JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Graph.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/graphs/CallBack.html)
* [Next Class](http://docs.google.com/graphs/PrintCallBack.html)
* [Frames](http://docs.google.com/index.html?graphs/Graph.html)
* [No Frames](http://docs.google.com/Graph.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#3dy6vkm) |
* [Method](#4d34og8)

graphs

## Class Graph<E>

* java.lang.Object
  + graphs.Graph<E>
* Type Parameters:E -  
    
  public class Graph<E>  
  extends java.lang.Object  
  Implements a graph. We use two maps: one map for adjacency properties (adjancencyMap) and one map (dataMap) to keep track of the data associated with a vertex.Author: cmsc132

### Constructor SummaryConstructors

|  |
| --- |
| * + Constructor and Description |
| * + [**Graph**](http://docs.google.com/graphs/Graph.html#Graph())() Initializes the adjacency and data maps. |

### Method SummaryMethods

|  |  |
| --- | --- |
| * + Modifier and Type | * + Method and Description |
| * + void | * + [**addDirectedEdge**](http://docs.google.com/graphs/Graph.html#addDirectedEdge(java.lang.String,%20java.lang.String,%20int))(java.lang.String startVertexName, java.lang.String endVertexName, int cost) Adds or updates a directed edge with the specified cost. |
| * + void | * + [**addVertex**](http://docs.google.com/graphs/Graph.html#addVertex(java.lang.String,%20E))(java.lang.String vertexName, [E](http://docs.google.com/graphs/Graph.html) data) Adds a vertex to the graph by adding to the adjacency map an entry for the vertex. |
| * + void | * + [**doBreadthFirstSearch**](http://docs.google.com/graphs/Graph.html#doBreadthFirstSearch(java.lang.String,%20graphs.CallBack))(java.lang.String startVertexName, [CallBack](http://docs.google.com/graphs/CallBack.html)<[E](http://docs.google.com/graphs/Graph.html)> callback) Computes Breadth-First Search of the specified graph. |
| * + void | * + [**doDepthFirstSearch**](http://docs.google.com/graphs/Graph.html#doDepthFirstSearch(java.lang.String,%20graphs.CallBack))(java.lang.String startVertexName, [CallBack](http://docs.google.com/graphs/CallBack.html)<[E](http://docs.google.com/graphs/Graph.html)> callback) Computes Depth-First Search of the specified graph. |
| * + int | * + [**doDijkstras**](http://docs.google.com/graphs/Graph.html#doDijkstras(java.lang.String,%20java.lang.String,%20java.util.ArrayList))(java.lang.String startVertexName, java.lang.String endVertexName, java.util.ArrayList<java.lang.String> shortestPath) Computes the shortest path and shortest path cost using Dijkstras's algorithm. |
| * + java.util.Map<java.lang.String,java.lang.Integer> | * + [**getAdjacentVertices**](http://docs.google.com/graphs/Graph.html#getAdjacentVertices(java.lang.String))(java.lang.String vertexName) Returns a map with information about vertices adjacent to vertexName. |
| * + int | * + [**getCost**](http://docs.google.com/graphs/Graph.html#getCost(java.lang.String,%20java.lang.String))(java.lang.String startVertexName, java.lang.String endVertexName) Returns the cost associated with the specified edge. |
| * + [E](http://docs.google.com/graphs/Graph.html) | * + [**getData**](http://docs.google.com/graphs/Graph.html#getData(java.lang.String))(java.lang.String vertex) Returns the data component associated with the specified vertex. |
| * + java.util.Set<java.lang.String> | * + [**getVertices**](http://docs.google.com/graphs/Graph.html#getVertices())() Returns a Set with all the graph vertices. |
| * + java.lang.String | * + [**toString**](http://docs.google.com/graphs/Graph.html#toString())() Returns a string with information about the Graph. |

### Methods inherited from class java.lang.Objectequals, getClass, hashCode, notify, notifyAll, wait, wait, wait

### Constructor Detail

#### Graph public Graph() Initializes the adjacency and data maps.

### Method Detail

#### addVertex public void addVertex(java.lang.String vertexName, [E](http://docs.google.com/graphs/Graph.html) data) Adds a vertex to the graph by adding to the adjacency map an entry for the vertex. This entry will be an empty map. An entry in the dataMap will store the provided data.Parameters:vertexName - vertex's namedata - data associated with the vertex Throws: java.lang.IllegalArgumentException - If the vertex already exists in the graph. Use any error message.

#### addDirectedEdge public void addDirectedEdge(java.lang.String startVertexName, java.lang.String endVertexName, int cost) Adds or updates a directed edge with the specified cost.Parameters:startVertexName - endVertexName - cost - Throws: java.lang.IllegalArgumentException - If any of the vertices are not part of the graph. Use any error message.

#### toString public java.lang.String toString() Returns a string with information about the Graph. Notice that vertices are printed in sorted order and information about adjacent edges is printed in sorted order (by vertex name). You may not use Collections.sort or Arrays.sort in order to implement this method. See the sample output for formatting details. return string with graph information**Overrides:** toString in class java.lang.Object

#### getAdjacentVertices public java.util.Map<java.lang.String,java.lang.Integer> getAdjacentVertices(java.lang.String vertexName) Returns a map with information about vertices adjacent to vertexName. If the vertex has no adjacents, an empty map is returned.Parameters:vertexName - Returns:map

#### getCost public int getCost(java.lang.String startVertexName, java.lang.String endVertexName) Returns the cost associated with the specified edge.Parameters:startVertexName - endVertexName - Returns:edge cost Throws: java.lang.IllegalArgumentException - If any of the vertices are not part of the graph. Use any error message.

#### getVertices public java.util.Set<java.lang.String> getVertices() Returns a Set with all the graph vertices.Returns:set with vertices.

#### getData public [E](http://docs.google.com/graphs/Graph.html) getData(java.lang.String vertex) Returns the data component associated with the specified vertex.Parameters:vertex - Returns:data Throws: java.lang.IllegalArgumentException - If the vertex is not part of the graph. Use any error message.

#### doDepthFirstSearch public void doDepthFirstSearch(java.lang.String startVertexName, [CallBack](http://docs.google.com/graphs/CallBack.html)<[E](http://docs.google.com/graphs/Graph.html)> callback) Computes Depth-First Search of the specified graph.Parameters:startVertexName - callback - Represents the processing to apply to each vertex Throws: java.lang.IllegalArgumentException - If the vertex is not part of the graph. Use any error message.

#### doBreadthFirstSearch public void doBreadthFirstSearch(java.lang.String startVertexName, [CallBack](http://docs.google.com/graphs/CallBack.html)<[E](http://docs.google.com/graphs/Graph.html)> callback) Computes Breadth-First Search of the specified graph.Parameters:startVertexName - callback - Represents the processing to apply to each vertex Throws: java.lang.IllegalArgumentException - If the vertex is not part of the graph. Use any error message.

#### doDijkstras public int doDijkstras(java.lang.String startVertexName, java.lang.String endVertexName, java.util.ArrayList<java.lang.String> shortestPath) Computes the shortest path and shortest path cost using Dijkstras's algorithm. It initializes shortestPath with the names of the vertices corresponding to the shortest path. If there is no shortest path, shortestPath will be have entry "None".Parameters:startVertexName - endVertexName - shortestPath - Initialized by the method with the shortest path or "None". Returns:Shortest path cost or -1 if no path exist. Throws: java.lang.IllegalArgumentException - If any of the vertices are not part of the graph. Use any error message.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Graph.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* [Prev Class](http://docs.google.com/graphs/CallBack.html)
* [Next Class](http://docs.google.com/graphs/PrintCallBack.html)
* [Frames](http://docs.google.com/index.html?graphs/Graph.html)
* [No Frames](http://docs.google.com/Graph.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* [Constr](#3znysh7) |
* [Method](#2et92p0)
* Detail:
* Field |
* [Constr](#3dy6vkm) |
* [Method](#4d34og8)