Class Graph

java.lang.Object
    Graph

```
public class Graph
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
`private java.util.Map<java.lang.String,java.util.Set<java.lang.String>>`	`adj`
`private int`	`E`
`private int`	`V`

Constructor Summary

Constructors

Constructor and Description
`Graph()` Initializes an empty graph

Method Summary

All Methods   Instance Methods   Concrete Methods

Modifier and Type	Method and Description
`void`	`addEdge(java.lang.String v, java.lang.String w)` Adds the undirected edge v-w to this graph.
`boolean`	`addVertex(java.lang.String v)` Adds the vertex v to this graph
`int`	`degree(java.lang.String v)` Returns the degree of vertex v.
`int`	`edges()` Returns the number of edges in this graph.
`java.util.Iterator<java.lang.String>`	`getAdjacent(java.lang.String v)` Returns the vertices adjacent to vertex v.
`java.lang.String`	`toString()` Returns a string representation of this graph.

`private void`	`validateVertex(java.lang.String v)` Ensures the argument is a valid vertex in the graph
`boolean`	`validVertex(java.lang.String v)` Is this a valid vertex in the graph?
`int`	`vertices()` Returns the number of vertices in this graph.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

V

```
private int V
```

E

```
private int E
```

adj

```
private java.util.Map<java.lang.String,java.util.Set<java.lang.String>> adj
```

Constructor Detail

Graph

```
public Graph()
```

Initializes an empty graph

Method Detail

vertices

```
public int vertices()
```

Returns the number of vertices in this graph.

Returns:

`the number of vertices in this graph`

edges

```
public int edges()
```

Returns the number of edges in this graph.

Returns:

`the number of edges in this graph`

validVertex

```
public boolean validVertex(java.lang.String v)
```

Is this a valid vertex in the graph?

Parameters:

`v - vertex to be tested`

Returns:

`true if vertex is valid, false otherwise`

validateVertex

```
private void validateVertex(java.lang.String v)
```

Ensures the argument is a valid vertex in the graph

Parameters:

`v - one vertex in the graph`

Throws:

`java.lang.IllegalArgumentException - if v is not a valid vertex`

addVertex

```
public boolean addVertex(java.lang.String v)
```

Adds the vertex v to this graph

Parameters:

`v - one vertex in the graph`

Returns:

`true if v was added, false otherwise`

addEdge

```
public void addEdge(java.lang.String v,
                    java.lang.String w)
```

Adds the undirected edge v-w to this graph. The arguments must be valid vertices in the graph.

Parameters:

v - one vertex in the edge

w - the other vertex in the edge

Throws:

java.lang.IllegalArgumentException - if either vertex does not exist

getAdjacent

```
public java.util.Iterator<java.lang.String> getAdjacent(java.lang.String v)
```

Returns the vertices adjacent to vertex v.

Parameters:

v - the vertex

Returns:

an Iterator containing the vertices adjacent to vertex v

Throws:

java.lang.IllegalArgumentException - if v is not a valid vertex

degree

```
public int degree(java.lang.String v)
```

Returns the degree of vertex v.

Parameters:

v - the vertex

Returns:

the degree of vertex v

Throws:

java.lang.IllegalArgumentException - if v is not a valid vertex

toString

```
public java.lang.String toString()
```

Returns a string representation of this graph.

Overrides:

toString in class java.lang.Object

Returns:

the number of vertices *V*, followed by the number of edges *E*, followed by the *V* adjacency lists