Diego Nicodemo Santosuosso
EECS Username: nicodemo
EECS3311 Lab 2

Building an Analyzer for a Simple Programming Language Design Report

Lab 2

In order to design my software, I divided my work process into three main components: Preparation, DGraph Interface development, and Analyzer development. The development of the initial components were crucial to the development of the next ones.

(1) Preparation:

The very first step of my software development process was to prepare for its implementation. I first had to review all the concepts involved in the project. I reviewed the basics of linked lists, graph theory and graph traversals. I watched a few tutorials on Youtube explaining their theoretical applications and the way the different data types are supposed to work. I read the lab instructions a couple of times to ensure that I understood every task.

After completely grasping into the theoretical components of the project, I started studying the starter code. I went through each package and each class, carefully understanding how they were supposed to be implemented. I went through the *DGraph* interface and studied each of their abstract methods in order to understand exactly how they were supposed to function. I read their documentation, function definitions and parameters.

Only after I was finished studying the specifications, the core concepts and the starter code; I moved to the next stage of my software development process - developing the *DGraph* interface.

(2) DGraph Interface Development (graph package)

Before starting to implement the *ListDGraph* class - which extends the *DGraph* interface, I started by developing the *Edge* and the *Vertex* classes. I carefully developed each of the #TODO methods specified by our instructor and I tested them on the *StudentTest* class to ensure that they worked as desired.

When I finished developing these two classes, I started developing the *ListDGraph* class by following the instructions of its *DGraph* interface documentation. The *branches* method was definitely the hardest one to implement as it required a Depth First Search graph traversal algorithm in order to generate the different branches of the directed graphs.

However, after many hours of work, I was able to complete its implementation and to finish the development of the *ListDGraph* class. At this point, I created numerous unit tests in the StudentTest class to ensure that each class had a coverage of at least 80% and to ensure that each implemented method was working correctly.

(3) Analyzer Development (analyzer package)

Once I finished developing the graph package and all its classes (ListDGraph, Edge and Vertex), I started implementing the *CFGAnalyzer* class along with the main method of the *Main* class. This was definitely the most challenging task of the assignment. The analyzer read a Python program and created a context free graph from it.

By encapsulating its construction, the CFGAnalyzer class was developed in steps. The various steps were (1) defining how to read the Python keywords (def, if, for, else, etc), (2) loading the data, (3) reading the Python file and storing each line into a list, (4) and then running the algorithm to identify boundaries and create the respective vertices and edges. This is an example of a Builder design pattern implementation in which a step-by-step process using the correct sequence of actions allows for the creation of a complex object.

Class Diagram