrapht.alg	Algorithms provided with JGraphT.
org.jgrapht.alg.util	Utilities used by JGraphT algorithms.
org.jgrapht.demo	Demo programs that help to get started with JGraphT.
org.jgrapht.event	Event classes and listener interfaces, used to provide a change notification mechanism on graph modification events.
org.jgrapht.ext	Extensions and integration means to other products.
org.jgrapht.generate	Generators for graphs of various topologies.
org.jgrapht.graph	Implementations of various graphs.
org.jgrapht.traverse	Graph traversal means.
org.jgrapht.util	Non-graph-specific data structures, algorithms, and utilities used by JGraphT.

Graph objects

- All graphs derive from
  - Interface `Graph<V,E>`
  - V = type of vertices
  - E = type of edges
    - usually `DefaultEdge` or `DefaultWeightedEdge`
- Main interfaces
  - `DirectedGraph<V,E>`
  - `UndirectedGraph<V,E>`
  - `WeightedGraph<V,E>`

JGraphT main interfaces



Graph classes

org.jgrapht

Graph I

DirectedGraph I

UndirectedGraph I

WeightedGraph I

org.jgrapht.graphs

SimpleGraph

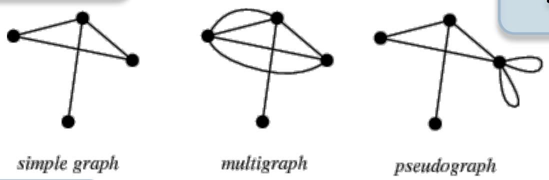
SimpleWeightedGraph

DefaultDirectedGraph

SimpleDirectedGraph

DefaultDirectedWeightedGraph

SimpleDirectedWeightedGraph



DirectedMultigraph

Multigraph

DirectedWeightedMultigraph

WeightedMultigraph

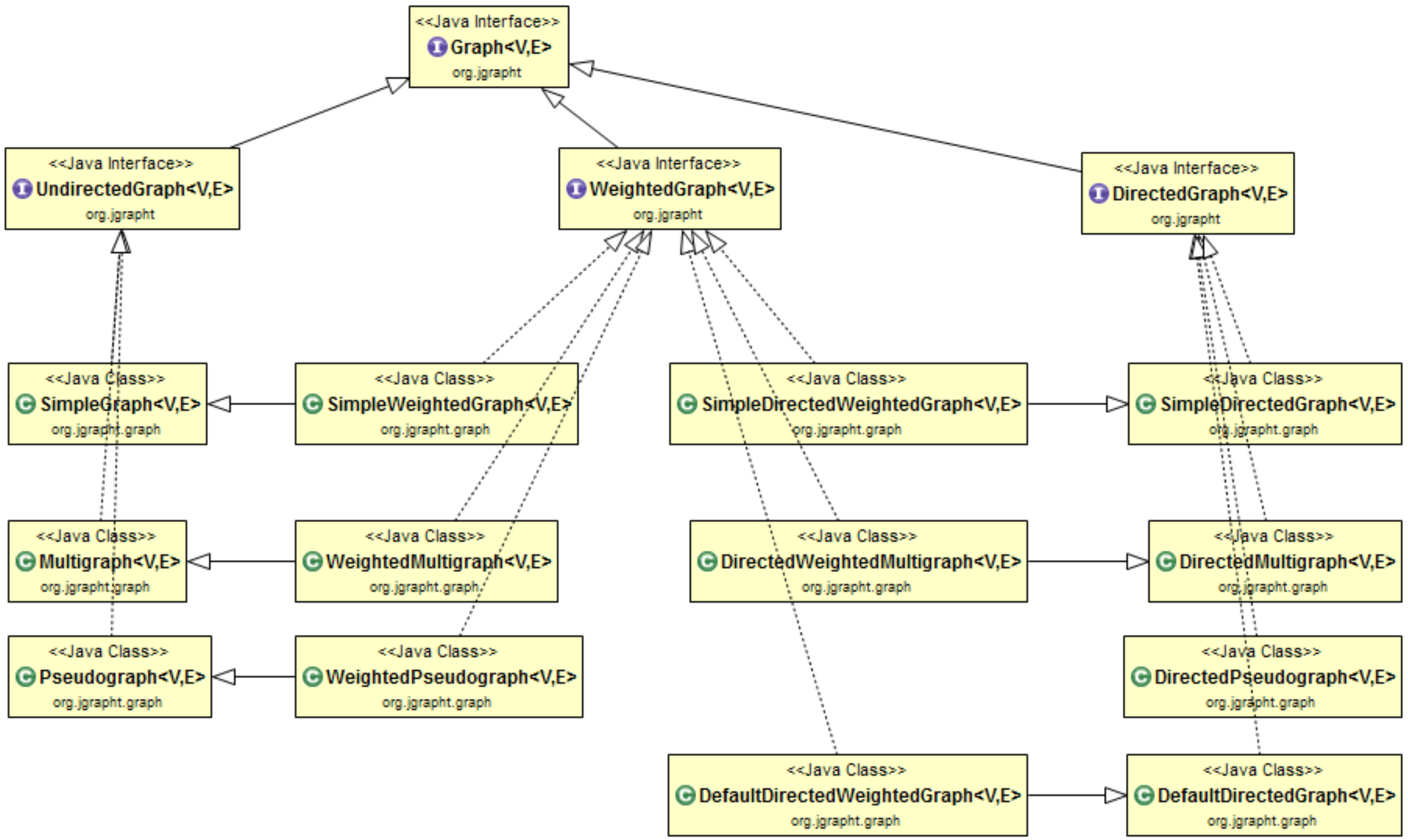
DirectedPseudograph

DirectedWeightedPseudograph

Pseudograph

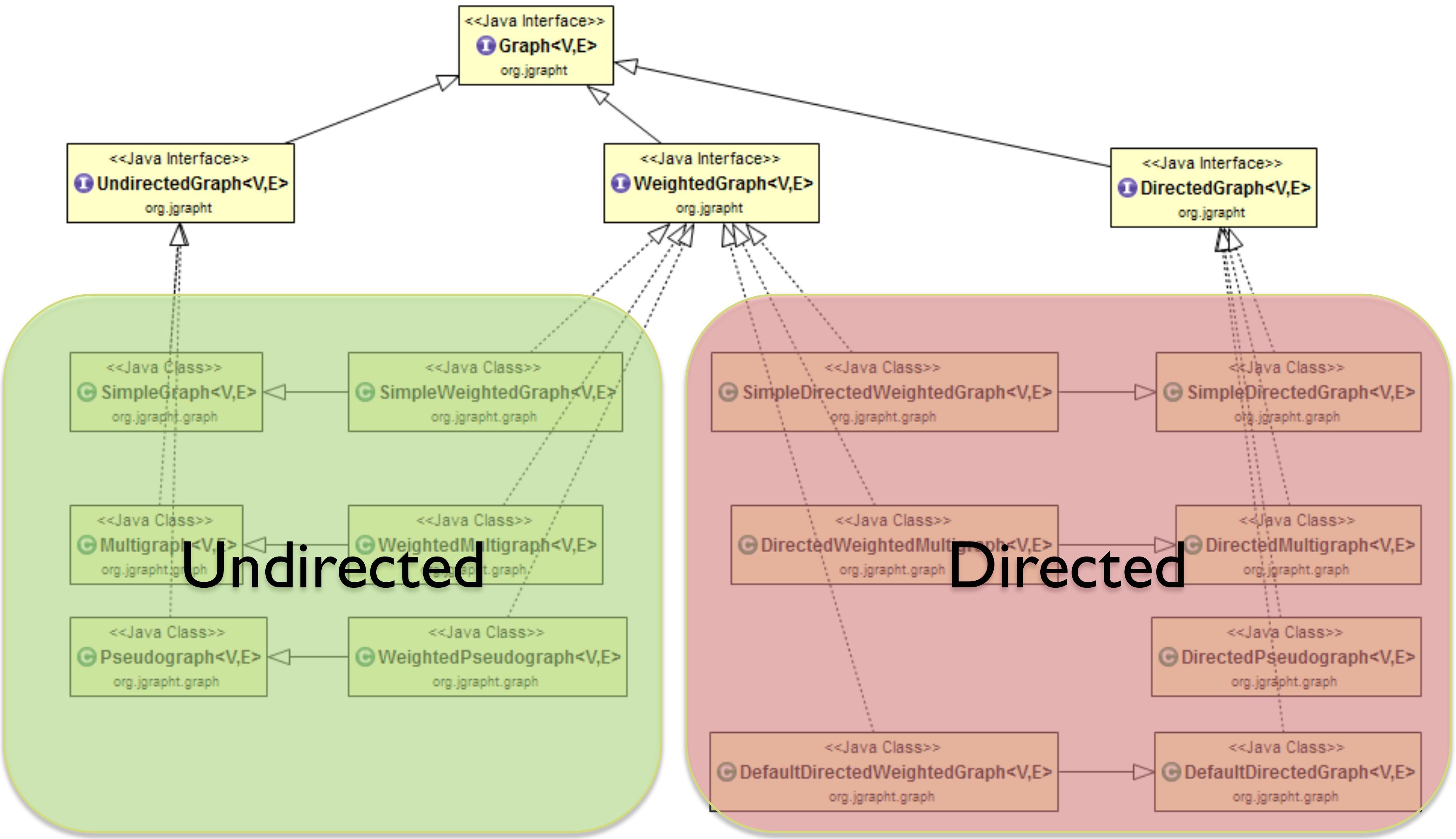
WeightedPseudograph

Graph classes

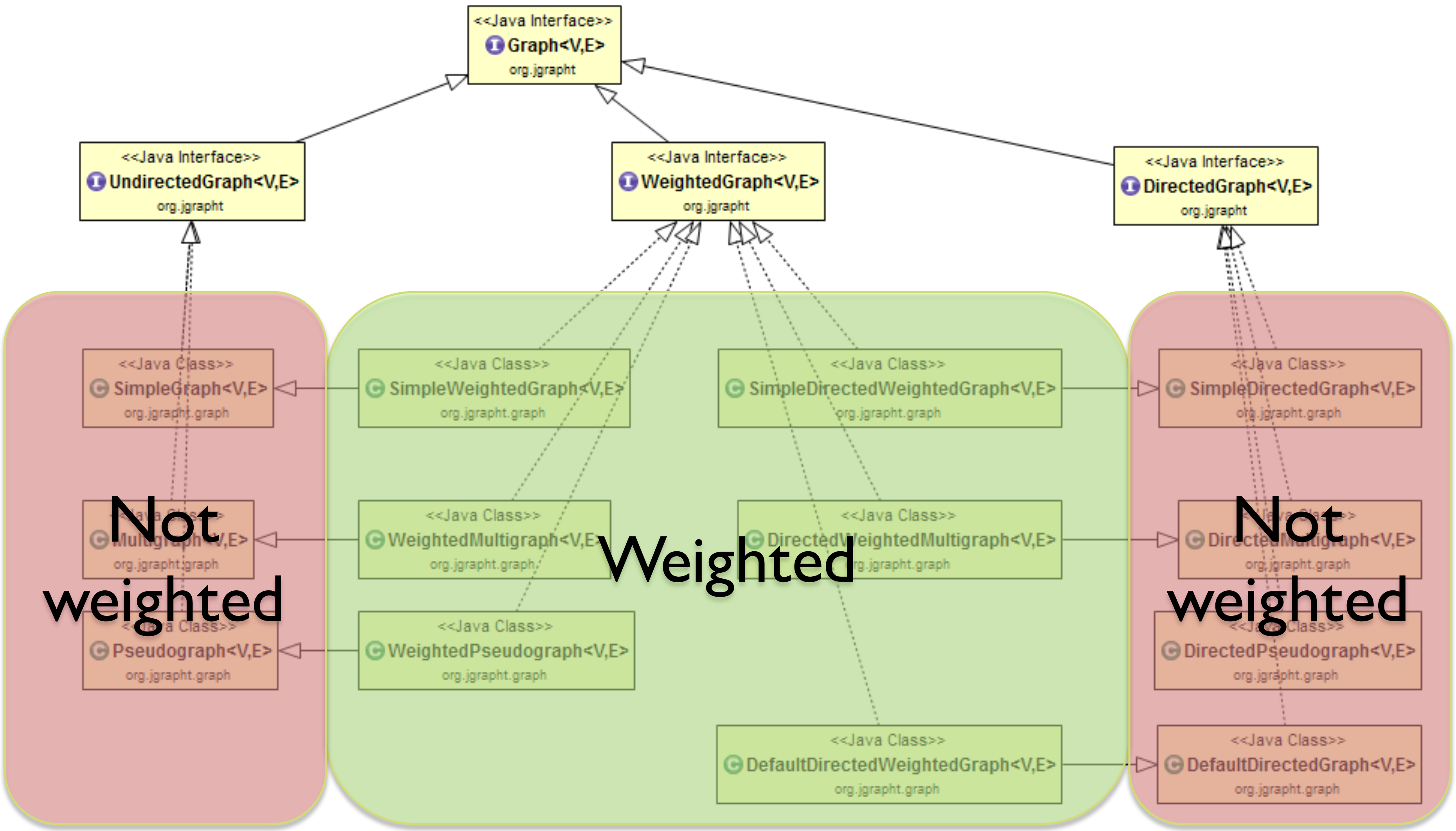


<<Java Interface>> Graph<V,E> org.jgrapht
<<Java Interface>> UndirectedGraph<V,E> org.jgrapht
<<Java Interface>> WeightedGraph<V,E> org.jgrapht
<<Java Interface>> DirectedGraph<V,E> org.jgrapht
<<Java Class>> SimpleGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleDirectedWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleDirectedGraph<V,E> org.jgrapht.graph
<<Java Class>> Multigraph<V,E> org.jgrapht.graph
<<Java Class>> WeightedMultigraph<V,E> org.jgrapht.graph
<<Java Class>> DirectedWeightedMultigraph<V,E> org.jgrapht.graph
<<Java Class>> DirectedMultigraph<V,E> org.jgrapht.graph
<<Java Class>> Pseudograph<V,E> org.jgrapht.graph
<<Java Class>> WeightedPseudograph<V,E> org.jgrapht.graph
<<Java Class>> DirectedPseudograph<V,E> org.jgrapht.graph
<<Java Class>> DefaultDirectedWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> DefaultDirectedGraph<V,E> org.jgrapht.graph

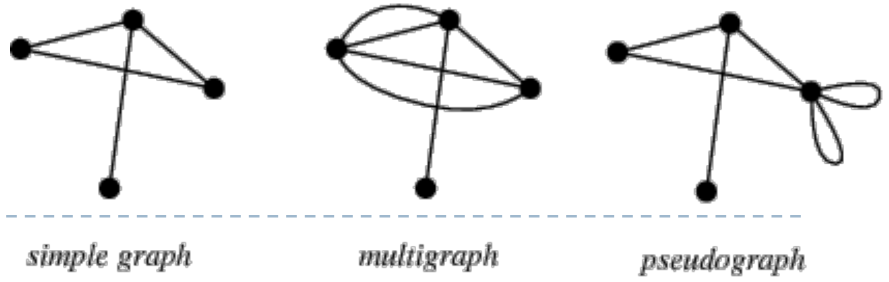
Graph classes

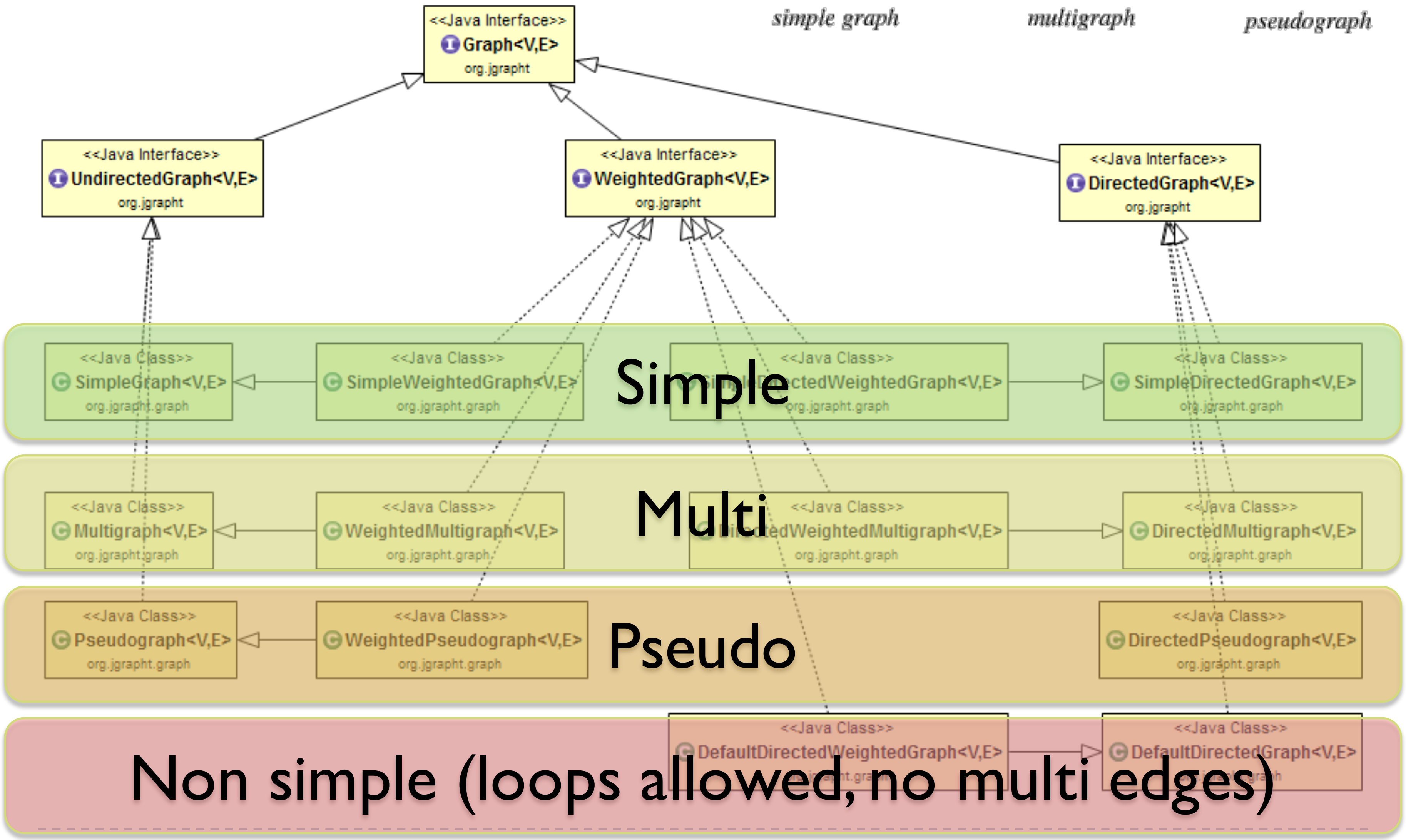


Graph classes



Graph classes




<<Java Interface>> Graph<V,E> org.jgrapht
<<Java Interface>> UndirectedGraph<V,E> org.jgrapht
<<Java Interface>> WeightedGraph<V,E> org.jgrapht
<<Java Interface>> DirectedGraph<V,E> org.jgrapht
Simple
<<Java Class>> SimpleGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleDirectedWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> SimpleDirectedGraph<V,E> org.jgrapht.graph
Multi
<<Java Class>> Multigraph<V,E> org.jgrapht.graph
<<Java Class>> WeightedMultigraph<V,E> org.jgrapht.graph
<<Java Class>> DirectedWeightedMultigraph<V,E> org.jgrapht.graph
<<Java Class>> DirectedMultigraph<V,E> org.jgrapht.graph
Pseudo
<<Java Class>> Pseudograph<V,E> org.jgrapht.graph
<<Java Class>> WeightedPseudograph<V,E> org.jgrapht.graph
<<Java Class>> DirectedPseudograph<V,E> org.jgrapht.graph
<<Java Class>> DefaultDirectedWeightedGraph<V,E> org.jgrapht.graph
<<Java Class>> DefaultDirectedGraph<V,E> org.jgrapht.graph
Non simple (loops allowed, no multi edges)

Creating graphs

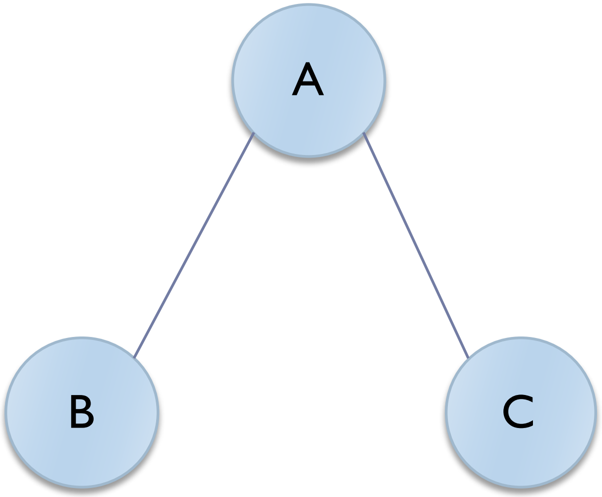
The jGraphT library

Creating graphs

- Construct your desired type of graph
- 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()
- Warning: E and V should correctly implement .equals() and .hashCode()

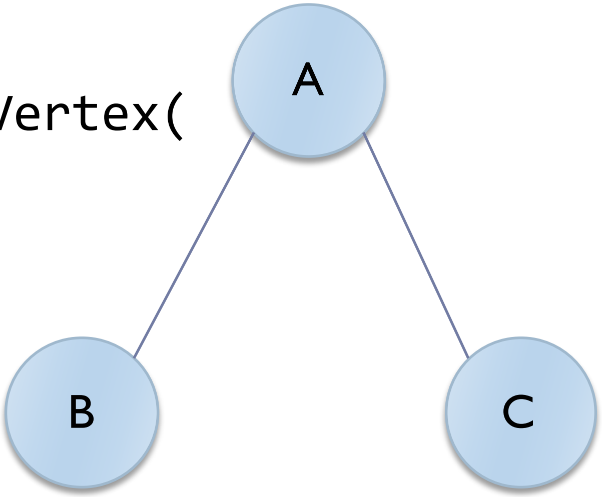
Example

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

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

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

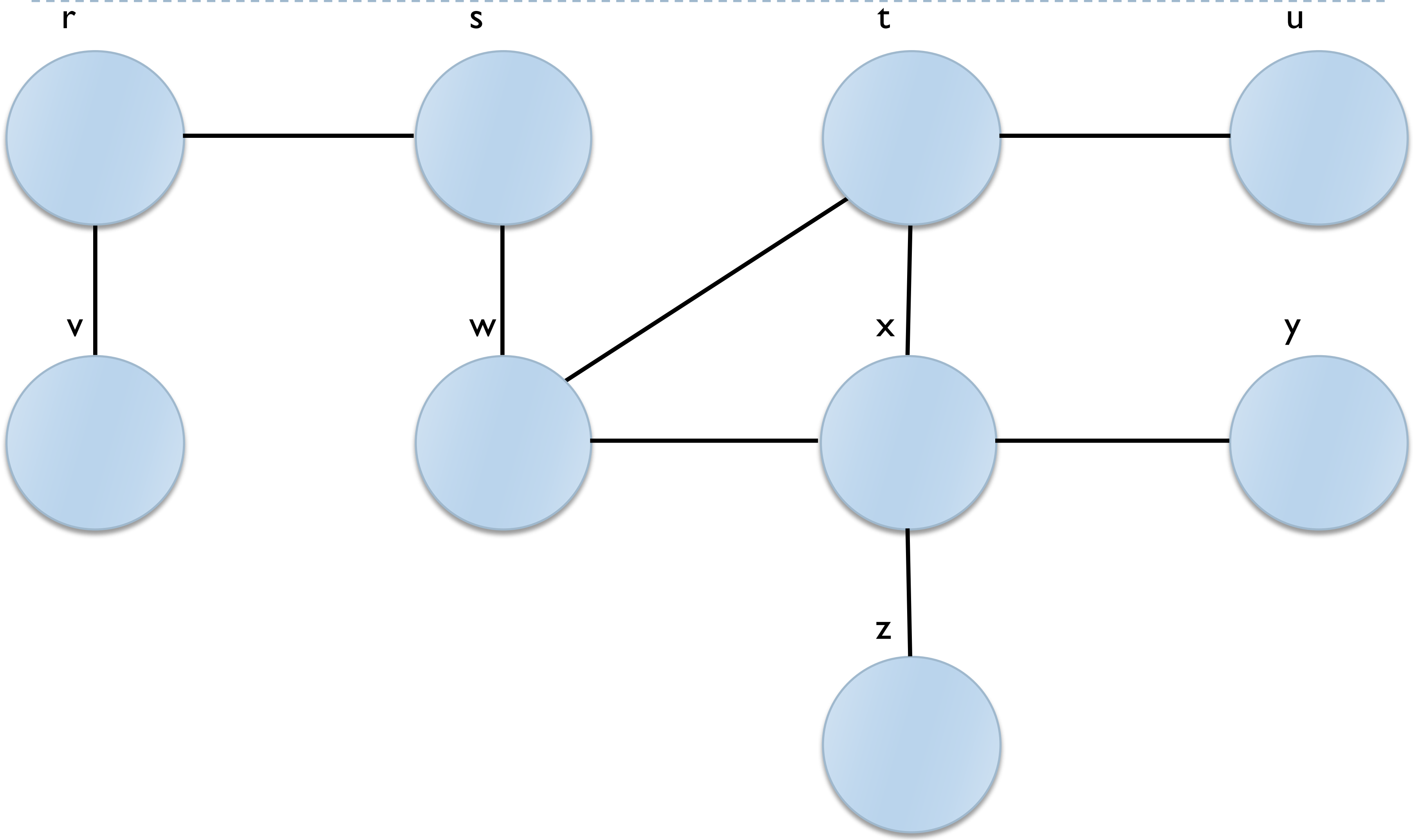


Example

```
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



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)

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)
- 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)

Licenza d'uso