CMPE 202 - Personal Project - UML Parser

PROJECT DESCRIPTION

Objectives:

In this project, you will be applying what you learned about UML Class Diagram by creating a Parser which converts Java (or alternative OOP language) Source Code into a UML Diagram.

Requirements:

- The programming language, tools, sdk, libraries that you chose to use is entirely up to you.
- The parser must be executable on the command line with the following format:
  - **umlparser** <classpath> <output file name>
  - <classpath> is a folder name where all the .java source files will be
  - <output file name> is the name of the output image file you program will generate (.jpg, .png or .pdf format)
- All the Java source files will be in the "default" package. That is, there will be no sub-directories in the classpath.
- You must register for a "free" account on **https://kanbanflow.com/** (https://kanbanflow.com/) and maintain a weekly update of your Kanban Board. The content of Board is up to you to plan, but you must track your weekly count of number of cards "completed" and "in-progress". This can be done by simply taking a "snapshot" of your board for submission.

Grading:

- **40 points** will be based on 4 weekly submissions of Kanban board and report
  - Weekly Kanban board submissions must be submitted on-time on Canvas
  - Missing the due date will result in a -5 point penalty (within the 24-hour late window)
  - Missing the late window will result in "Zero Points"
- **50 points** of your project will be graded based on 5 test cases.
  - You must submit output files and the source code for your project.
  - Your output for each test case should match at least 80% of what is expected for full credit
- **10 points** for a short one page PDF file describing the libraries and tools you used to create the parser.

- *If you don't submit your source code, or if your source code is found to be similar to another student's work, your final project submission with be worth zero out of 60.*

Challenge Problems (Extra Credit - Select One / Not Both):

- Implement your UML Parser to parse in a different OOP language (instead of Java Source). You must create the 5 test case input source file similar in structure to the Java Source Code such that the UML Class Diagram output produces the same expected result. **(20 points)**

- Extended your UML Parser to perform **dynamic analysis** of the Java Source code to generate a UML Sequence Diagram. The UML Sequence Diagram must be generated by executing the Java Source Code. Extra credit must be demonstrated "live" in class. Use the Observer Test Case available here: **https://github.com/paulnguyen/design/tree/master/umlparser/uml-sequence-test (https://github.com/paulnguyen/design/tree/master/umlparser/uml-sequence-test)** . **(20 points)**

Hints for Parser:

- **Default Package:** All Java source files will be in the "default" package. That is, there will be only one directory (i.e. no subdirectories)
- **Dependencies & Uses Relationships for Interfaces Only:** Do not include dependencies in output UML diagram except in the cases of "interfaces/uses"
- **Class Declarations with optional Extends and Implements**
- **Only Include Public Methods** (ignore private, package and protected scope)
- **Only Include Private and Public Attributes** (ignore package and protected scope)
- **Java Setters and Getters:** Must Support also Java Style Public Attributes as "setters and getters"
- **Must Include Types for Attributes, Parameters and Return types on Methods**
- **Classifier vs Attributes Compartment:** If there is a Java source file, then there should be a "UML Class" on the Diagram for it. That is, if there is no Java source file for a class and the class is part of an instance variable, put the class/property in the attribute compartment
- **Interfaces - Implements and Uses Notation:** Show Interfaces along with Clients of Interfaces (as dependencies).

Notes and Exclusions:

- **Static and Abstract Notation**: Static and Abstract notation in UML are usually denoted as "underline" and "italic", but rarely used in practice. Parsing this is not a requirement for this project.
- **Relationships Between Interfaces**: Although conceptually possible in UML, relationships between Interfaces (i.e. inheritance, dependencies) are rarely thought of in practice and generally bad practice. As such, this project does not expect parser to detect these situations. Focus you work on the relationships and dependencies from Classes to Interfaces.
- **For Additional Pointers on UML Notation**, see lecture notes and **http://www.uml-diagrams.org/class-reference.html (http://www.uml-diagrams.org/class-reference.html)**

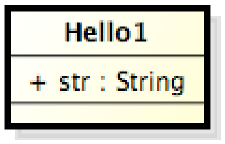
Example for How to Parse Java Setters/Getters:

The expected output for Class Source with Java Setters/Getters is to interpret as a "Public Attribute" instead of Private with public setter/getter methods. Also, the public setter and getter methods should not be visible (i.e. not on class diagram) in the operations compartment.

For example Hello1.java :

```
class Hello1 {
    private String str ;
    public String getStr() { return this.str ; }
    public void   setStr(String value) { this.str = value ; }
}
```

Should be drawn as:



Recommended Tools:

You may use any tools (java based, or other languages). Tools for parsing Java Source and generating UML diagrams can be used.

For example (but not limited to):

- **Parsers:**
  - **https://github.com/javaparser/javaparser (https://github.com/javaparser/javaparser)**
  - **https://javacc.java.net (https://javacc.java.net)**
  - **http://beaver.sourceforge.net (http://beaver.sourceforge.net)**
  - **http://www.antlr.org (http://www.antlr.org/)**
- **UML Generators:**
  - **http://yuml.me/diagram/scruffy/class/draw (http://yuml.me/diagram/scruffy/class/draw)**
  - **http://bramp.github.io/js-sequence-diagrams (http://bramp.github.io/js-sequence-diagrams)**
  - **http://www.umlgraph.org (http://www.umlgraph.org)**
  - **http://plantuml.com (http://plantuml.com)**

PROJECT TEST CASES

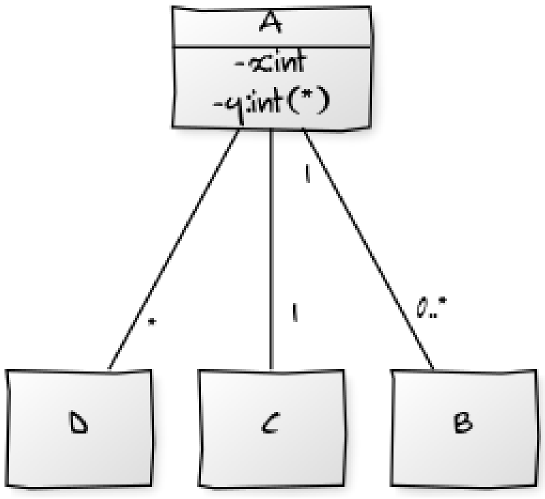
Test Case #1:

- **Java Source Code:** uml-parser-test-1.zip (https://sjsu.instructure.com/courses/1188568/files/42435055/download?wrap=1) (https://sjsu.instructure.com/courses/1188568/files/42435055/download?wrap=1)
- **Sample yUML "Intermediate" Code from Parser:**

[A|-x:int;-y:int(*)]1-0..*[B],

[A]-1[C],
[A]-*[D]

- **Sample yUML Output (Resulting in Diagram):**
  **** Note: yUML uses brackets and so parenthesis are used instead for arrays ****



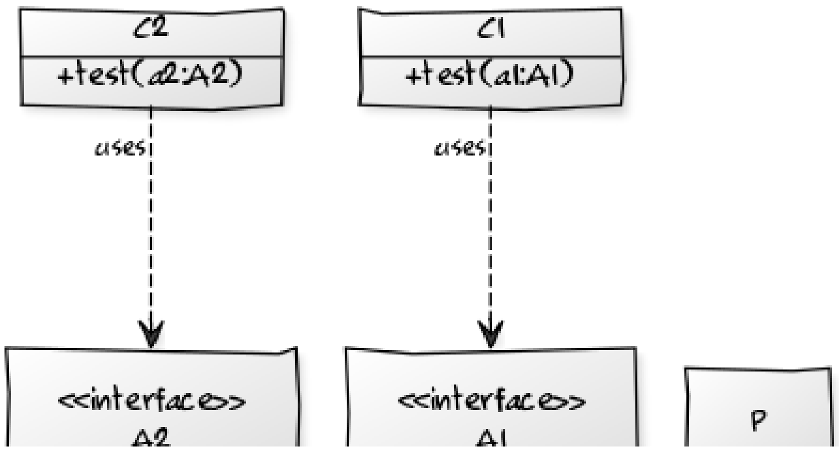
Test Case #2: [uml-parser-test-2.zip (https://sjsu.instructure.com/courses/1188568/files/42887670/download?wrap=1)](https://sjsu.instructure.com/courses/1188568/files/42887670/download?wrap=1) [(https://sjsu.instructure.com/courses/1188568/files/42887670/download?wrap=1)](https://sjsu.instructure.com/courses/1188568/files/42887670/download?wrap=1)

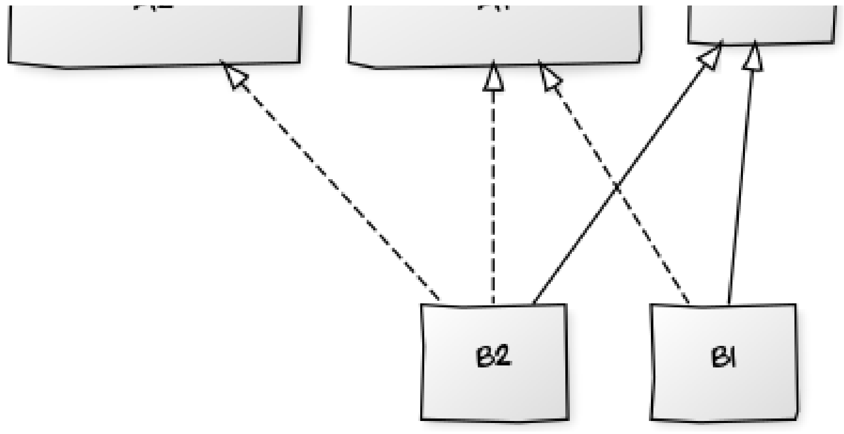
- **Java Source Code:**

- **Sample yUML "Intermediate" Code from Parser:**

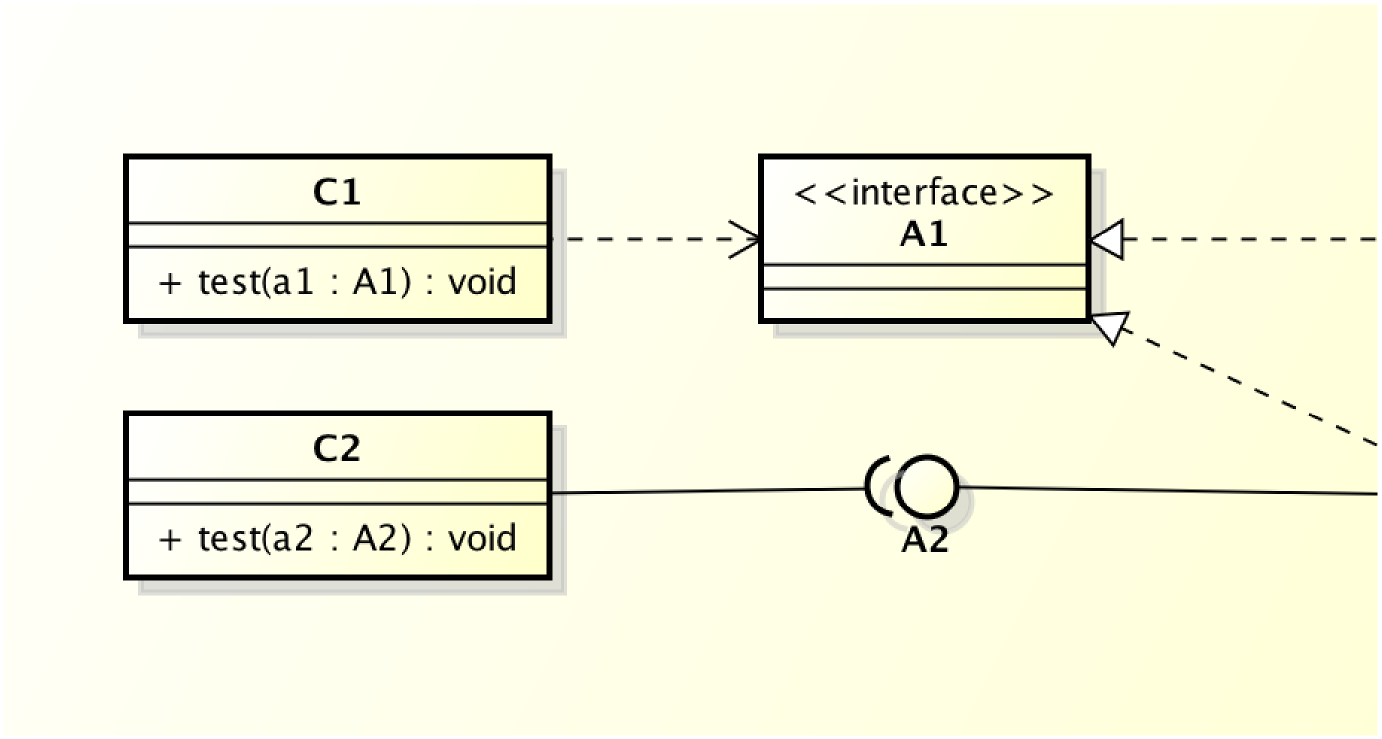
  [C1|+test(a1:A1)]uses -.->[<<interface>>;A1],
  [<<interface>>;A1]^-.-[B1],
  [<<interface>>;A1]^-.-[B2],
  [P]^-[B1],
  [P]^-[B2],
  [<<interface>>;A2]^-.-[B2],
  [C2|+test(a2:A2)]uses -.->[<<interface>>;A2]

- **Sample yUML Output (resulting in Diagram):**



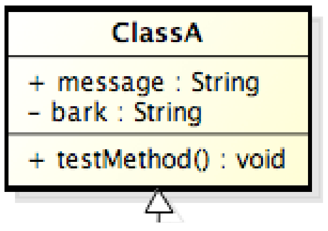


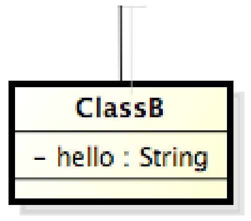
- **Sample Astah UML Model (With C2, A2 and B2 in Ball & Socket Notation):**



Test Case #3:

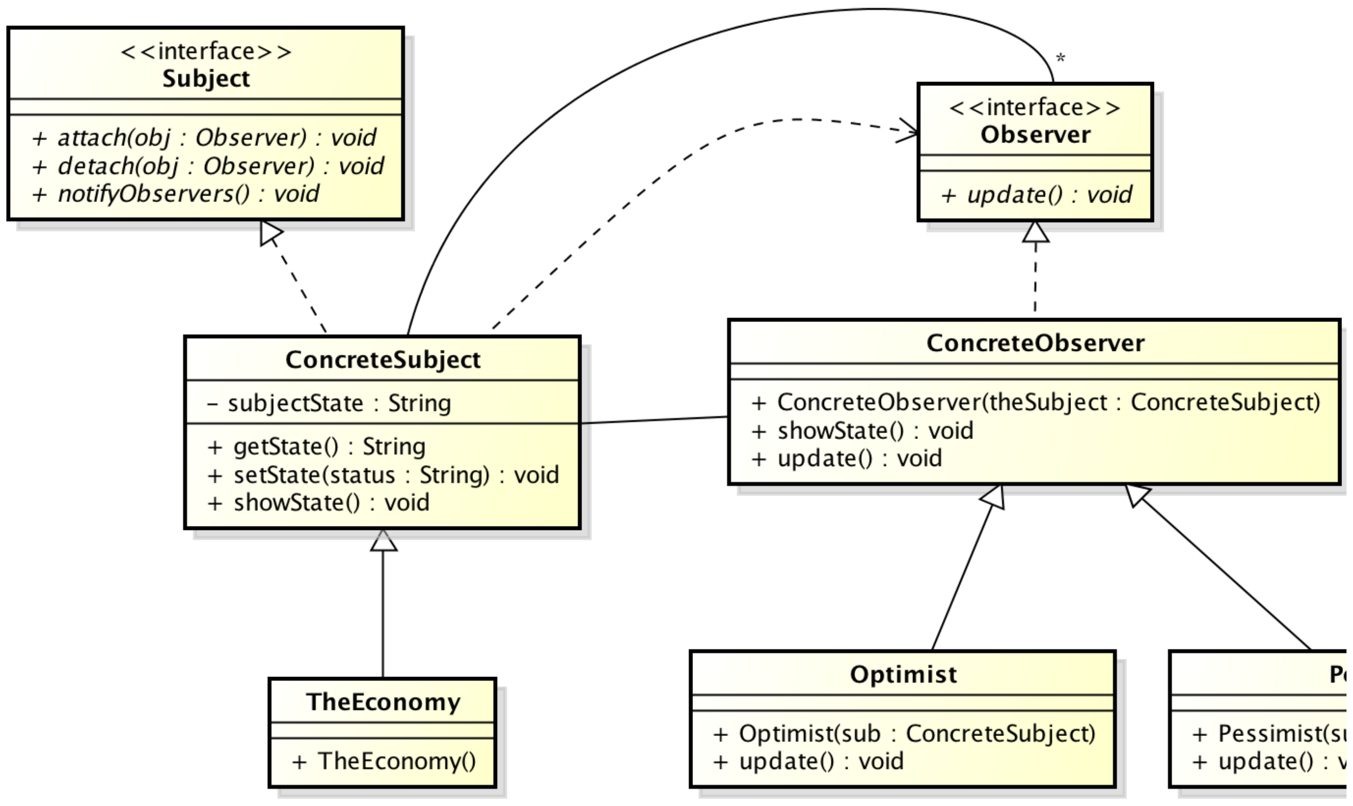
- **Java Source Code:** [uml-parser-test-3.zip](https://sjsu.instructure.com/courses/1188568/files/42435059/download?wrap=1) (https://sjsu.instructure.com/courses/1188568/files/42435059/download?wrap=1) (https://sjsu.instructure.com/courses/1188568/files/42435059/download?wrap=1)
- **Expected Class Diagram:**



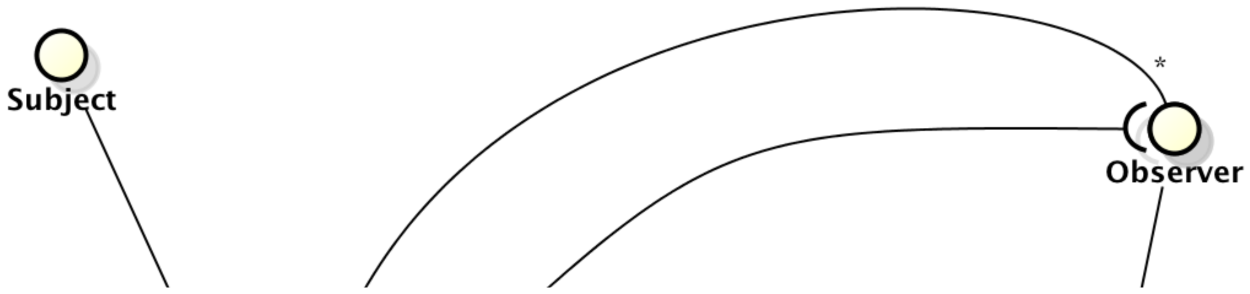


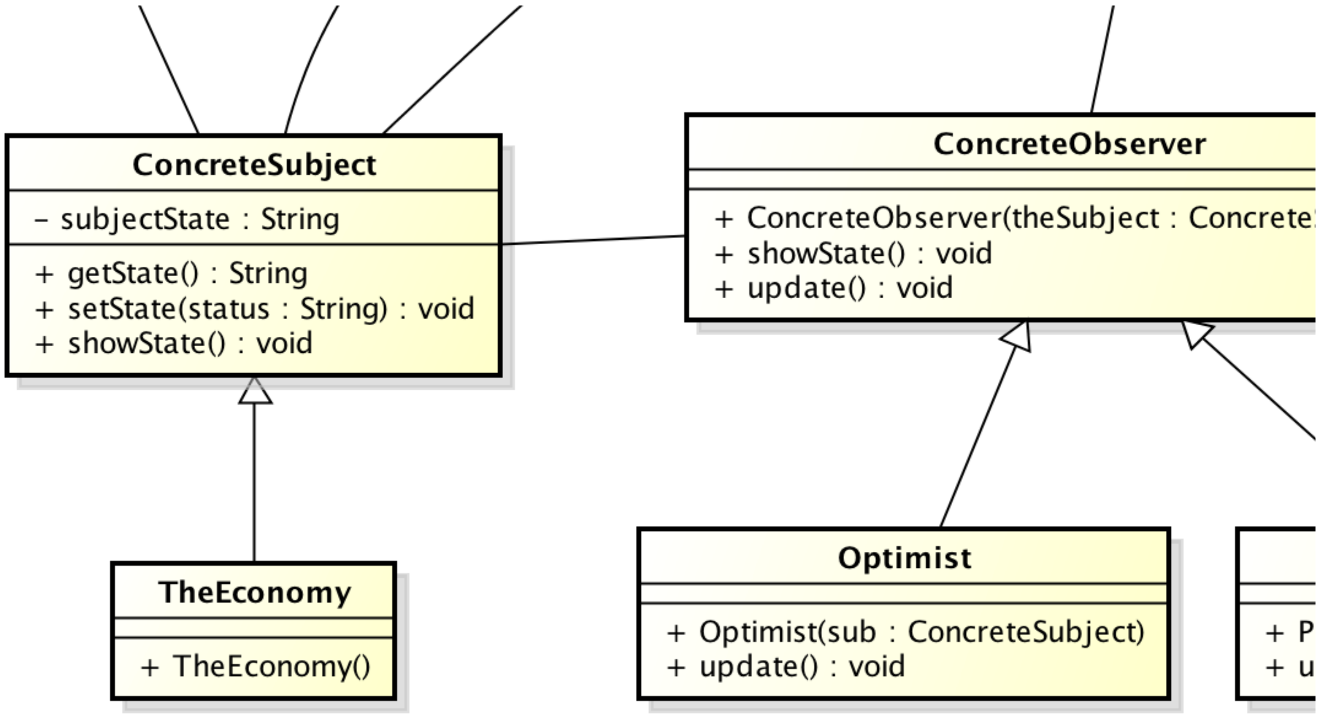
Test Case #4:

- **Java Source Code:** uml-parser-test-4.zip (https://sjsu.instructure.com/courses/1188568/files/42435064/download?wrap=1) (https://sjsu.instructure.com/courses/1188568/files/42435064/download?wrap=1)
- **Expected Class Diagram:**



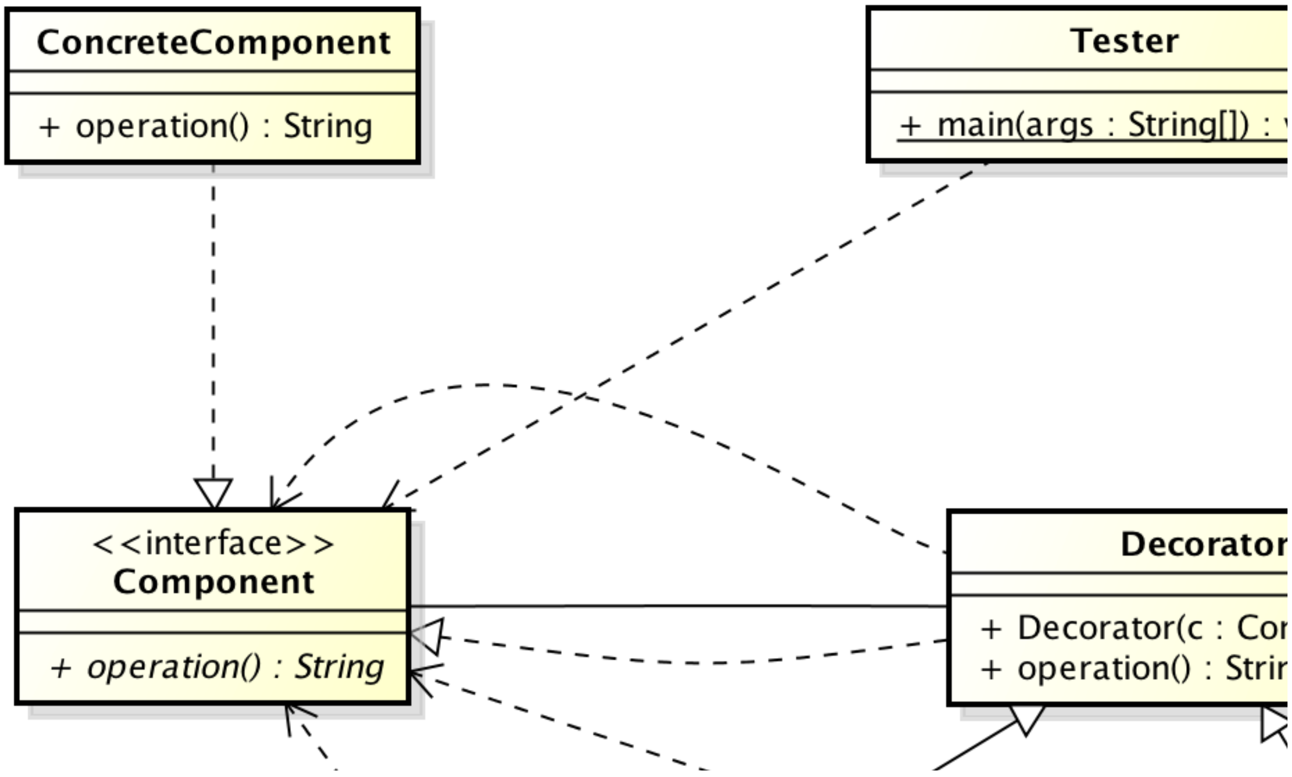
- **Expected Class Diagram (Alternative Icon Notation for Interfaces):**

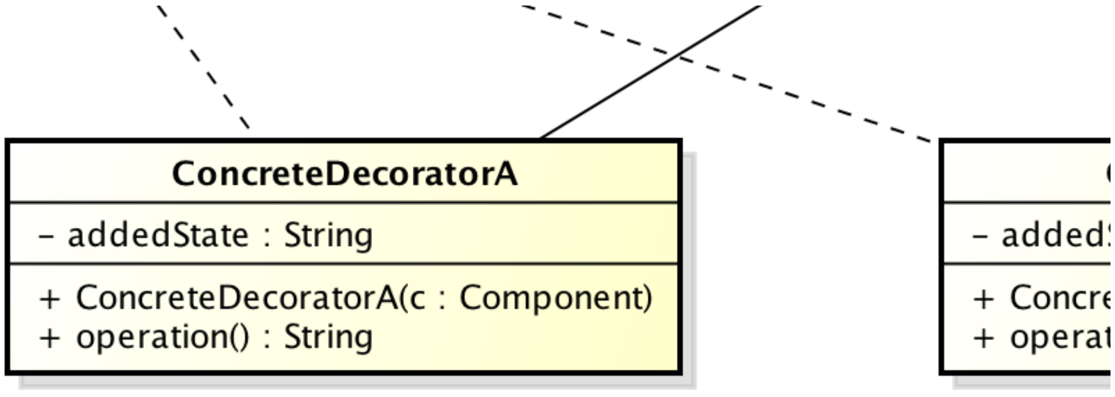




Test Case #5:

- **Java Source Code:** uml-parser-test-5.zip (https://sjsu.instructure.com/courses/1188568/files/42435062/download?wrap=1) (https://sjsu.instructure.com/courses/1188568/files/42435062/download?wrap=1)
- **Expected Class Diagram:**




ConcreteDecoratorA
– addedState : String
+ ConcreteDecoratorA(c : Component)
+ operation() : String
– added
+ Concre
+ operat