## Graph Implementation

## Class Diagram

```
V, E
                        Graph
+ Graph()
+ clear()
                                                                   «interface»
                                                                   Iterable
+ clearEdges()
+ vertexCount(): int
+ edgeCount(): int
                                                                DuplicateVertexException
+ addVertex(v: Vertex<V>)
+ addVertices(vertices: ArrayList<Vertex<V>>)
+ containsVertex(v: Vertex<V>): boolean
                                                                 DuplicateEdgeException
+ addEdge(e: Edge<V, E>)
                                                                  NoSuchVertexException
+ addEdges(edges: ArrayList<Edge<V, E>>)
+ containsEdge(e: Edge<V, E>): boolean
                                                                   NoSuchEdgeException
+ removeVertex(v: Vertex<V>): ArrayList<Edge<V, E>>
+ removeEdge(e: Edge<V, E>)
                                                                           «interface»
+ adjacentCount(v: Vertex<V>): int
                                                                          Comparable
+ vertices(): ArrayList<Vertex<V>>
+ adjacent(v: Vertex<V>): ArrayList<Vertex<V>>
+ edges(): ArrayList<Edge<V, E>>
+ iterator(): Iterator<Vertex<V>>
                                                                             Vertex
+ minSpanningTree(v: Vertex<V>): Tree<V, E>
                                                              + Vertex(label: V)
+ totalVertexWeight(): double
                                                              + Vertex(label: V, weight: double)
+ totalEdgeWeight(): double
                                                              + getLabel(): V
+ canReach(v1: Vertex<V>, v2: Vertex<V>): boolean
                                                              + getWeight(): double
+ hasCycle(): boolean
                                                              + setLabel(label: V)
                                                              + setWeight(weight: double)
                                        V, E
                   Tree
                                                              + compareTo(other: Vertex<V>): int
                                                              + toString(): String
+ Tree()
- setParent(parent: Tree<V, E>)
+ getParent(): Tree<V, E>
                                                                                              V, E
+ getVertex(): Vertex<V>
                                                                     Edge
+ isRoot(): boolean
                                               Edge(v1: Vertex<V>, v2: Vertex<V>, label: E)
+ isLeaf(): boolean
                                               Edge(v1: Vertex<V>, v2: Vertex<V>, lable: E,
+ isAncestorOf(tree: Tree<V, E>): boolean
+ isSiblingOf(tree: Tree<V, E>): boolean
                                               weight: double)
                                               getVertex1(): Vertex<V>
+ childCount(): int
+ size(): int
                                               getVertex2(): Vertex<V>
+ clear()
                                               getLabel(): E
                                               getWeight(): double
+ height(): int
                                               setVertex1(v: Vertex<V>)
+ level(): int
                                               setVertex2(v: Vertex<V>)
+ add(child: Tree<V, E>)
                                               setLabel(label: E)
+ remove(child: Tree<V, E>)
                                               setWeight(weight: double)
+ isBalanced(): boolean
                                               equals(other: Edge<V, E>): boolean
+ pathCost(tree: Tree<V, E>): double
                                               toString(): String
+ isDegenerate(): boolean
+ makeNewRoot(t: Tree<V, E>)
```

## 1 Graph and Tree Example

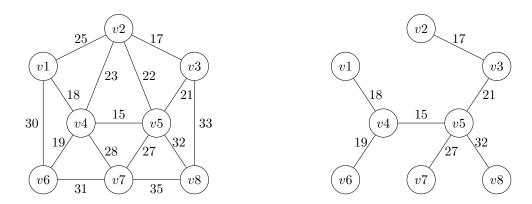


Figure 1: A graph and its minimum spanning tree.