POLITECNICO DI TORINO

e-Lite

# The jGraphT library

Tecniche di Programmazione – A.A. 2015/2016

# Summary

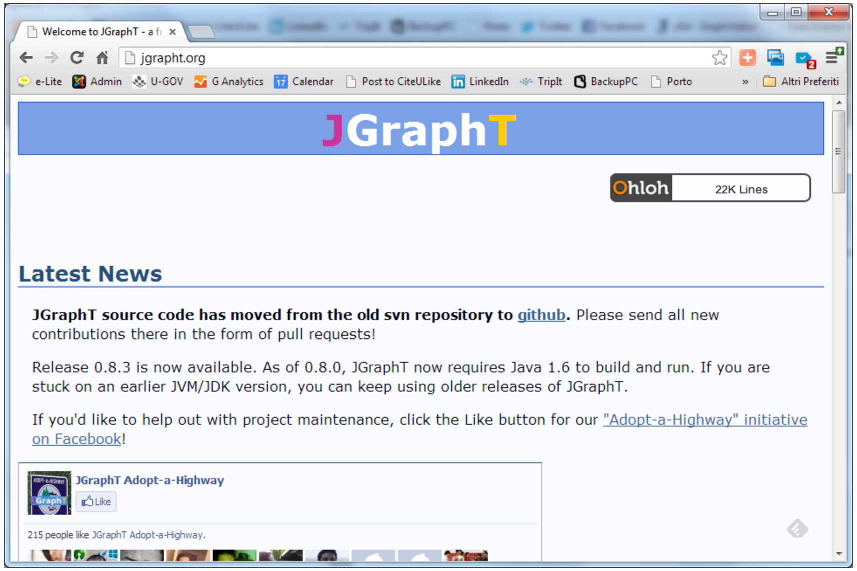
- The JGraphT library
- Creating graphs
- Visits in JGraphT

# 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!

215 people like JGraphT Adopt-a-Highway.

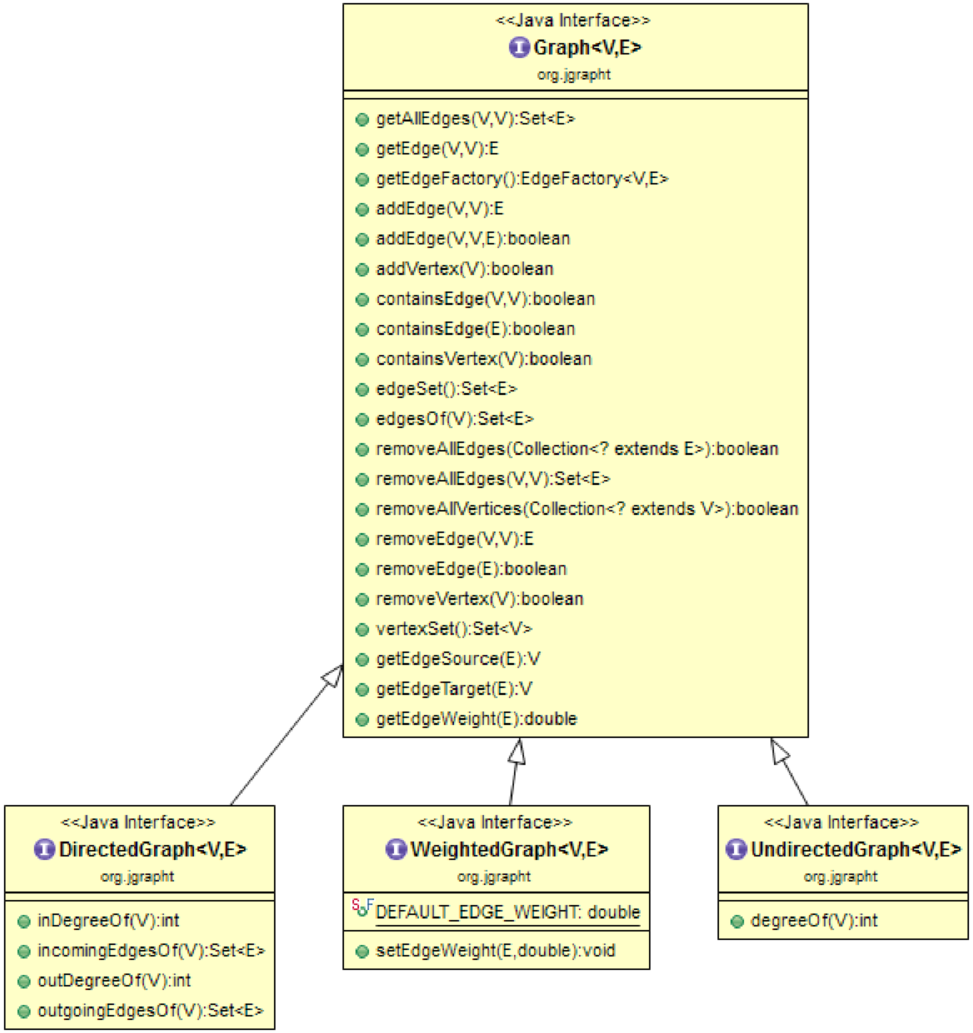
# JGraphT structure

| 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



<<Java Interface>>
**Graph<V,E>**
org.jgrapht

- getAllEdges(V,V):Set<E>
- getEdge(V,V):E
- getEdgeFactory():EdgeFactory<V,E>
- addEdge(V,V):E
- addEdge(V,V,E):boolean
- addVertex(V):boolean
- containsEdge(V,V):boolean
- containsEdge(E):boolean
- containsVertex(V):boolean
- edgeSet():Set<E>
- edgesOf(V):Set<E>
- removeAllEdges(Collection<? extends E>):boolean
- removeAllEdges(V,V):Set<E>
- removeAllVertices(Collection<? extends V>):boolean
- removeEdge(V,V):E
- removeEdge(E):boolean
- removeVertex(V):boolean
- vertexSet():Set<V>
- getEdgeSource(E):V
- getEdgeTarget(E):V
- getEdgeWeight(E):double

<<Java Interface>>
**DirectedGraph<V,E>**
org.jgrapht

- inDegreeOf(V):int
- incomingEdgesOf(V):Set<E>
- outDegreeOf(V):int
- outgoingEdgesOf(V):Set<E>

<<Java Interface>>
**WeightedGraph<V,E>**
org.jgrapht

DEFAULT_EDGE_WEIGHT: double

- setEdgeWeight(E,double):void

<<Java Interface>>
**UndirectedGraph<V,E>**
org.jgrapht

- degreeOf(V):int

# Graph classes

org.jgrapht

Graph I

DirectedGraph I

UndirectedGraph I

WeightedGraph I

org.jgrapht.graphs

DefaultDirectedGraph

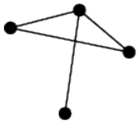
DefaultDirectedWeightedGraph

SimpleGraph

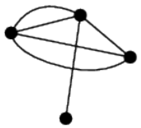
SimpleWeightedGraph

SimpleDirectedGraph

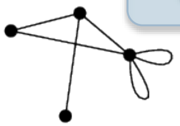
SimpleDirectedWeightedGraph



*simple graph*



*multigraph*



*pseudograph*

DirectedMultigraph

Multigraph

DirectedWeightedMultigraph

WeightedMultigraph

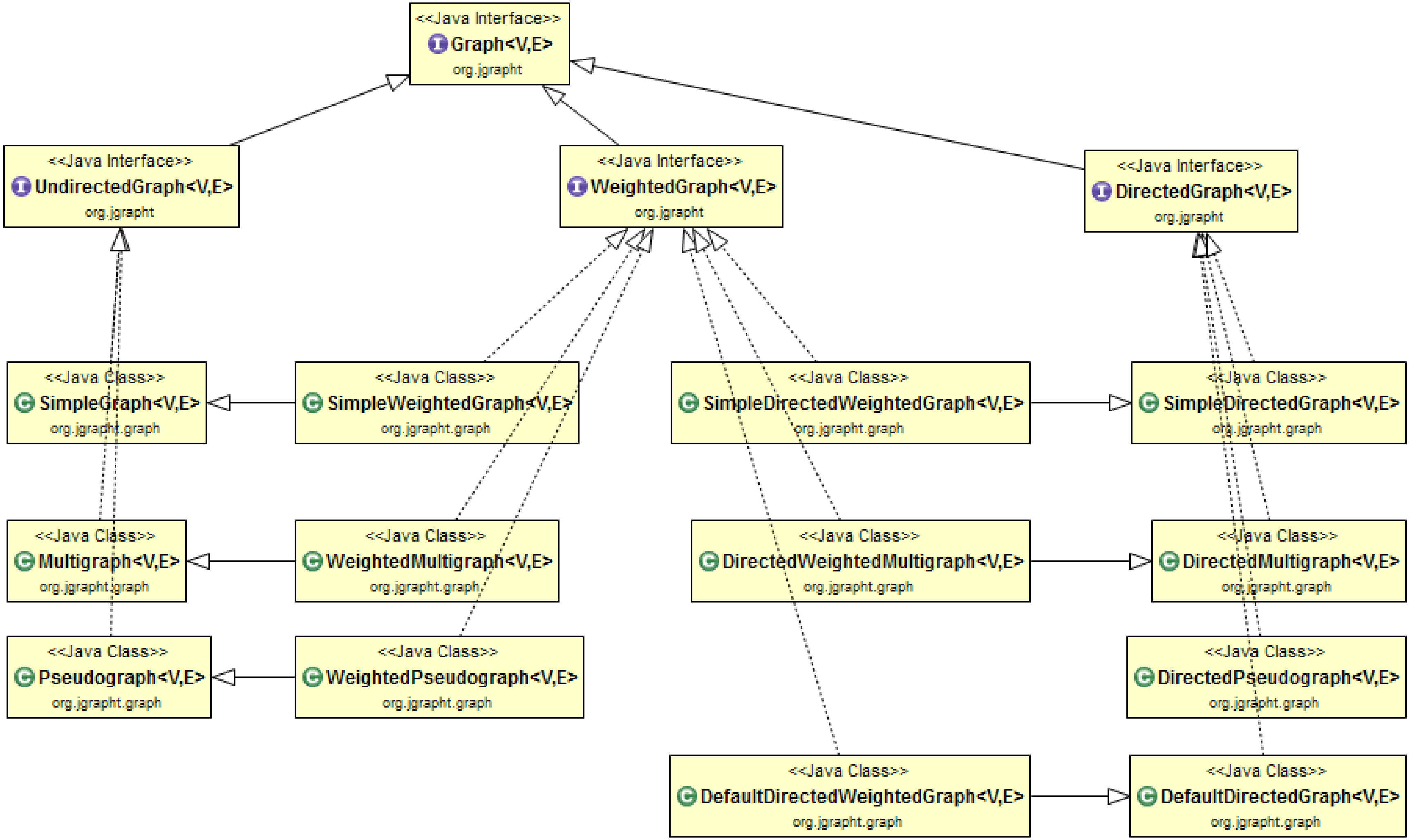
DirectedPseudograph

DirectedWeightedPseudograph

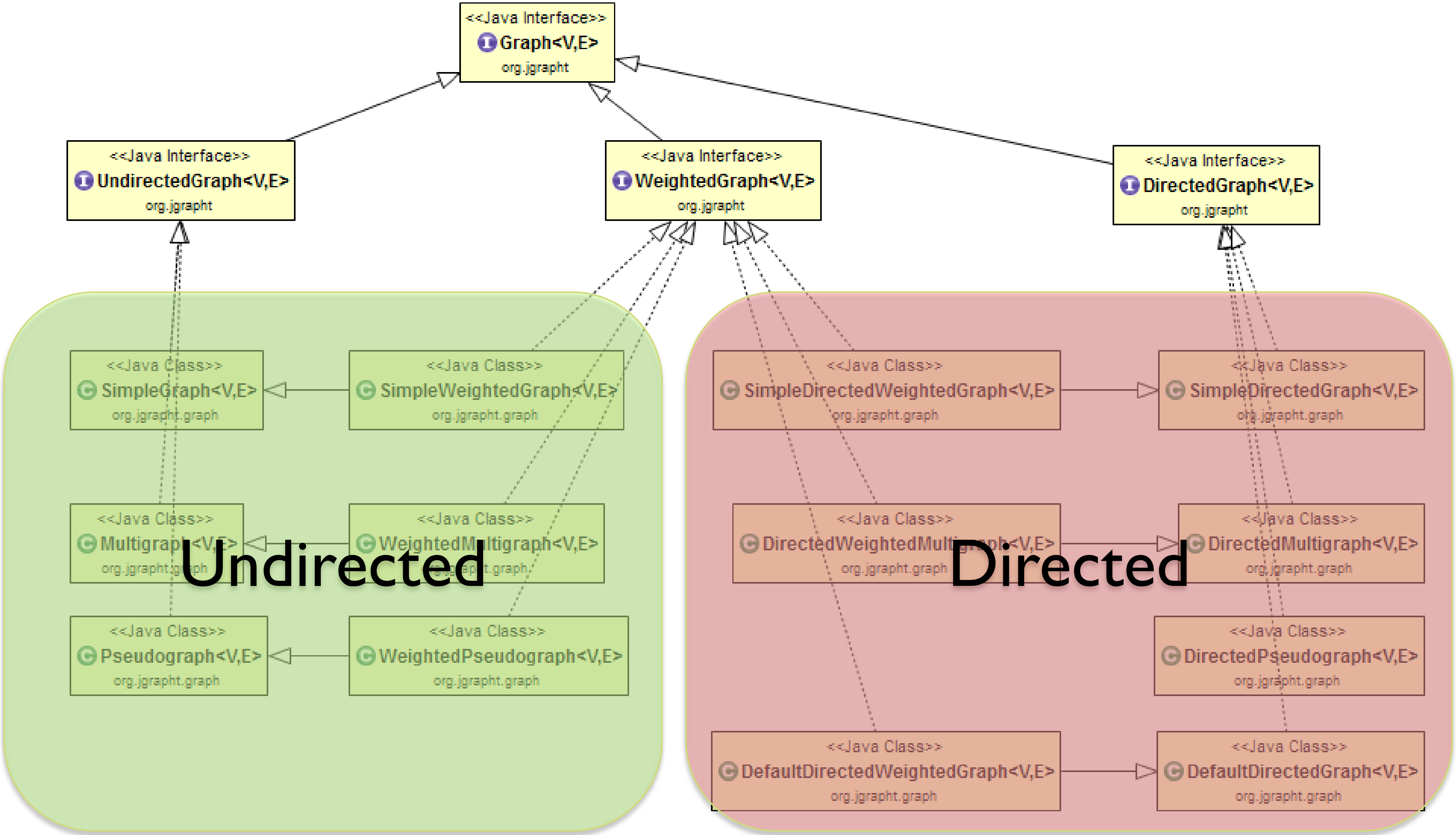
Pseudograph

WeightedPseudograph

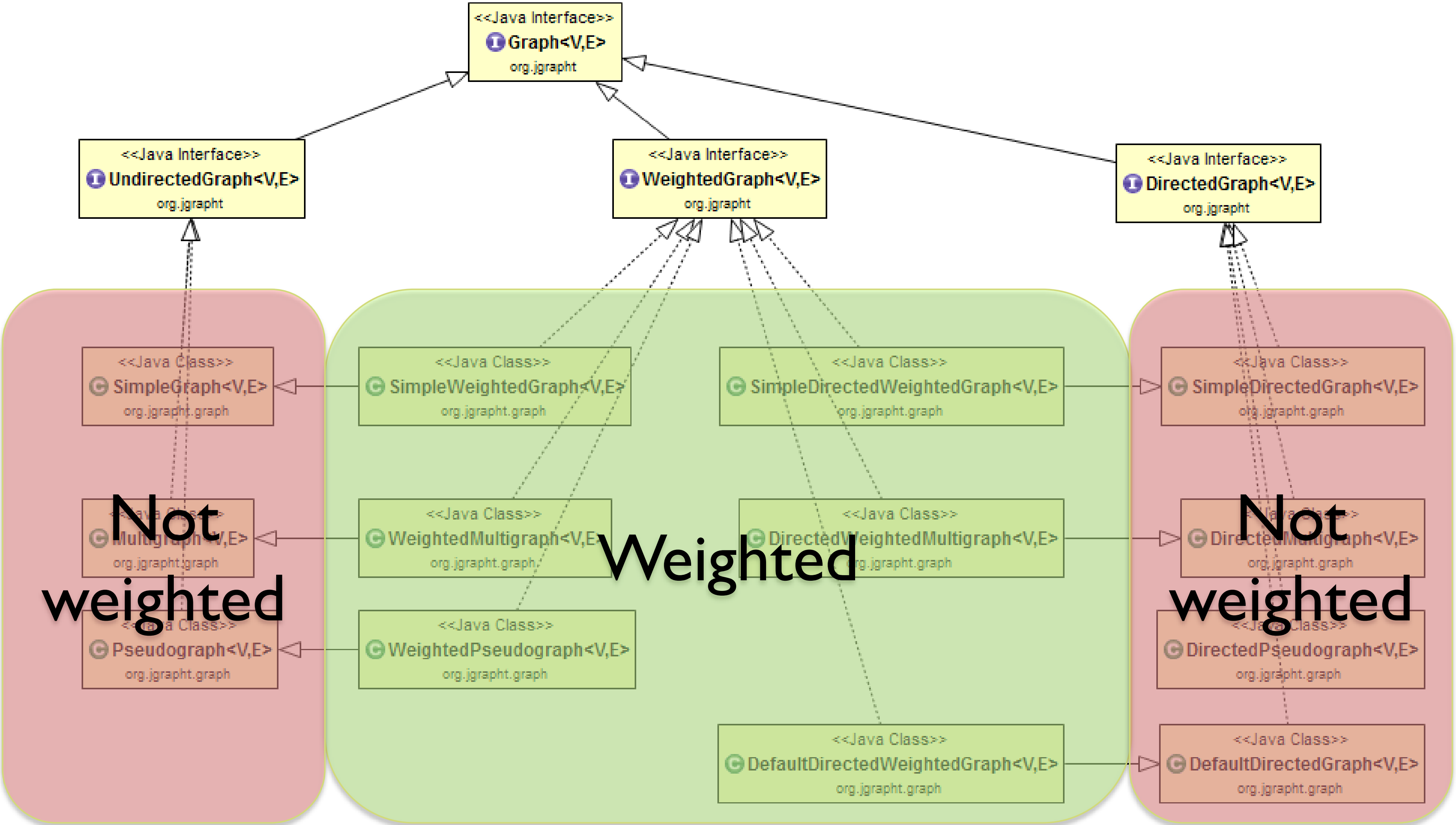
# Graph classes



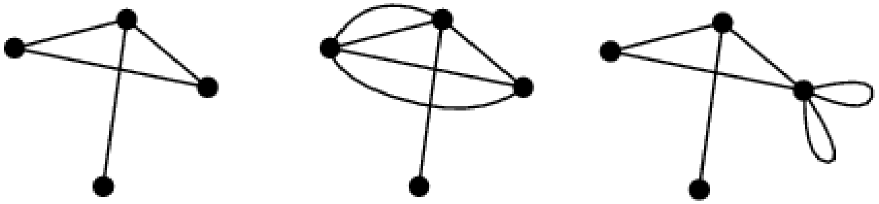
# Graph classes



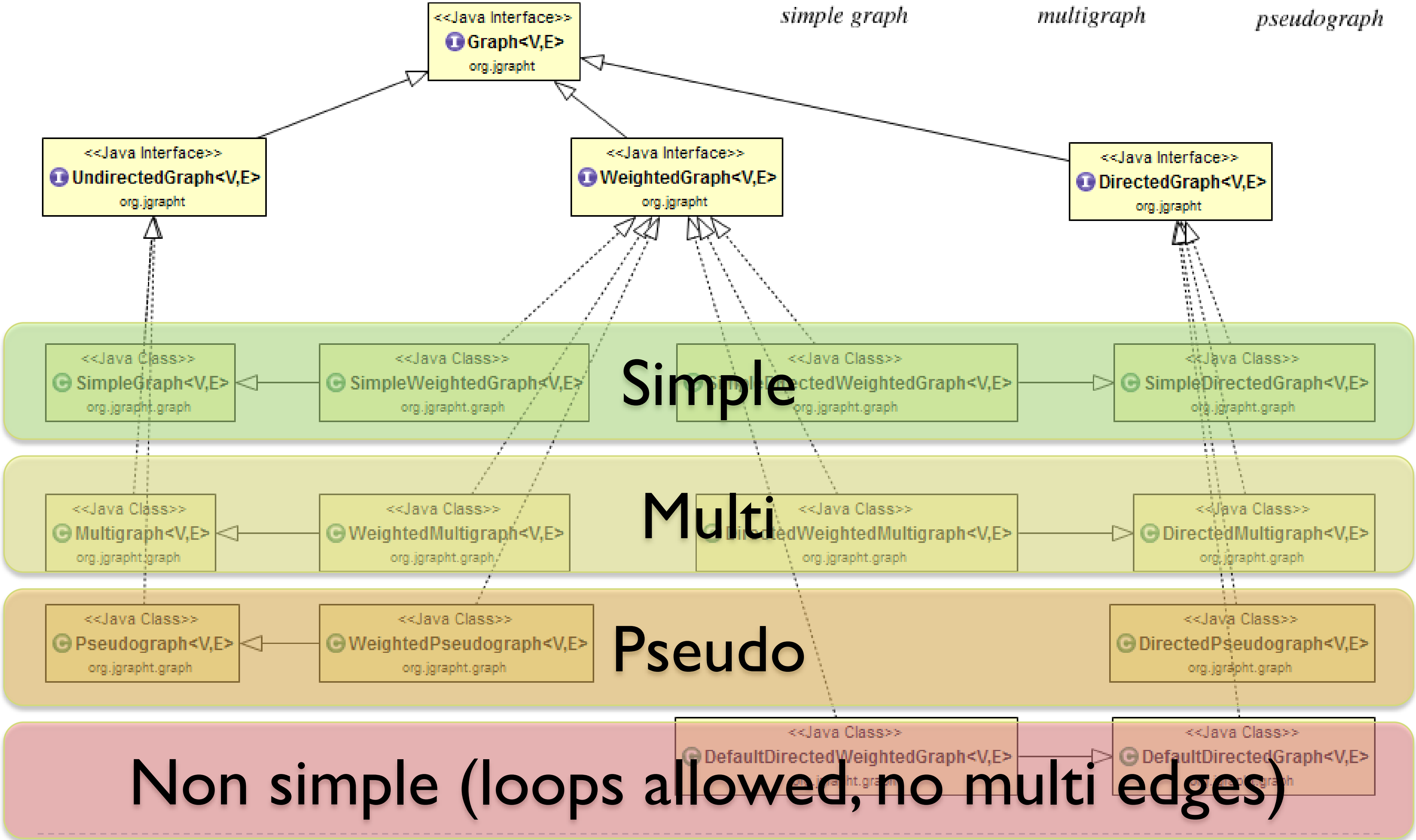
# Graph classes



# Graph classes



*simple graph* *multigraph* *pseudograph*



Simple

Multi

Pseudo

Non simple (loops allowed, no multi edges)

# Creating graphs

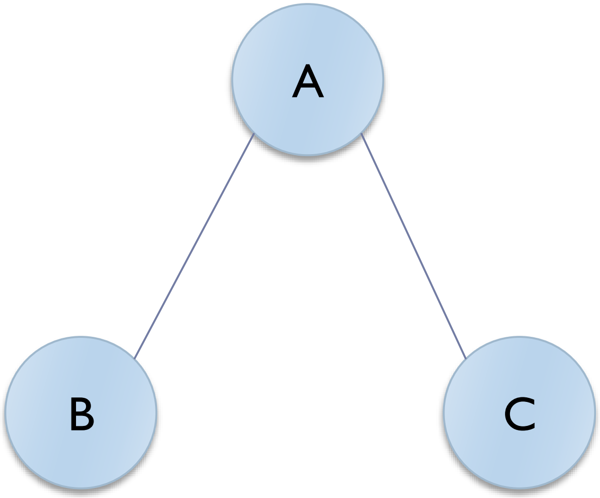
The jGraphT library

# 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()

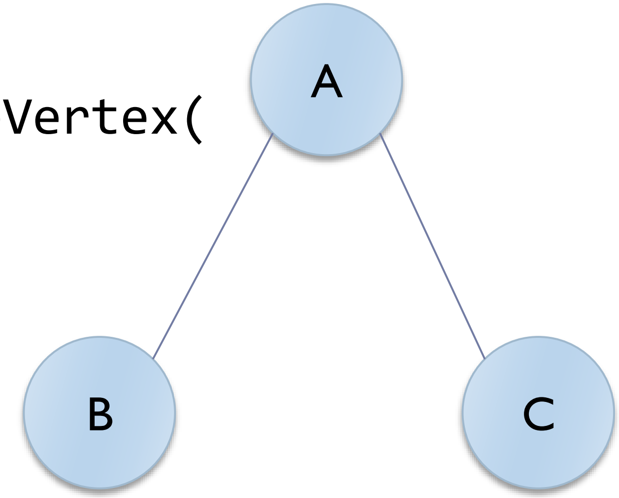
# Example

```
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") ;
```

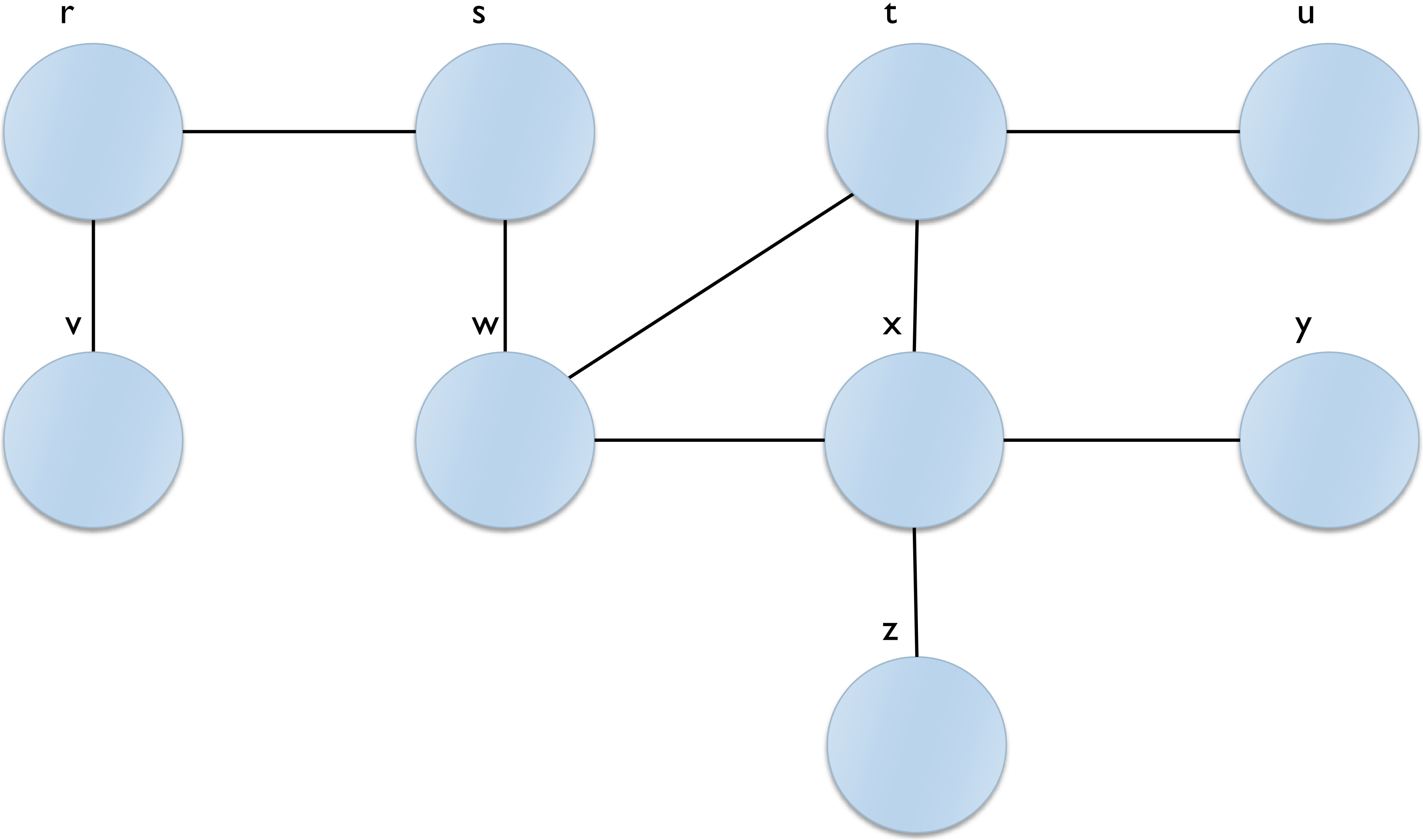


# Example

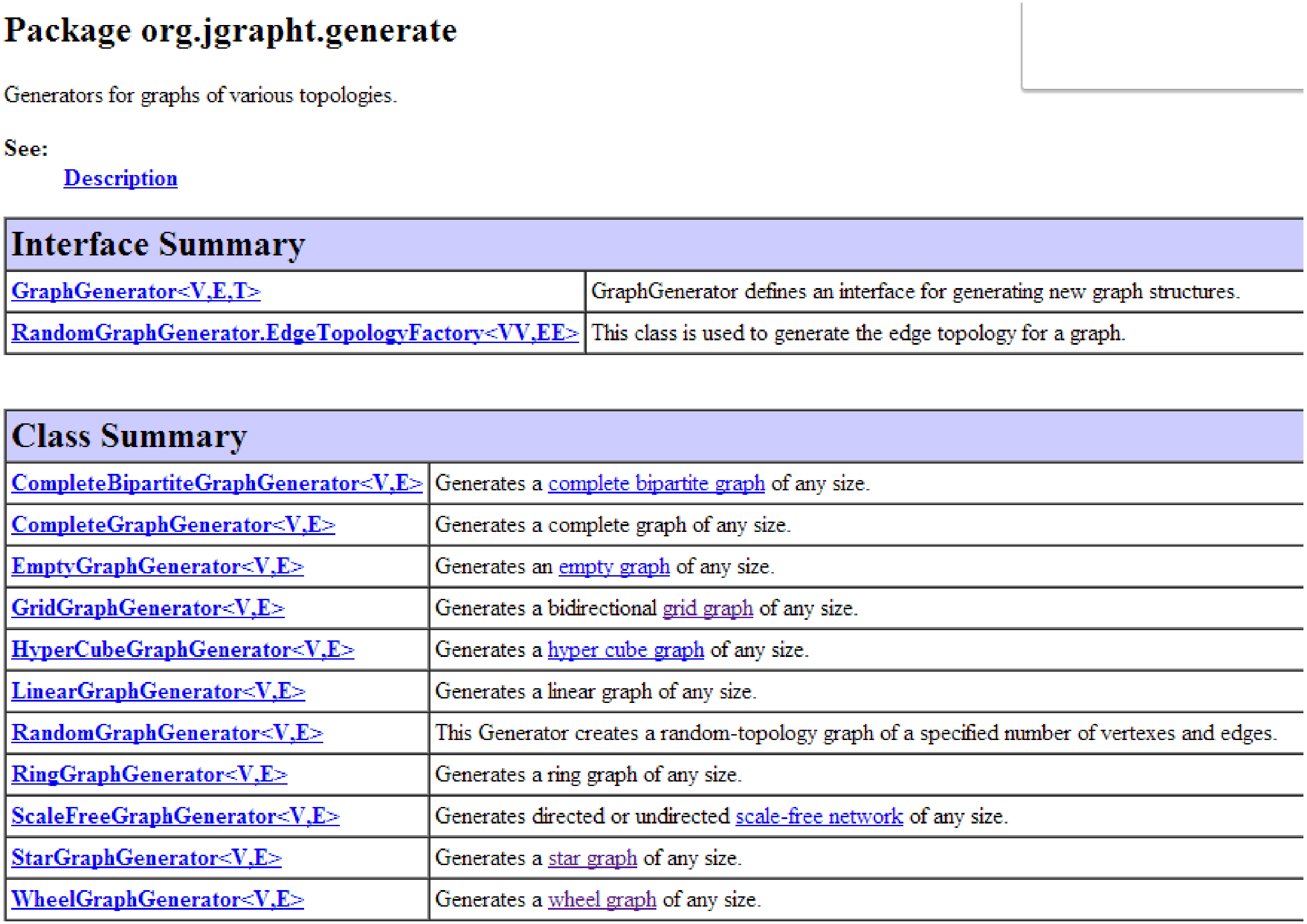
```
for( String s: graph.vertexSet() ) {
      System.out.println("Vertex "+s) ;
      for( DefaultEdge e: graph.edgesOf(s) ) {
            System.out.println("Degree: “
                  +graph.degreeOf(s)) ;
            System.out.println(
                  Graphs.getOppositeVertex(
              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



# Example



# For testing…



## Package org.jgrapht.generate

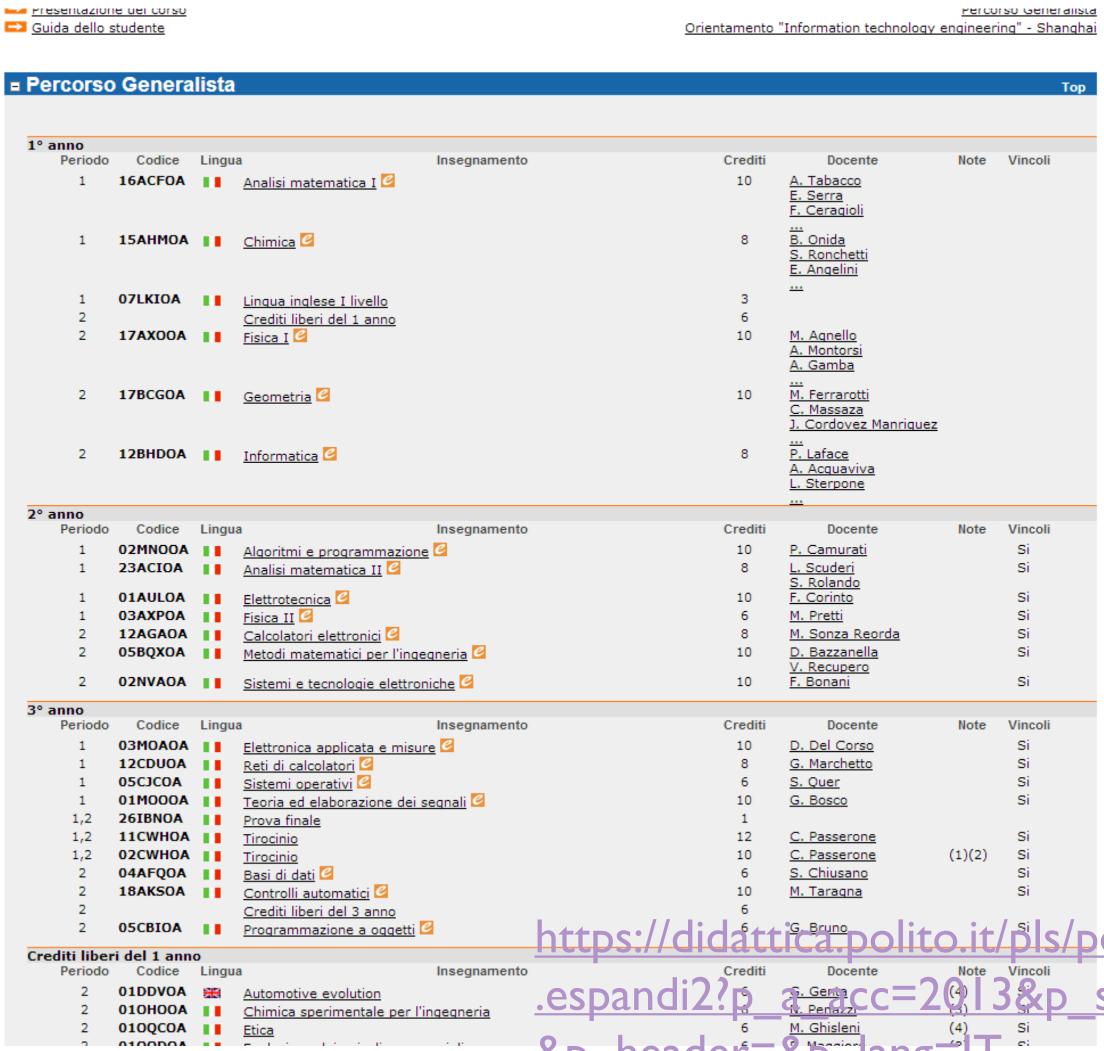
Generators for graphs of various topologies.

**See:**
Description

| Interface Summary | |
|---|---|
| GraphGenerator<V,E,T> | GraphGenerator defines an interface for generating new graph structures. |
| RandomGraphGenerator.EdgeTopologyFactory<VV,EE> | This class is used to generate the edge topology for a graph. |

| Class Summary | |
|---|---|
| CompleteBipartiteGraphGenerator<V,E> | Generates a complete bipartite graph of any size. |
| CompleteGraphGenerator<V,E> | Generates a complete graph of any size. |
| EmptyGraphGenerator<V,E> | Generates an empty graph of any size. |
| GridGraphGenerator<V,E> | Generates a bidirectional grid graph of any size. |
| HyperCubeGraphGenerator<V,E> | Generates a hyper cube graph of any size. |
| LinearGraphGenerator<V,E> | Generates a linear graph of any size. |
| RandomGraphGenerator<V,E> | This Generator creates a random-topology graph of a specified number of vertexes and edges. |
| RingGraphGenerator<V,E> | Generates a ring graph of any size. |
| ScaleFreeGraphGenerator<V,E> | Generates directed or undirected scale-free network of any size. |
| StarGraphGenerator<V,E> | Generates a star graph of any size. |
| WheelGraphGenerator<V,E> | Generates a wheel graph of any size. |

# Example



Presentazione del corso
Guida dello studente

Percorso Generalista
Orientamento "Information technology engineering" - Shanghai

## Percorso Generalista

Top

**1° anno**

| Periodo | Codice | Lingua | Insegnamento | Crediti | Docente | Note | Vincoli |
|---|---|---|---|---|---|---|---|
| 1 | 16ACFOA | 🇮🇹 | Analisi matematica I | 10 | A. Tabacco<br>E. Serra<br>F. Ceragioli<br>... | | |
| 1 | 15AHMOA | 🇮🇹 | Chimica | 8 | B. Onida<br>S. Ronchetti<br>E. Angelini<br>... | | |
| 1 | 07LKIOA | 🇮🇹 | Lingua inglese I livello | 3 | | | |
| 2 | | | Crediti liberi del 1 anno | 6 | | | |
| 2 | 17AXOOA | 🇮🇹 | Fisica I | 10 | M. Agnello<br>A. Montorsi<br>A. Gamba<br>... | | |
| 2 | 17BCGOA | 🇮🇹 | Geometria | 10 | M. Ferrarotti<br>C. Massaza<br>J. Cordovez Manriquez<br>... | | |
| 2 | 12BHDOA | 🇮🇹 | Informatica | 8 | P. Laface<br>A. Acquaviva<br>L. Sterpone<br>... | | |

**2° anno**

| Periodo | Codice | Lingua | Insegnamento | Crediti | Docente | Note | Vincoli |
|---|---|---|---|---|---|---|---|
| 1 | 02MNOOA | 🇮🇹 | Algoritmi e programmazione | 10 | P. Camurati | | Si |
| 1 | 23ACIOA | 🇮🇹 | Analisi matematica II | 8 | L. Scuderi<br>S. Rolando | | Si |
| 1 | 01AULOA | 🇮🇹 | Elettrotecnica | 10 | F. Corinto | | Si |
| 1 | 03AXPOA | 🇮🇹 | Fisica II | 6 | M. Pretti | | Si |
| 2 | 12AGAOA | 🇮🇹 | Calcolatori elettronici | 8 | M. Sonza Reorda | | Si |
| 2 | 05BQXOA | 🇮🇹 | Metodi matematici per l'ingegneria | 10 | D. Bazzanella<br>V. Recupero | | Si |
| 2 | 02NVAOA | 🇮🇹 | Sistemi e tecnologie elettroniche | 10 | F. Bonani | | Si |

**3° anno**

| Periodo | Codice | Lingua | Insegnamento | Crediti | Docente | Note | Vincoli |
|---|---|---|---|---|---|---|---|
| 1 | 03MOAOA | 🇮🇹 | Elettronica applicata e misure | 10 | D. Del Corso | | Si |
| 1 | 12CDUOA | 🇮🇹 | Reti di calcolatori | 8 | G. Marchetto | | Si |
| 1 | 05CJCOA | 🇮🇹 | Sistemi operativi | 6 | S. Quer | | Si |
| 1 | 01MOOOA | 🇮🇹 | Teoria ed elaborazione dei segnali | 10 | G. Bosco | | Si |
| 1,2 | 26IBNOA | 🇮🇹 | Prova finale | 1 | | | |
| 1,2 | 11CWHOA | 🇮🇹 | Tirocinio | 12 | C. Passerone | | Si |
| 1,2 | 02CWHOA | 🇮🇹 | Tirocinio | 10 | C. Passerone | (1)(2) | Si |
| 2 | 04AFQOA | 🇮🇹 | Basi di dati | 6 | S. Chiusano | | Si |
| 2 | 18AKSOA | 🇮🇹 | Controlli automatici | 10 | M. Taragna | | Si |
| 2 | | | Crediti liberi del 3 anno | 6 | | | |
| 2 | 05CBIOA | 🇮🇹 | Programmazione a oggetti | 6 | G. Bruno | | Si |

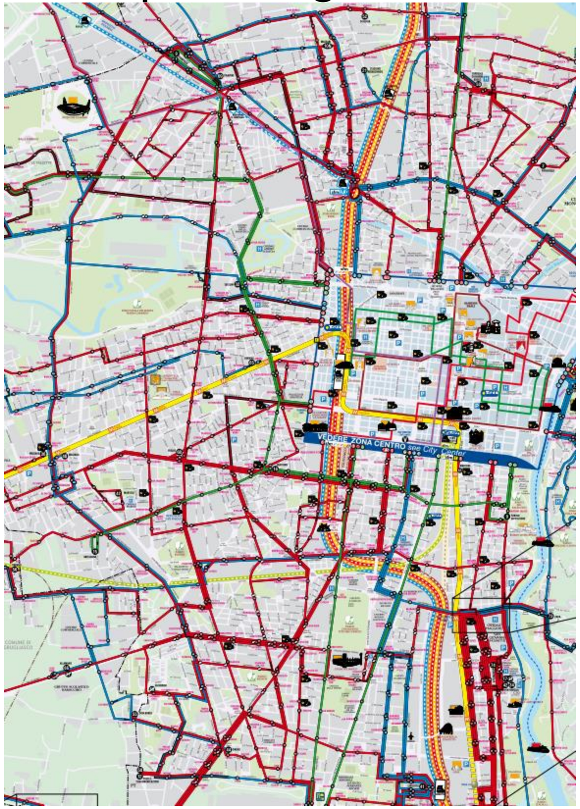
**Crediti liberi del 1 anno**

| Periodo | Codice | Lingua | Insegnamento | Crediti | Docente | Note | Vincoli |
|---|---|---|---|---|---|---|---|
| 2 | 01DDVOA | 🇬🇧 | Automotive evolution | 6 | G. Genta | (4) | [illegible] |
| 2 | 01OHOOA | 🇮🇹 | Chimica sperimentale per l'ingegneria | [illegible] | N. Penazzi | (3) | Si |
| 2 | 01OQCOA | 🇮🇹 | Etica | 6 | M. Ghisleni | (4) | Si |

https://didattica.polito.it/pls/portal30/gap.a_mds.espandi2?p_a_acc=2013&p_sdu=37&p_cds=3&p_header=&p_lang=IT

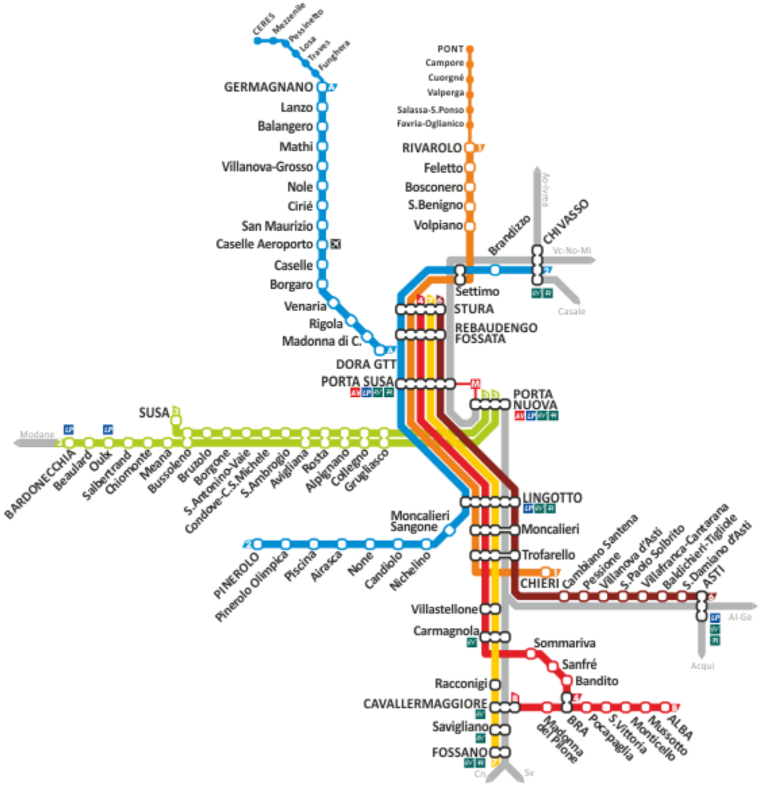
# Example: Turin public transportation

http://www.gtt.to.it/



GRUPPO TORINESE TRASPORTI

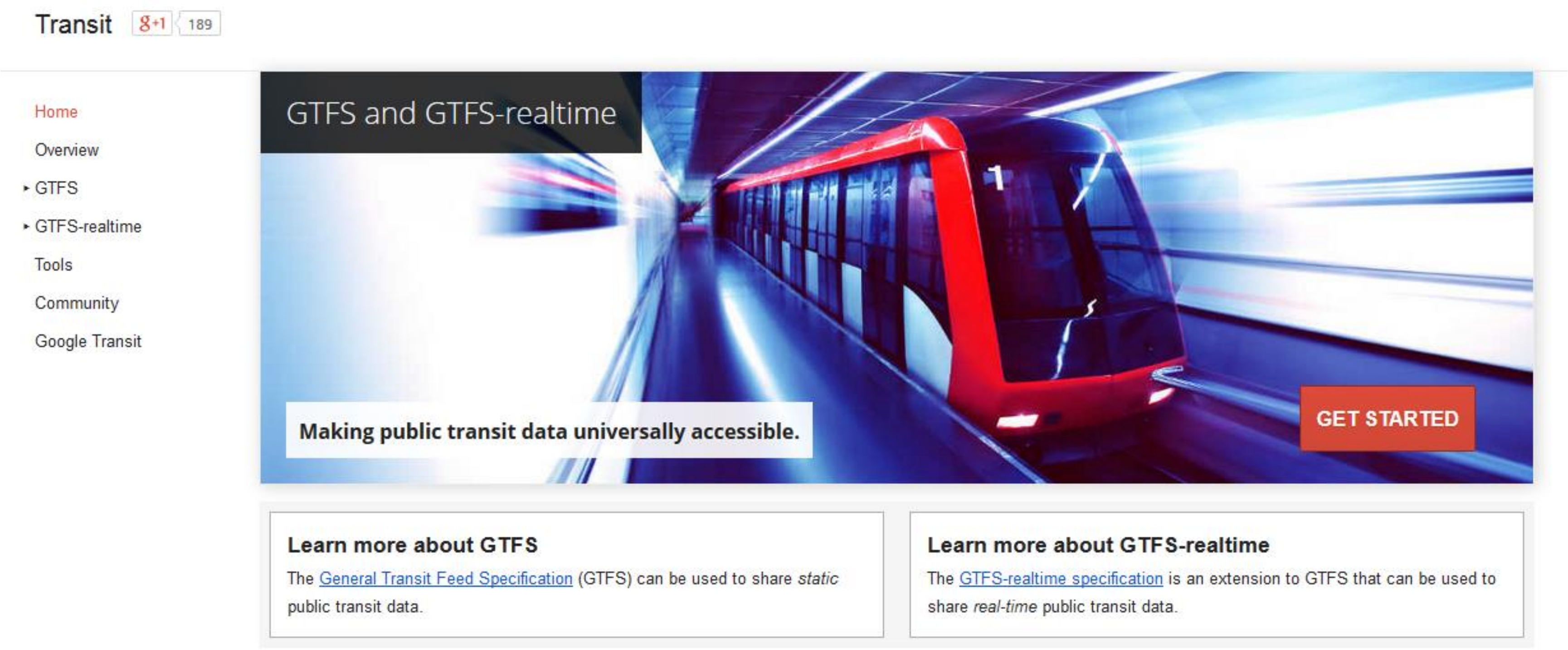
http://www.sfmtorino.it/



REGIONE PIEMONTE
sfm
servizio ferroviario metropolitano

# Google's GTFS standard

https://developers.google.com/transit/



Transit

GTFS and GTFS-realtime

Making public transit data universally accessible.

**Learn more about GTFS**

The General Transit Feed Specification (GTFS) can be used to share *static* public transit data.

**Learn more about GTFS-realtime**

The GTFS-realtime specification is an extension to GTFS that can be used to share *real-time* public transit data.

# GTFS Specification

| Filename | Required | Defines |
|---|---|---|
| **agency.txt** | Required | One or more transit agencies that provide the data in this feed. |
| **stops.txt** | Required | Individual locations where vehicles pick up or drop off passengers. |
| **routes.txt** | Required | Transit routes.  A route is a group of trips that are displayed to riders as a single service. |
| **trips.txt** | Required | Trips for each route.  A trip is a sequence of two or more stops that occurs at specific time. |
| **stop_times.txt** | Required | Times that a vehicle arrives at and departs from individual stops for each trip. |
| **calendar.txt** | Required | Dates for service IDs using a weekly schedule. Specify when service starts and ends, as well as days of the week where service is available. |
| calendar_dates.txt | Optional | Exceptions for the service IDs defined in the calendar.txt file. If calendar_dates.txt includes ALL dates of service, this file may be specified instead of calendar.txt. |
| fare_attributes.txt | Optional | Fare information for a transit organization's routes. |
| fare_rules.txt | Optional | Rules for applying fare information for a transit organization's routes. |
| shapes.txt | Optional | Rules for drawing lines on a map to represent a transit organization's routes. |
| frequencies.txt | Optional | Headway (time between trips) for routes with variable frequency of service. |
| transfers.txt | Optional | Rules for making connections at transfer points between routes. |
| feed_info.txt | Optional | Additional information about the feed itself, including publisher, version, and expiration information. |

https://developers.google.com/transit/gtfs/reference

# Where to find data?

http://opendata.5t.torino.it/gtfs/torino_it.zip

http://opendata.5t.torino.it/gtfs/sfm_torino_it.zip

Need more?

http://www.gtfs-data-exchange.com/

# 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)

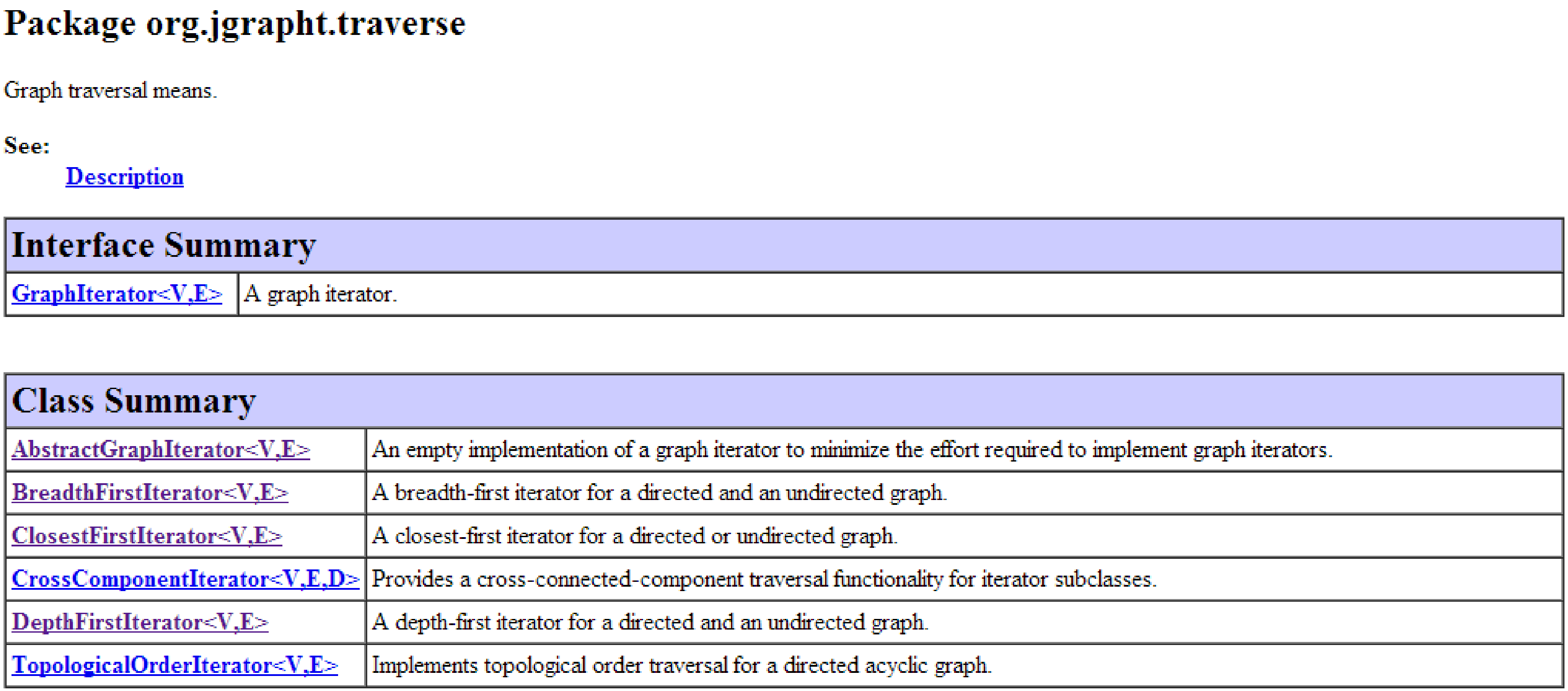
# Visits in JGraphT

## Representing and visiting graphs

# JGraphT and visits

- Visits are called "traversals"
- Implemented through **iterator** classes
- Package **org.jgrapht.traverse**

# Graph traversal classes



## Package org.jgrapht.traverse

Graph traversal means.

**See:**
Description

| Interface Summary | |
|---|---|
| GraphIterator<V,E> | A graph iterator. |

| Class Summary | |
|---|---|
| AbstractGraphIterator<V,E> | An empty implementation of a graph iterator to minimize the effort required to implement graph iterators. |
| BreadthFirstIterator<V,E> | A breadth-first iterator for a directed and an undirected graph. |
| ClosestFirstIterator<V,E> | A closest-first iterator for a directed or undirected graph. |
| CrossComponentIterator<V,E,D> | Provides a cross-connected-component traversal functionality for iterator subclasses. |
| DepthFirstIterator<V,E> | A depth-first iterator for a directed and an undirected graph. |
| TopologicalOrderIterator<V,E> | Implements topological order traversal for a directed acyclic graph. |

# Graph iterators

- Usual hasNext() and next() methods
- May register event listeners to traversal steps
  - void **addTraversalListener**(TraversalListener<V,E> l)
- TraversalListeners may react to:
  - Edge traversed
  - Vertex traversed
  - Vertex finished
  - Connected component started
  - Connected component finished

# Types of traversal iterators

- BreadthFirstIterator
- DepthFirstIterator
- ClosestFirstIterator
  - The metric for *closest* here is the path length from a start vertex. Graph.getEdgeWeight(Edge) is summed to calculate path length. Optionally, path length may be bounded by a finite radius.
- TopologicalOrderIterator
  - A topological sort is a permutation *p* of the vertices of a graph such that an edge {*i*,*j*} implies that *i* appears before *j* in *p*. Only directed acyclic graphs can be topologically sorted.

# Resources

- JGraphT Library: http://jgrapht.org/

# 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/