Lab : Generics

Exercise 1 – Directed Unweighted graph with adjacency list

2. Implement this interface by defining the class ListGraph. This class contains a field representing the adjacency list of the graph. What is the type of this field ?

We defined the adjacency list as a HashMap of Vertex (as key) and Set of vertices (as mapped values).

```
HashMap<V, Set<V>> adjacencyList;
```

Exercise 2 – Depth-First-Search

— Why do we not parameterize the class DepthFirstSearch ?

We do not parameterize the class DepthFirstSearch because this algorithm is not specific to a type. A parameterized class is useful to define the behavior for Sets in general. However, the class DepthFirstSearch is not defined by a specific collection type (Set), it uses V as signature but the format of the formal type parameter is not fixed.
If we changed the signature V to E in this method, it would make no difference to how it is used nor to its functionality.

Exercise 4 – Graphs of (colored) words (Inner classes & Wildcards)

2. Extend this class with the class ColoredWord which contains a unique field to represent the color of the word. A word can be colorized in red, green, blue, yellow, purple or turquoise. In order to correctly represent the field of ColoredWord, write an enum Color to represent its type. Because this enum makes no sense outside of the class ColoredWord, declare it as an inner class. Should it be static or not ?

It should be static because an instance of the outer class ColoredWord is needed to instantiate an instance of the inner class Color. The inner class Color is bounded to the outer class.

5. Implement this method. In which class you have to do this ?

We must implement this method in AbstractListGraph because it applies to both DirectedListGraph and UndirectedListGraph, in order to factorise the code.

6. Use this method to add a Set of Word in a graph. Now add a Set of ColoredWord in the graph. Why it does not compile ? Use the wildcard syntax to make it compile.

When we try to use this method to add a Set of Word or a Set of ColoredWord, it does not compile because the function expects a Set of Vertex. It does not recognise the type of ColoredWord or Word. To avoid this problem, we use a wildcard type. It is used for creating reusable methods that operate on collections of a specific type (in our case, we are using a Set). Since we do not know what type the Set is typed to (Vertex, Word or ColoredWord), we only read from the collection, and treat the parameters read as being Object instances.

Exercise 5 – Directed Weighted graph with adjacency list

4. You must override this method in AbstractWeightedListGraph to refine the display (add the label of the edge). Because the method edgeString has only two vertices as arguments, the implementation of this method in AbstractWeightedListGraph must instantiate a new Tuple using these two vertices to find in corresponding edge in the Map. Which methods must be overridden to make it work ? Override it to make it work.

AbstractWeightedListGraph must override the method edgeString to make it work, since it needs a Tuple to get the weight of the edge.