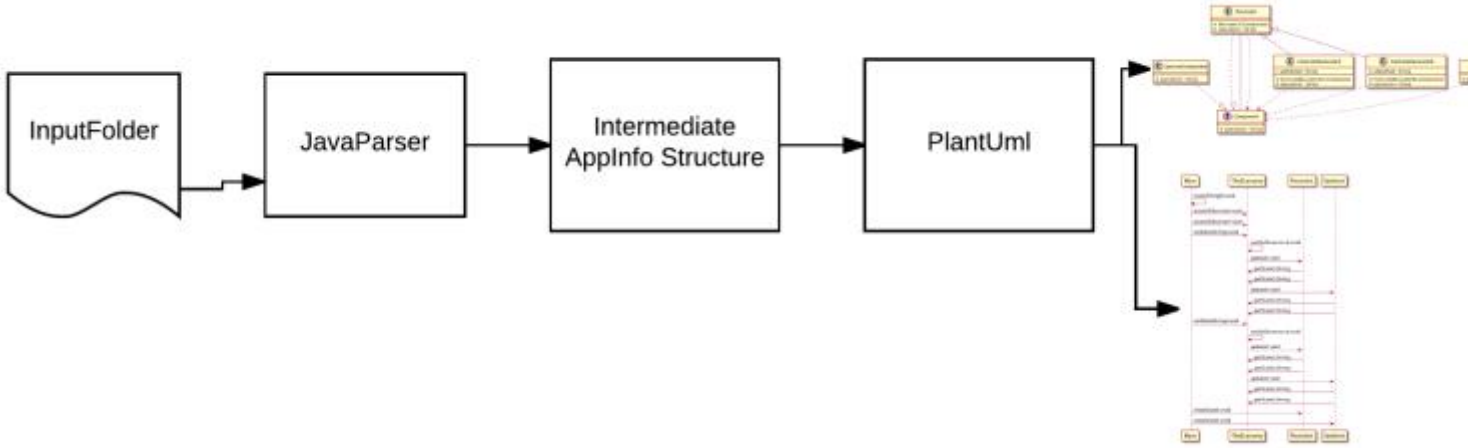
UML PARSER

Rashmi Sharma (011818729)

Github Link*: https://github.com/rashmishrm/cmpe202-UMLParser*



UML Parser is project which takes input as Folder containing classes and gives output as class diagram or sequence diagram depending on input.

Internally, it uses JavaParser to load class Abstract syntax notation, extracts information and creates intermediate class structure AppInfo.

From AppInfo, all the information and is processed and converted to plant uml text notation, which is fed to PlantUML to create diagram.

Intermediate App Info structure is created to decouple JavaParser and PlantUml implementation.

Libraries Used:

1) **Java Parser:**

Reference Link: http://javaparser.org

Java Parser is excellent static code analysis java library, which provide Abstract Syntax Notation of Java File, using which one can analyze, transform or even generate code.

In project UmlParser I have used Java Parser for analyzing code, i.e I am passing each Java File to JavaParser and fetching **CompilationUnit (AST)** and using it to traverse all the nodes which includes methods,constructors,variables and then extracting information out of them to create intermediate structure which will help me in creating plant uml's text.

Sample Code:

```
// traversing classes first
        new DirectoryIterator((level, path, file) -> path.endsWith(".java"), (level, path, file) -> {
            try {
                final CompilationUnit cu = JavaParser.parse(file);
                final ClassInfo cInfo = new ClassInfo();
                new ClassVisitor(appInfo).visit(cu, cInfo);

                appInfo.getClassInfoList().add(cInfo);


            } catch (final IOException e) {
                new RuntimeException(e);
            }
        }).explore(directory);
```

I found this library quite helpful and well documented.

2) **Plant UML:**
Reference Link: http://plantuml.com

Plant UML is excellent library to create beautiful UML diagrams using simple natural language style text.
I am using this library for creating class and sequence diagram.

Sample Code:

```
final String intermediateText = buildPlantUmlIntermediateText(appInfo);
// ConsoleLogger.printLog("Intermediate Text:" + intermediateText);
OutputStream png;
try {
   png = new FileOutputStream(fileName + ".png");
                        final SourceStringReader reader = new SourceStringReader(intermediateText);
   reader.generateImage(png);
```

3) **AspectJ**
AspectJ is excellent library which helps in weaving code at compile or at run time. It helps in removing cross cutting concerns etc.
I am using AspectJ to trace method calls while running program to generate sequence diagram.

How to use this application?

1. Go to *https://github.com/rashmishrm/cmpe202-UMLParser and Download* Executable-Jar folder
2. cd Executable-Jar/
3. sh umlparser.sh <input-folder-path> <file-name>

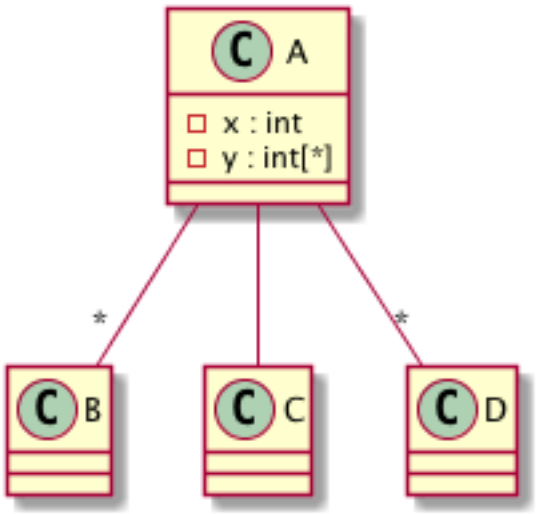
Project Demo Link:

Dishant is one of the CMPE-281 students I paired with, I am Tenant 1 in this video.

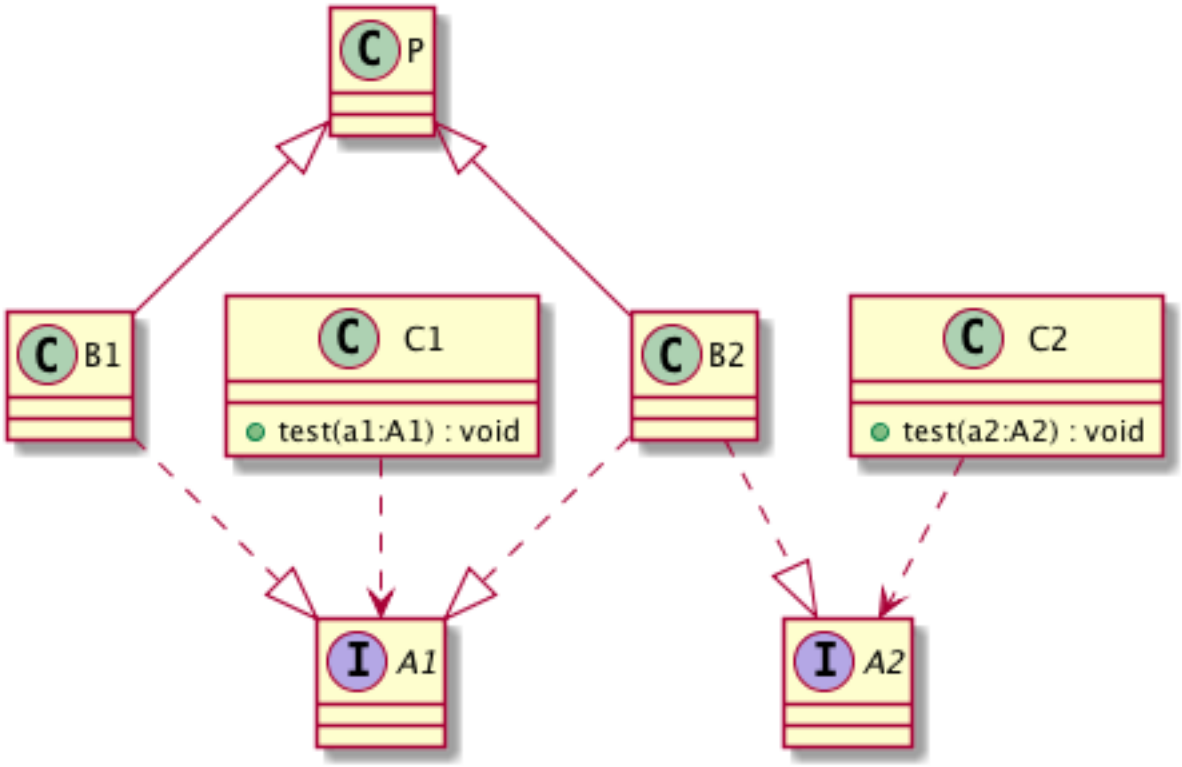
https://www.youtube.com/watch?v=OWMK0lBEMHY&feature=youtu.be

Diagram Outputs:

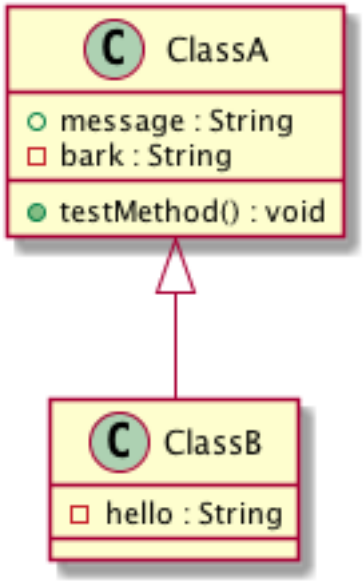
Test Case 1



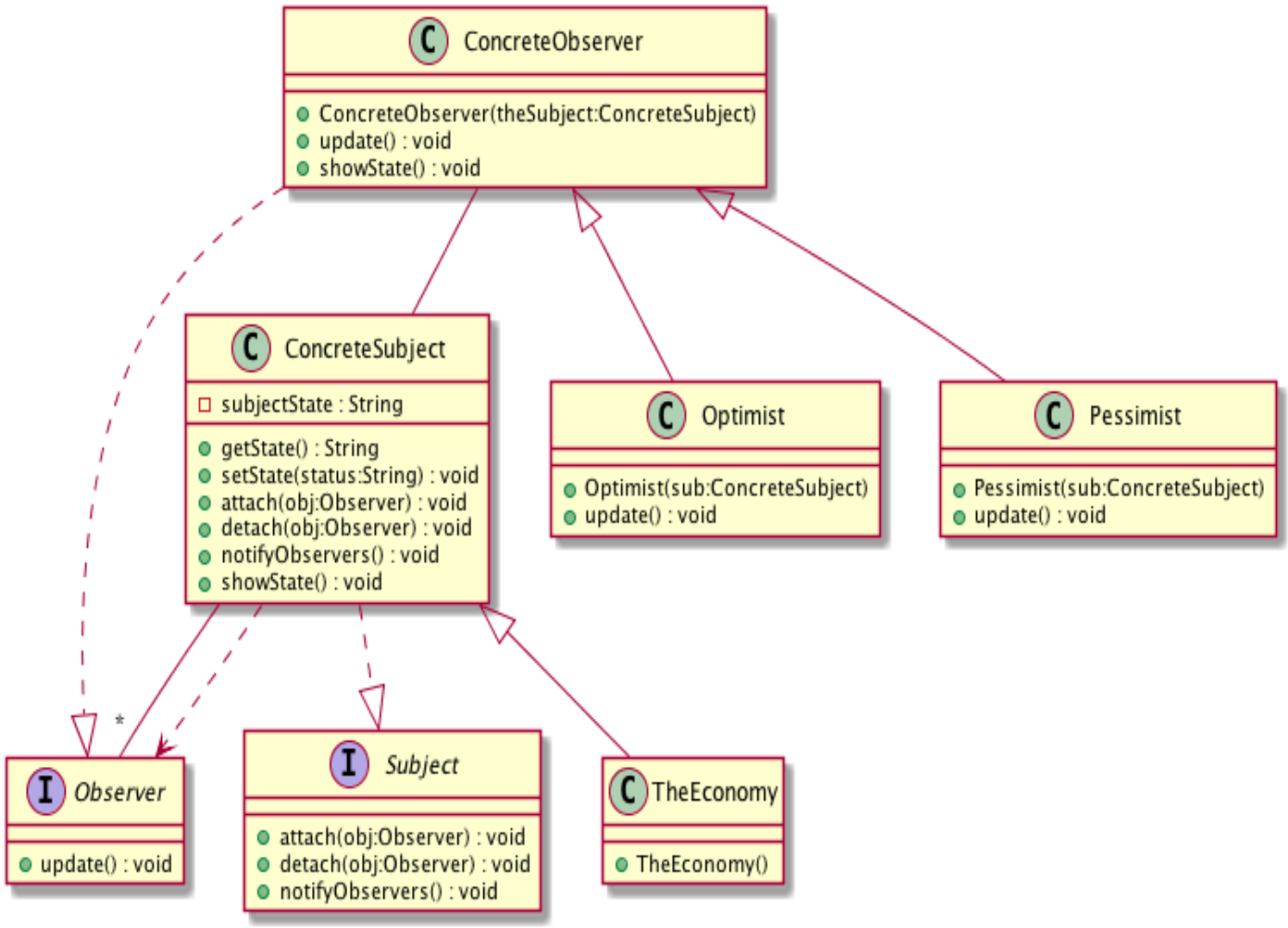
Test Case 2



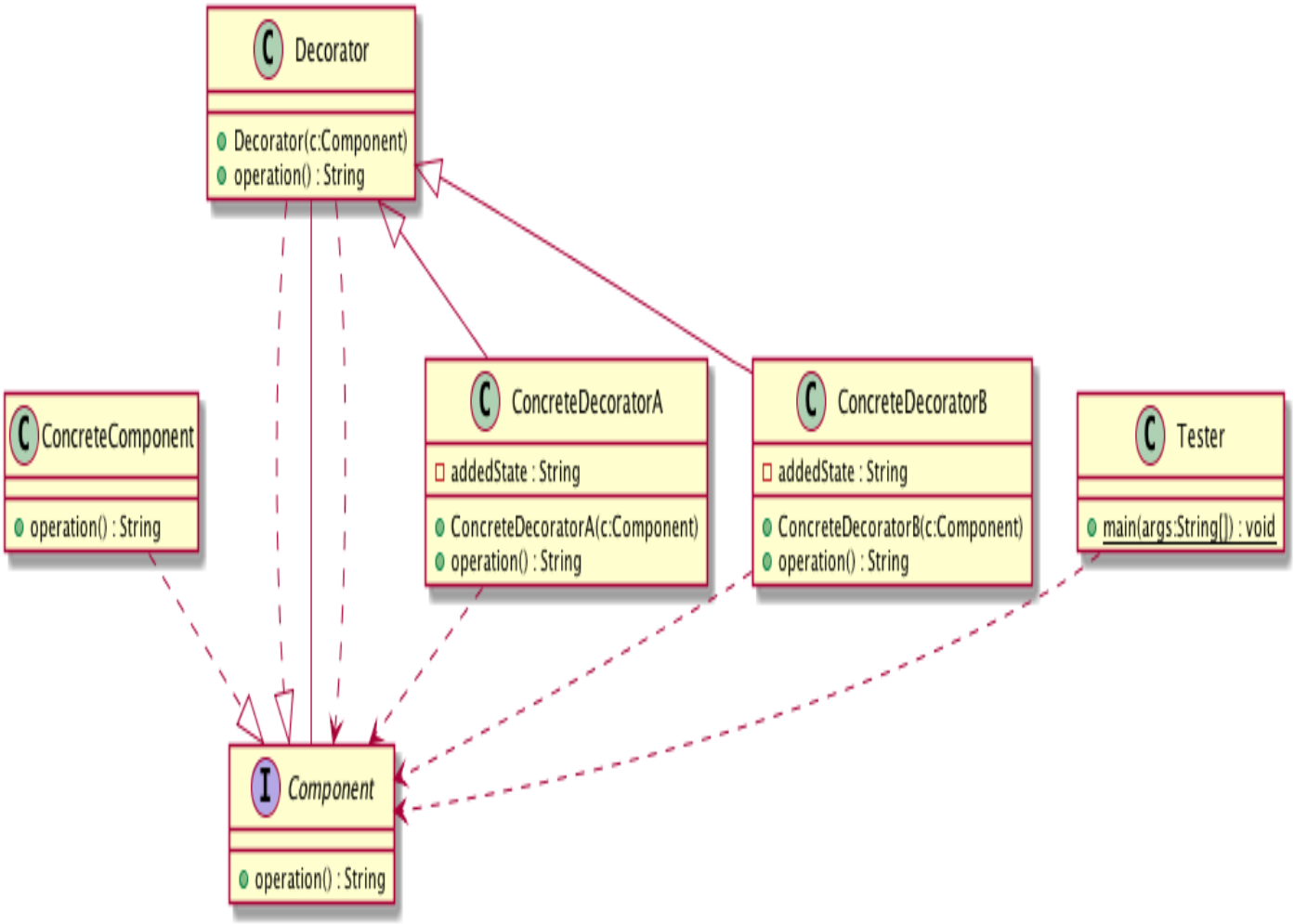
Test Case 3

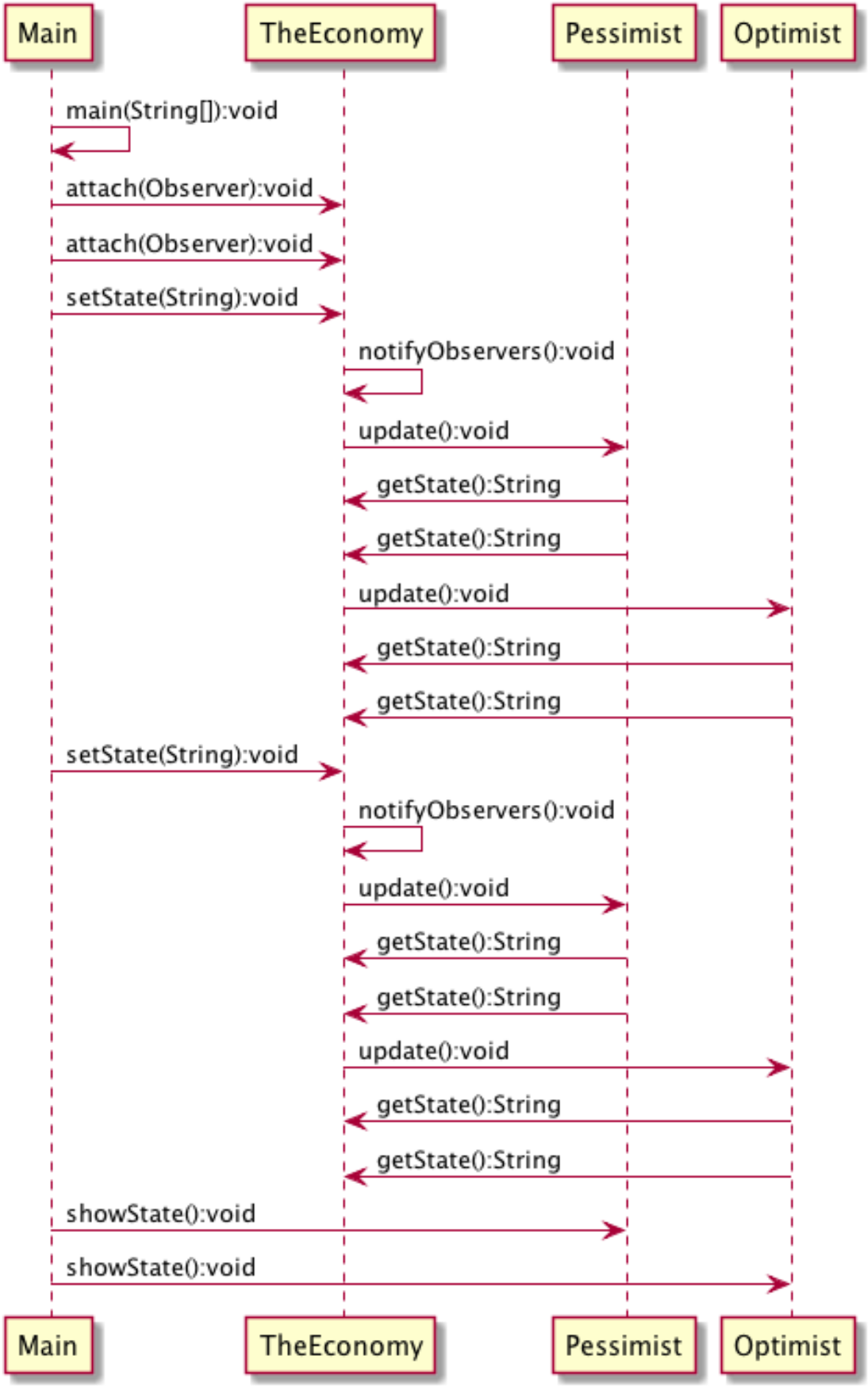

ClassA
message : String
bark : String
testMethod() : void
ClassB
hello : String

Test Case 4


ConcreteObserver
ConcreteObserver(theSubject:ConcreteSubject)
update() : void
showState() : void
ConcreteSubject
subjectState : String
getState() : String
setState(status:String) : void
attach(obj:Observer) : void
detach(obj:Observer) : void
notifyObservers() : void
showState() : void
Optimist
Optimist(sub:ConcreteSubject)
update() : void
Pessimist
Pessimist(sub:ConcreteSubject)
update() : void
*
Observer
update() : void
Subject
attach(obj:Observer) : void
detach(obj:Observer) : void
notifyObservers() : void
TheEconomy
TheEconomy()

Test Case 5


Decorator
Decorator(c:Component)
operation() : String
ConcreteComponent
operation() : String
ConcreteDecoratorA
addedState : String
ConcreteDecoratorA(c:Component)
operation() : String
ConcreteDecoratorB
addedState : String
ConcreteDecoratorB(c:Component)
operation() : String
Tester
main(args:String[]) : void
Component
operation() : String


Main
TheEconomy
Pessimist
Optimist
main(String[]):void
attach(Observer):void
attach(Observer):void
setState(String):void
notifyObservers():void
update():void
getState():String
getState():String
update():void
getState():String
getState():String
setState(String):void
notifyObservers():void
update():void
getState():String
getState():String
update():void
getState():String
getState():String
showState():void
showState():void
Main
TheEconomy
Pessimist
Optimist