Java Parser for UML Class diagram

Name : Harsha Muktamath

SJSU ID : 011825138

My YouTube Link – https://youtu.be/nosPm36FwT8

Github Link - https://github.com/muktamath81/UML-Parser

Kanban Link - https://waffle.io/muktamath81/UML-Parser

Google Drive link for zip files -
https://drive.google.com/open?id=0B4wVdfCPs4VhZVNmZkdXMlh5Vzg

For Extra Credit (Partnering with CMPE 281 students)

1) **Tuan Ung (SJSU ID – 007671040)**
   YouTube video https://www.youtube.com/watch?v=J1p6r9cU8-8
2) **Tanmay Bhatt (SJSU ID – 011499072)**
   YouTube video https://www.youtube.com/watch?v=LcwwtrVyi1k&feature=youtu.be

Initial setup -

• Install Java and use any available IDE for development. In my project, I have used Eclipse and to manage dependencies Maven.

• GraphViz must be installed on the system. It does not have plugins, hence it has to be installed (Link- http://www.graphviz.org/Download..php)

Instructions to execute the program

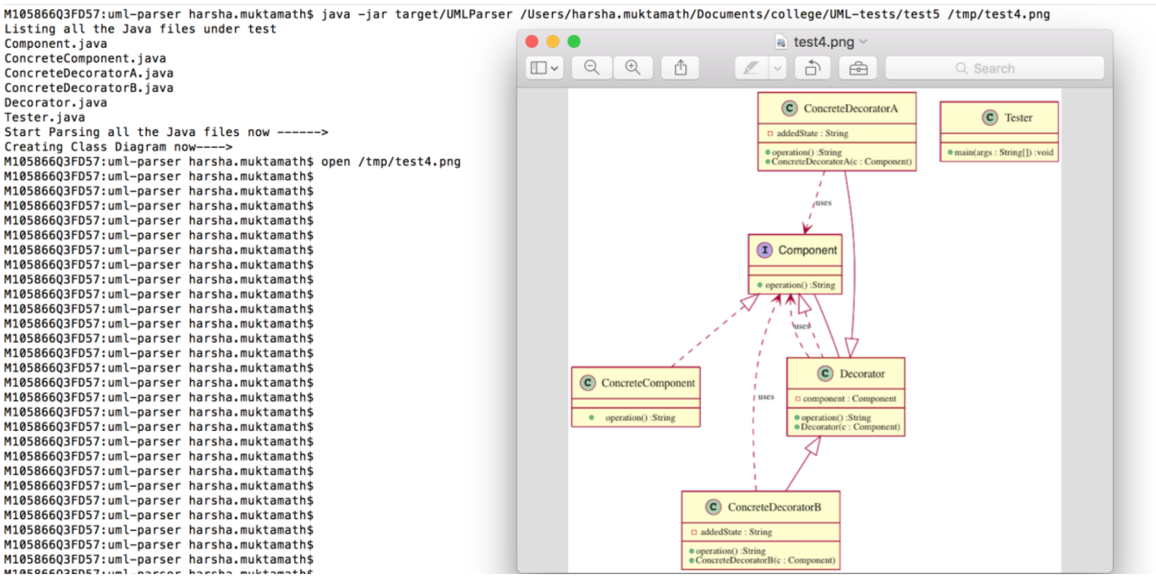
1) Download the package
2) Download the test files (test*.zip) and unzip it to a location (here – sourcefilepath)
3) Execute the code as below

• java –jar <jar file/Renamed file> <sourcefilepath> <outputfilenameandpath>

◦ sourcefilepath - Path to the default package folder.

◦ outputfilenameandpath – filename and path of the output image

In my example, I have executed as below (Here I have renamed *.jar file to UMLParser and executed)



External Libraries and Tools used

1. Javaparser

   Javaparser is lightweight and easy to use parser library which parses java code and provides AST (Abstract Syntax Tree). Javaparser uses javacc (Java Compiler Compiler) for generating AST from Java code. One can analyze code structure, Javadoc or comments using AST created by Javaparser library. GraphViz is open source graph visualization software. GraphViz supports dot(.) notation input for drawing directed graphs. The GraphViz software takes input as simple text file and converts it to diagram. GraphViz provides support for generating output diagram in PDF, Images or SVG format. It has many useful features such as custom coloring, shapes and custom messages.

2. GraphViz

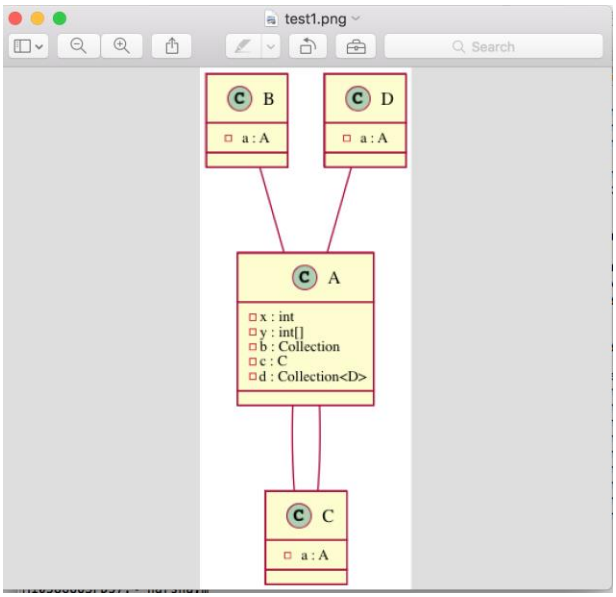
   Limitation: GraphViz application is stand-alone application and cannot be integrated in java project without third party library.

3. PlantUML

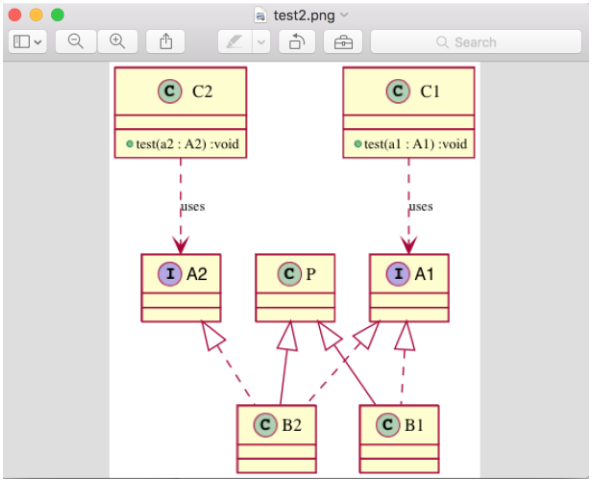
   PlantUML provides a Java based API for using GraphViz software through your Java application. This library is required because GraphViz does not have their Java API exposed which developers can use to integrate GraphViz directly in their application. As PlantUML uses GraphViz internally, most of the functions that are offered by GraphViz are supported by PlantUML.

   Sample Output from my UML Parser code –

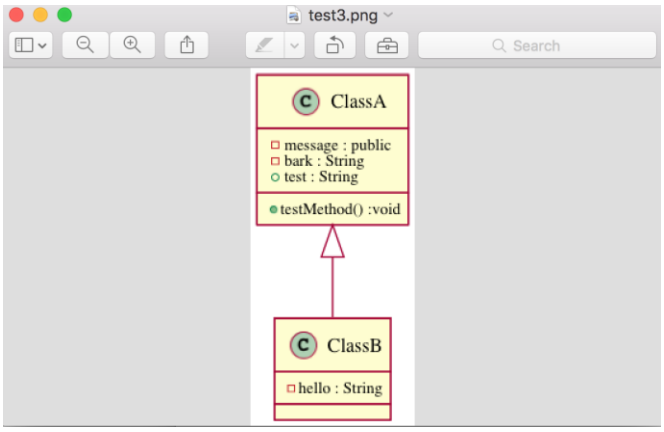
Test Case 1:



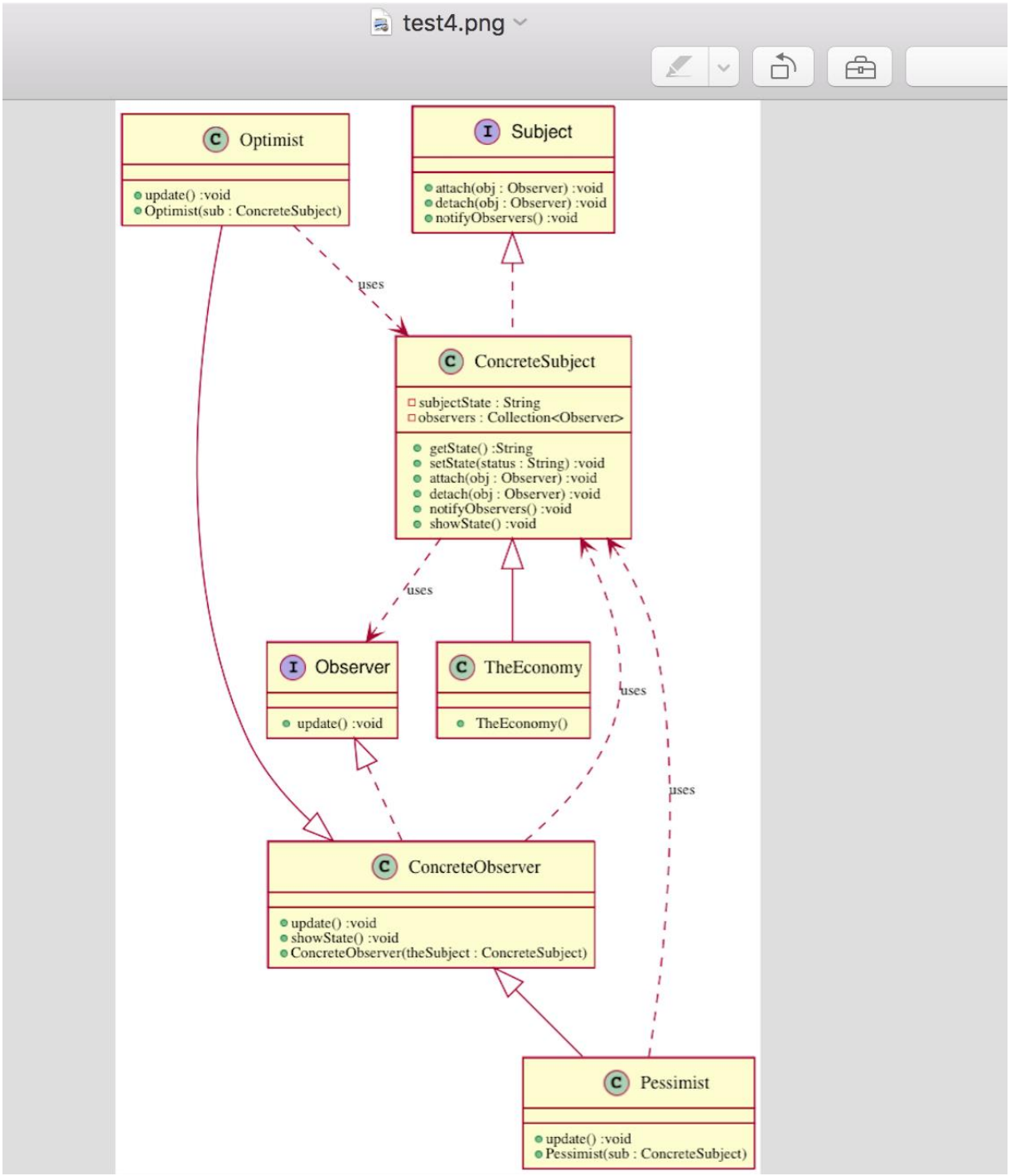
Test Case 2:



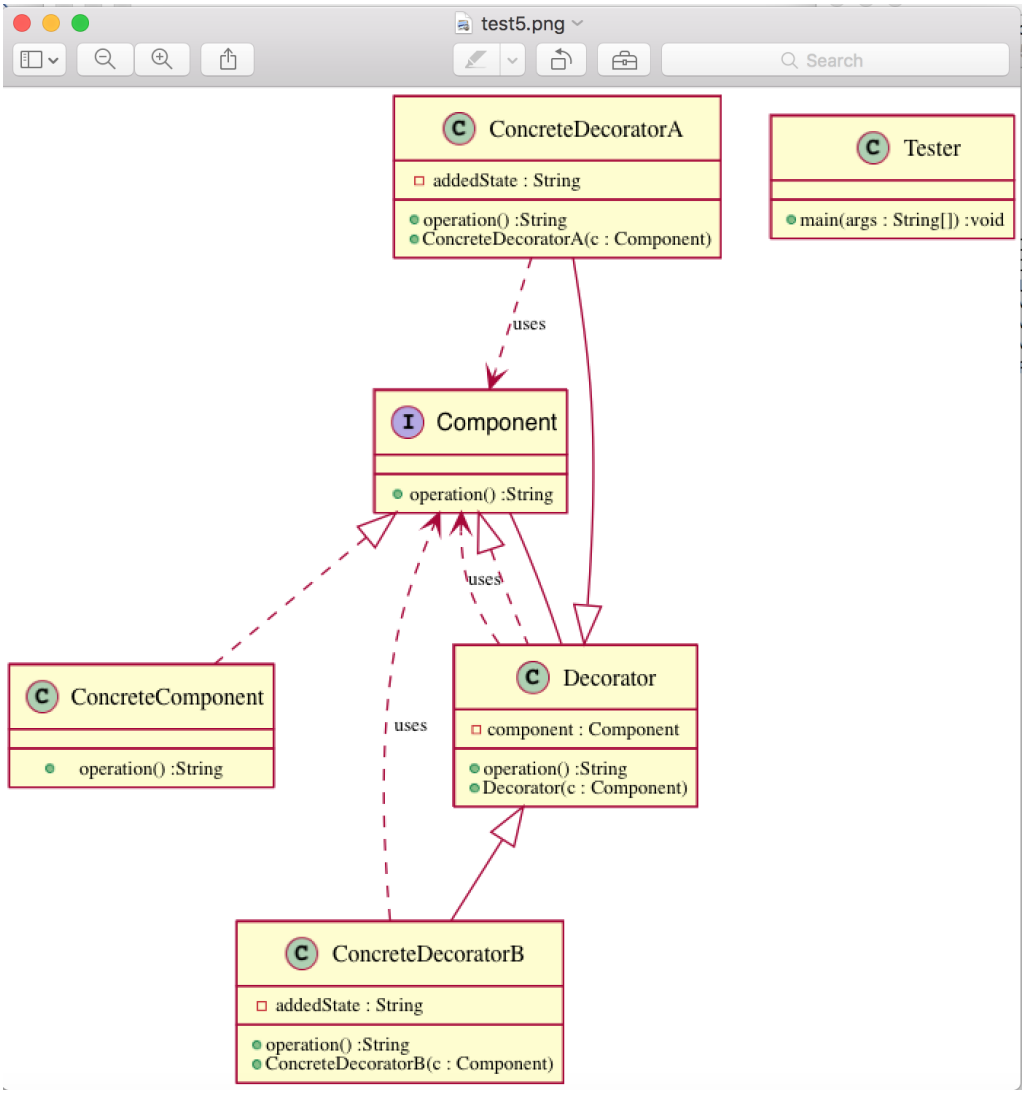
Test Case 3:


test3.png
Search
ClassA
message : public
bark : String
test : String
testMethod() :void
ClassB
hello : String

Test Case 4:


test4.png
Optimist
update() :void
Optimist(sub : ConcreteSubject)
Subject
attach(obj : Observer) :void
detach(obj : Observer) :void
notifyObservers() :void
uses
ConcreteSubject
subjectState : String
observers : Collection<Observer>
getState() :String
setState(status : String) :void
attach(obj : Observer) :void
detach(obj : Observer) :void
notifyObservers() :void
showState() :void
uses
Observer
update() :void
TheEconomy
TheEconomy()
uses
uses
ConcreteObserver
update() :void
showState() :void
ConcreteObserver(theSubject : ConcreteSubject)
Pessimist
update() :void
Pessimist(sub : ConcreteSubject)

Test Case 5:


test5.png
Search
ConcreteDecoratorA
addedState : String
operation() :String
ConcreteDecoratorA(c : Component)
Tester
main(args : String[]) :void
uses
Component
operation() :String
uses
ConcreteComponent
operation() :String
uses
Decorator
component : Component
operation() :String
Decorator(c : Component)
ConcreteDecoratorB
addedState : String
operation() :String
ConcreteDecoratorB(c : Component)