Assignment 4-5: Patterns

Submitted By:

- Sannidhya Pathania (spathan3)
- Karan Malik (kmalik8)
- Aashka Dave (adave8)
- Sarthak Vats (svats2)
- Darshan Prakashbhai Panchal (dpancha6)
- Thy Do (thydo)

Date: Dec 2, 2022

Table of Contents

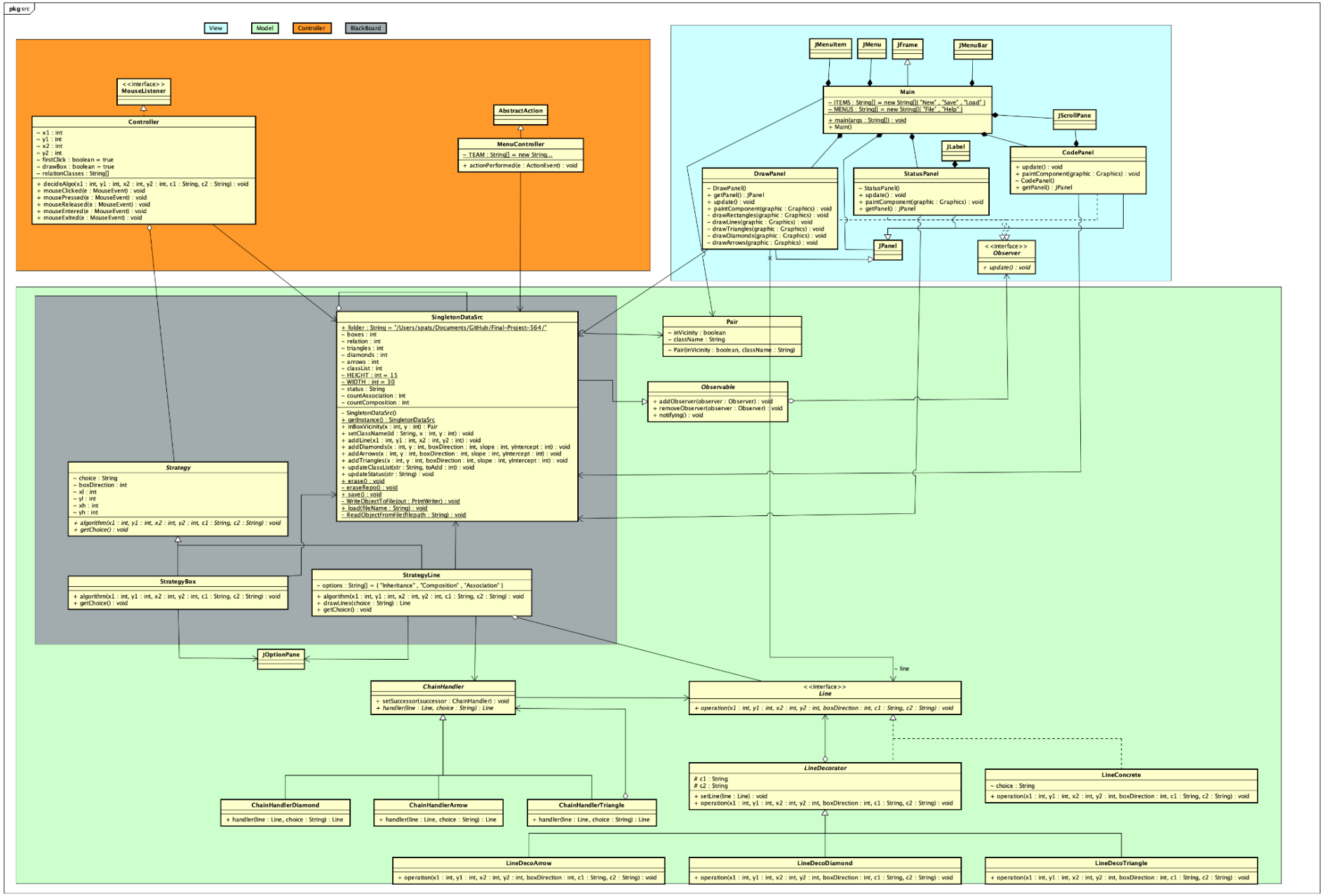
Java Files:

GitHub Link: https://github.com/spats18/Final-Project-564

Java Files: (Uploaded on Canvas alongside PDF)

Class diagram:

Uploaded separately as well on canvas.


pkg src
View
Model
Controller
BlackBoard
<<interface>>
MouseListener
Controller
~ x1 : int
~ y1 : int
~ x2 : int
~ y2 : int
~ firstClick : boolean = true
~ drawBox : boolean = true
~ relationClasses : String[]
+ decideAlgo(x1 : int, y1 : int, x2 : int, y2 : int, c1 : String, c2 : String) : void
+ mouseClicked(e : MouseEvent) : void
+ mousePressed(e : MouseEvent) : void
+ mouseReleased(e : MouseEvent) : void
+ mouseEntered(e : MouseEvent) : void
+ mouseExited(e : MouseEvent) : void
AbstractAction
MenuController
~ TEAM : String[] = new String...
+ actionPerformed(e : ActionEvent) : void
JMenuItem
JMenu
JFrame
JMenuBar
Main
~ ITEMS : String[] = new String[]{ "New" , "Save" , "Load" }
~ MENUS : String[] = new String[]{ "File" , "Help" }
+ main(args : String[]) : void
+ Main()
JScrollPane
JLabel
CodePanel
+ update() : void
+ paintComponent(graphic : Graphics) : void
~ CodePanel()
+ getPanel() : JPanel
DrawPanel
~ DrawPanel()
+ getPanel() : JPanel
+ update() : void
+ paintComponent(graphic : Graphics) : void
- drawRectangles(graphic : Graphics) : void
- drawLines(graphic : Graphics) : void
- drawTriangles(graphic : Graphics) : void
- drawDiamonds(graphic : Graphics) : void
- drawArrows(graphic : Graphics) : void
StatusPanel
~ StatusPanel()
+ update() : void
+ paintComponent(graphic : Graphics) : void
+ getPanel() : JPanel
JPanel
<<interface>>
Observer
+ update() : void
SingletonDataSrc
+ folder : String = "/Users/spats/Documents/GitHub/Final-Project-564/"
~ boxes : int
~ relation : int
~ triangles : int
~ diamonds : int
~ arrows : int
~ classList : int
~ HEIGHT : int = 15
~ WIDTH : int = 30
~ status : String
~ countAssociation : int
~ countComposition : int
- SingletonDataSrc()
+ getInstance() : SingletonDataSrc
+ inBoxVicinity(x : int, y : int) : Pair
+ setClassName(id : String, x : int, y : int) : void
+ addLine(x1 : int, y1 : int, x2 : int, y2 : int) : void
+ addDiamonds(x : int, y : int, boxDirection : int, slope : int, yIntercept : int) : void
+ addArrows(x : int, y : int, boxDirection : int, slope : int, yIntercept : int) : void
+ addTriangles(x : int, y : int, boxDirection : int, slope : int, yIntercept : int) : void
+ updateClassList(str : String, toAdd : int) : void
+ updateStatus(str : String) : void
+ erase() : void
~ eraseRepo() : void
+ save() : void
~ WriteObjectToFile(out : PrintWriter) : void
+ load(fileName : String) : void
~ Read(ObjectFromFile(filepath : String) : void
Pair
~ inVicinity : boolean
~ className : String
~ Pair(inVicinity : boolean, className : String)
Observable
+ addObserver(observer : Observer) : void
+ removeObserver(observer : Observer) : void
+ notifying() : void
Strategy
~ choice : String
~ boxDirection : int
~ xl : int
~ yl : int
~ xh : int
~ yh : int
+ algorithm(x1 : int, y1 : int, x2 : int, y2 : int, c1 : String, c2 : String) : void
+ getChoice() : void
StrategyBox
+ algorithm(x1 : int, y1 : int, x2 : int, y2 : int, c1 : String, c2 : String) : void
+ getChoice() : void
StrategyLine
~ options : String[] = { "Inheritance" , "Composition" , "Association" }
+ algorithm(x1 : int, y1 : int, x2 : int, y2 : int, c1 : String, c2 : String) : void
+ drawLines(choice : String) : Line
+ getChoice() : void
JOptionPane
- line
ChainHandler
+ setSuccessor(successor : ChainHandler) : void
+ handler(line : Line, choice : String) : Line
<<interface>>
Line
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void
LineDecorator
c1 : String
c2 : String
+ setLine(line : Line) : void
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void
LineConcrete
~ choice : String
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void
ChainHandlerDiamond
+ handler(line : Line, choice : String) : Line
ChainHandlerArrow
+ handler(line : Line, choice : String) : Line
ChainHandlerTriangle
+ handler(line : Line, choice : String) : Line
LineDecoArrow
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void
LineDecoDiamond
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void
LineDecoTriangle
+ operation(x1 : int, y1 : int, x2 : int, y2 : int, boxDirection : int, c1 : String, c2 : String) : void

Architectural Patterns:

1. Model View Controller

The model view controller pattern was used to separate the application into the 3 logical components:

Model:

The main logic of the code is mentioned in the model. It contains the SingletonDataSrc and the strategies(Knowledge Source) for deciding what to add to the resource (SingletonDataSrc).

View:

The view corresponds to the logic of the GUI of the application. The DrawPanel that the user will interact with is part of the view along with the menu, CodePanel, and status bar.

Controller:

The controller acts as an interface between the model and the view. Actions performed by the user in the view such as mouse clicks trigger the events in the controller. There are two classes within the controller - the MenuController and the Controller. The MenuController deals with the action events triggered by the menu bar. The Controller class is triggered by the mouse events and calls the Strategy pattern to handle the operation - making boxes or lines

2. Blackboard

The blackboard pattern was used to create a SingletonDataSrc that acts as a shared space for storing the information.

Design Patterns:

1. Observer

- **SingletonDataSrc (Observable)**
- **DrawPanel (Observers)**
- **CodePanel (Observers)**

- **StatusPanel (Observers)**

The SingletonDataSrc is the subject (Observable) with the DrawPanel, CodePanel, and StatusPanel as Observers. Whenever the user clicks on the DrawPanel, the SingletonDataSrc will update its common resource (via strategy pattern) creating a class or adding a line. Every time it sets the classname or updates it, or adds a relation between two classes, it notifies the observers. The DrawPanel calls its paintcomponent method to handle the task, the CodePanel writes the data on the screen, and the StatusPanel logs the event.

2. Singleton

- **SingletonDataSrc**

The singleton pattern was used to implement the class SingletonDataSrc. The class holds all the information like the coordinates of the boxes, lines, decorations, the names of the classes, and the relationship between the two classes. It also has the logic for computing whether the mouse click occurred in the vicinity of the box or not to determine whether a box is to be created or not.

3. Strategy

- **Strategy**
- **StrategyLine**
- **StrategyBox**

The strategy pattern was used to add two different implementations for when a user clicks in the DrawPanel. Upon a mouse click, the controller decides whether to call StrategyLine or StrategyBox. StrategyBox computes the coordinates of the box and then adds the box to SingletonDataSrc which notifies DrawPanel to draw the box. StrategyLine computes the coordinates for the line and asks the user to select between the 3 types of relations and initializes the chain of responsibility. The chain of responsibility will decide what to decorate and will call respective decorations and then will add the data to SingletonDataSrc, which will notify the DrawPanel, StatusPanel, and CodePanel to repaint themselves.

4. Chain of Responsibility

- **ChainHandler**
- **ChainHandlerArrow**
- **ChainHandlerDiamond**
- **ChainHandlerTriangle**

The chain of responsibility pattern was implemented where we have 3 handlers, one for each relation. The handlers are linked to form a chain. From the StrategyLine, the choice of the user is passed on to each handler, where the respective handler decides what decorations we need to add over the line.

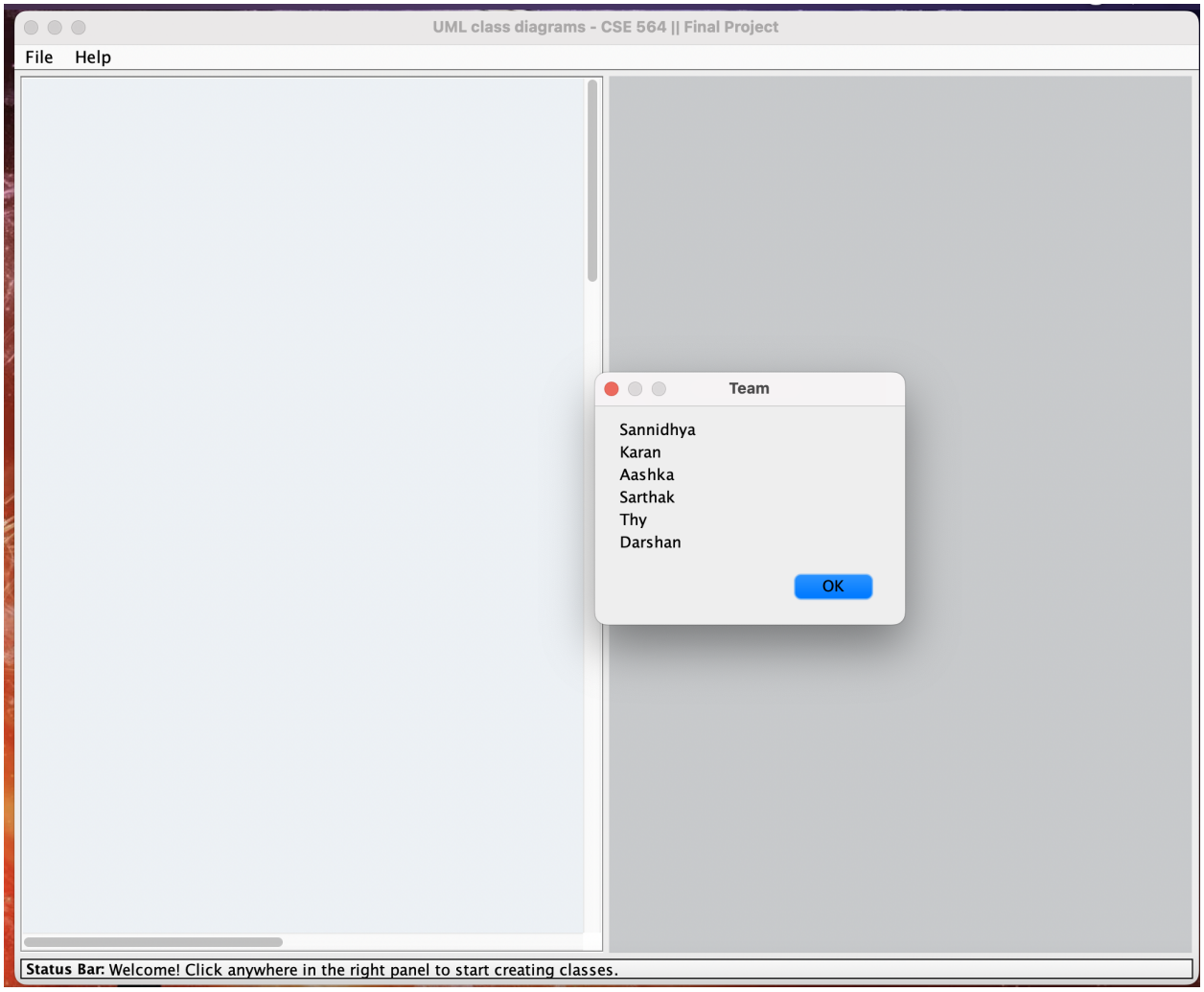
5. Decorator

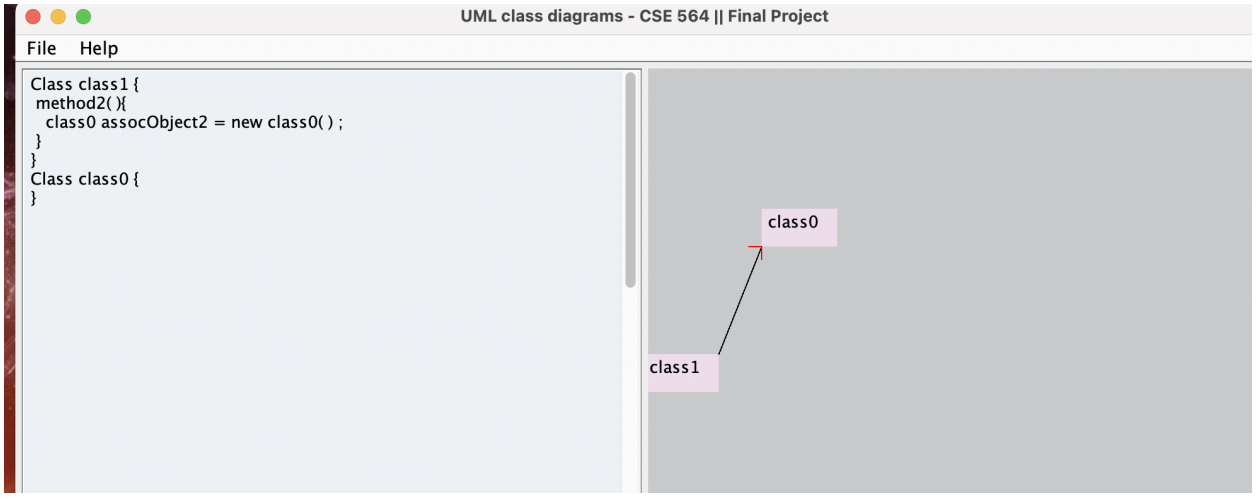
- **Line**
- **LineConcrete**
- **LineDecorator**
- **LineDecoArrow**
- **LineDecoDiamond**
- **LineDecoTriangle**

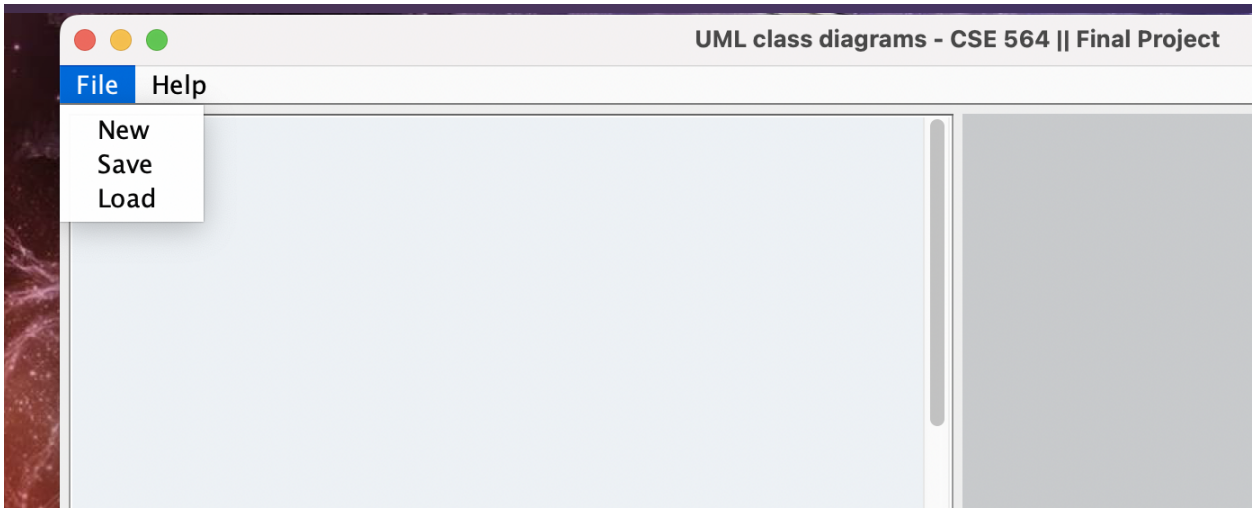
The decorator pattern was used to add the relation decorations to the base line. The decision from the chain handler was passed to the respective decorator class.

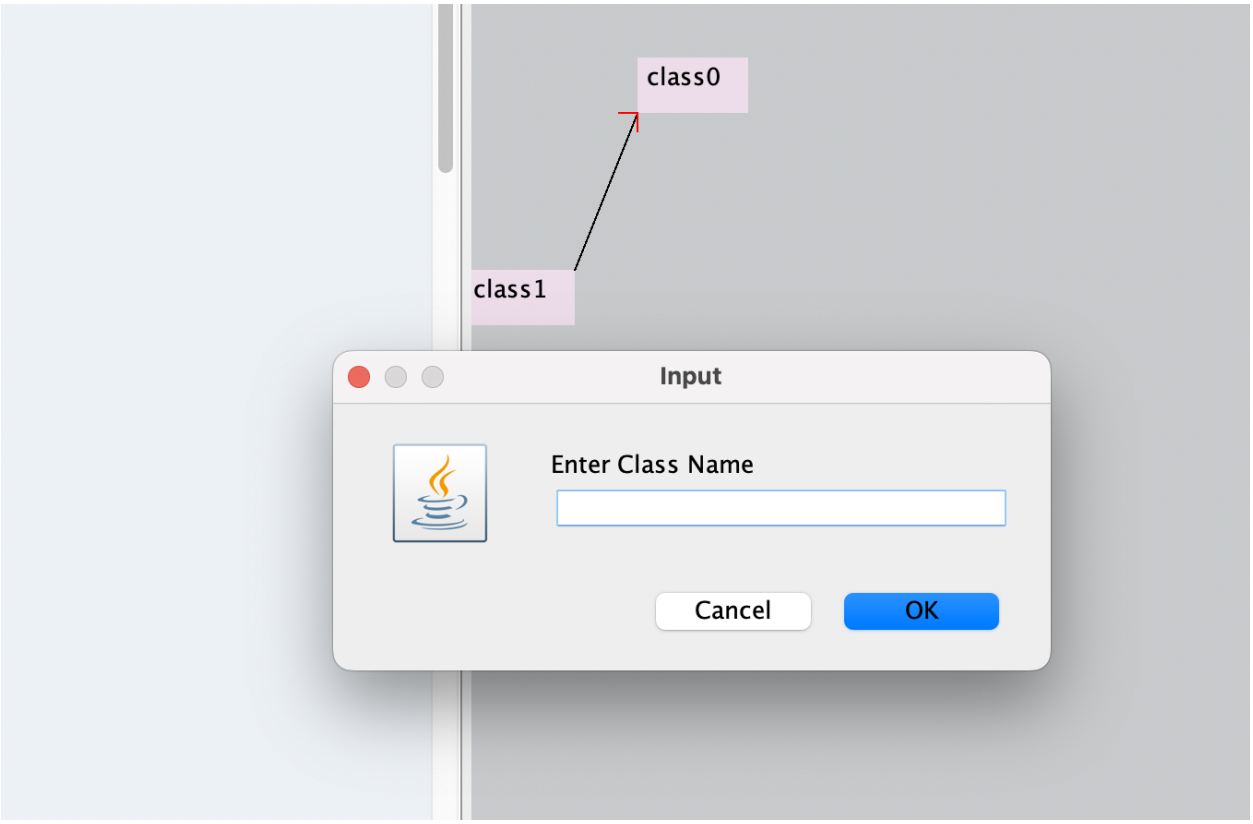
Output:

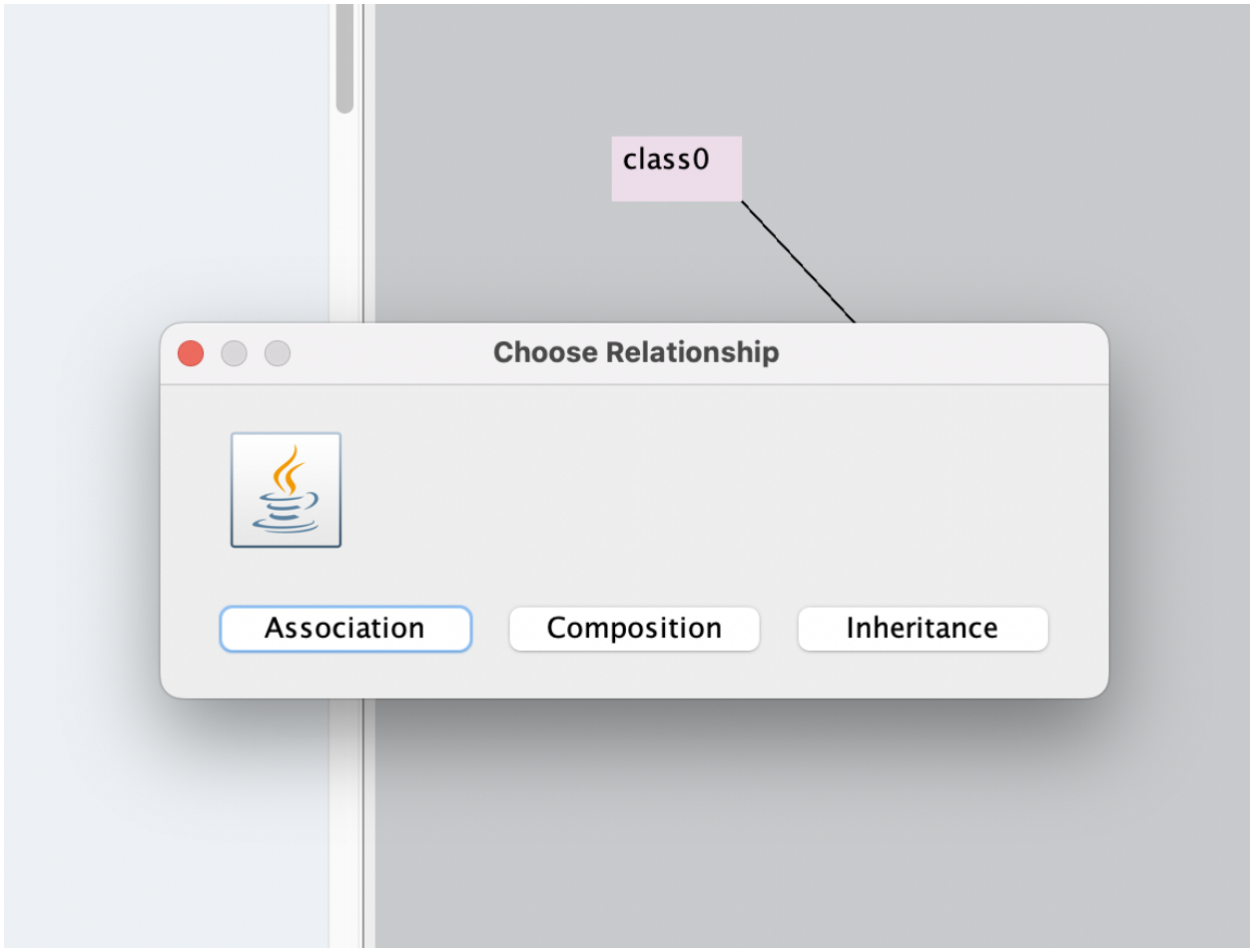

UML class diagrams - CSE 564 || Final Project
File
Help
Status Bar: Welcome! Click anywhere in the right panel to start creating classes.


UML class diagrams - CSE 564 || Final Project
File
Help
Team
Sannidhya
Karan
Aashka
Sarthak
Thy
Darshan
OK
Status Bar: Welcome! Click anywhere in the right panel to start creating classes.

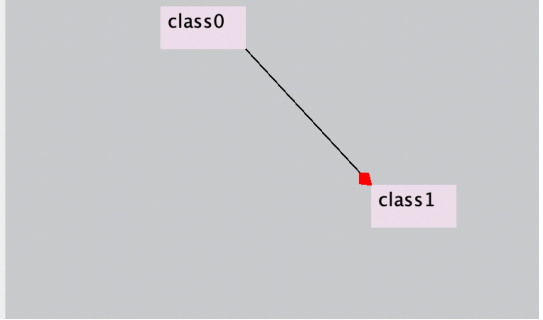

UML class diagrams - CSE 564 || Final Project
File
Help
Class class1 {
method2( ){
class0 assocObject2 = new class0( ) ;
}
}
Class class0 {
}
class0
class1

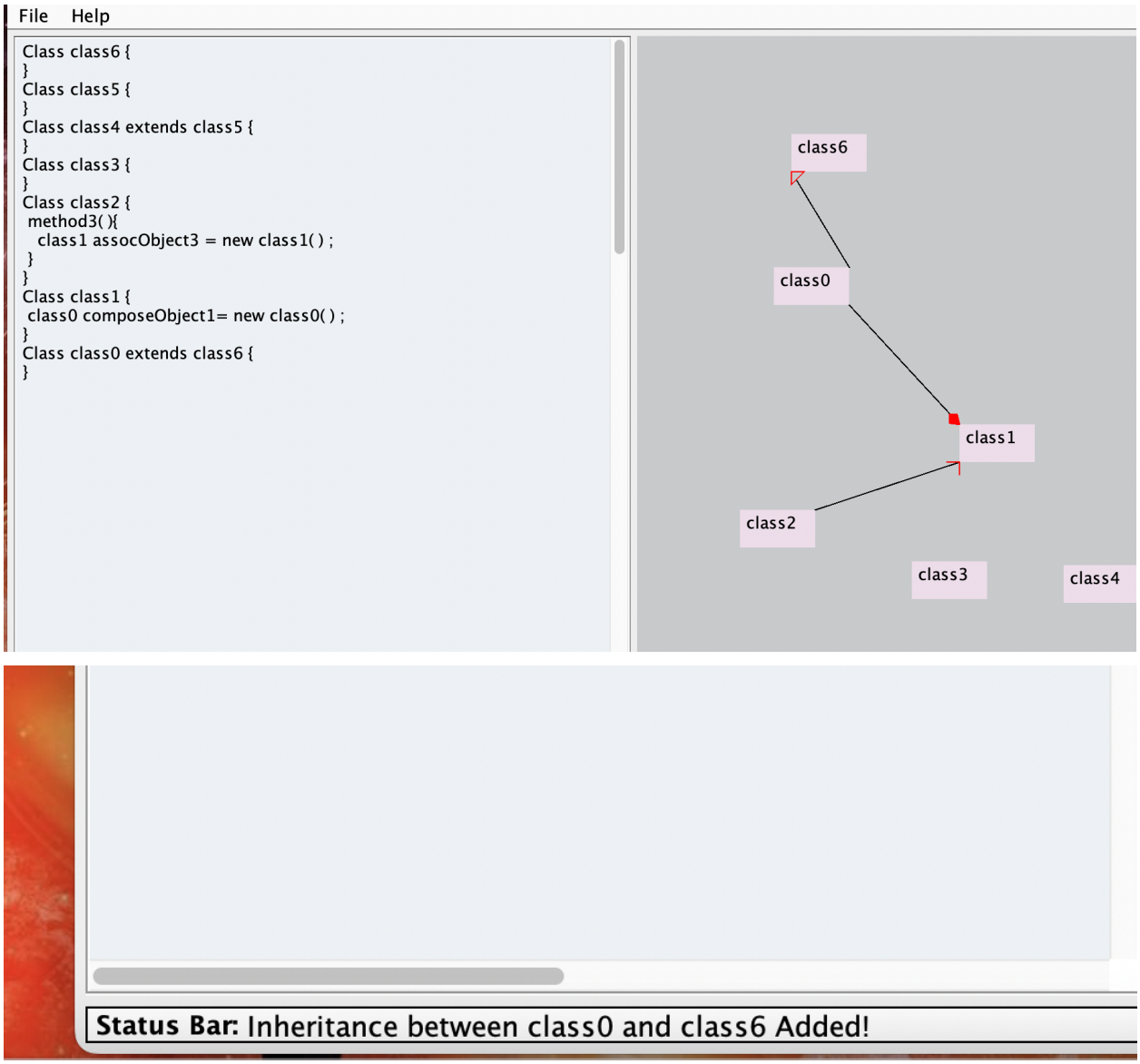

UML class diagrams - CSE 564 || Final Project
File
Help
New
Save
Load

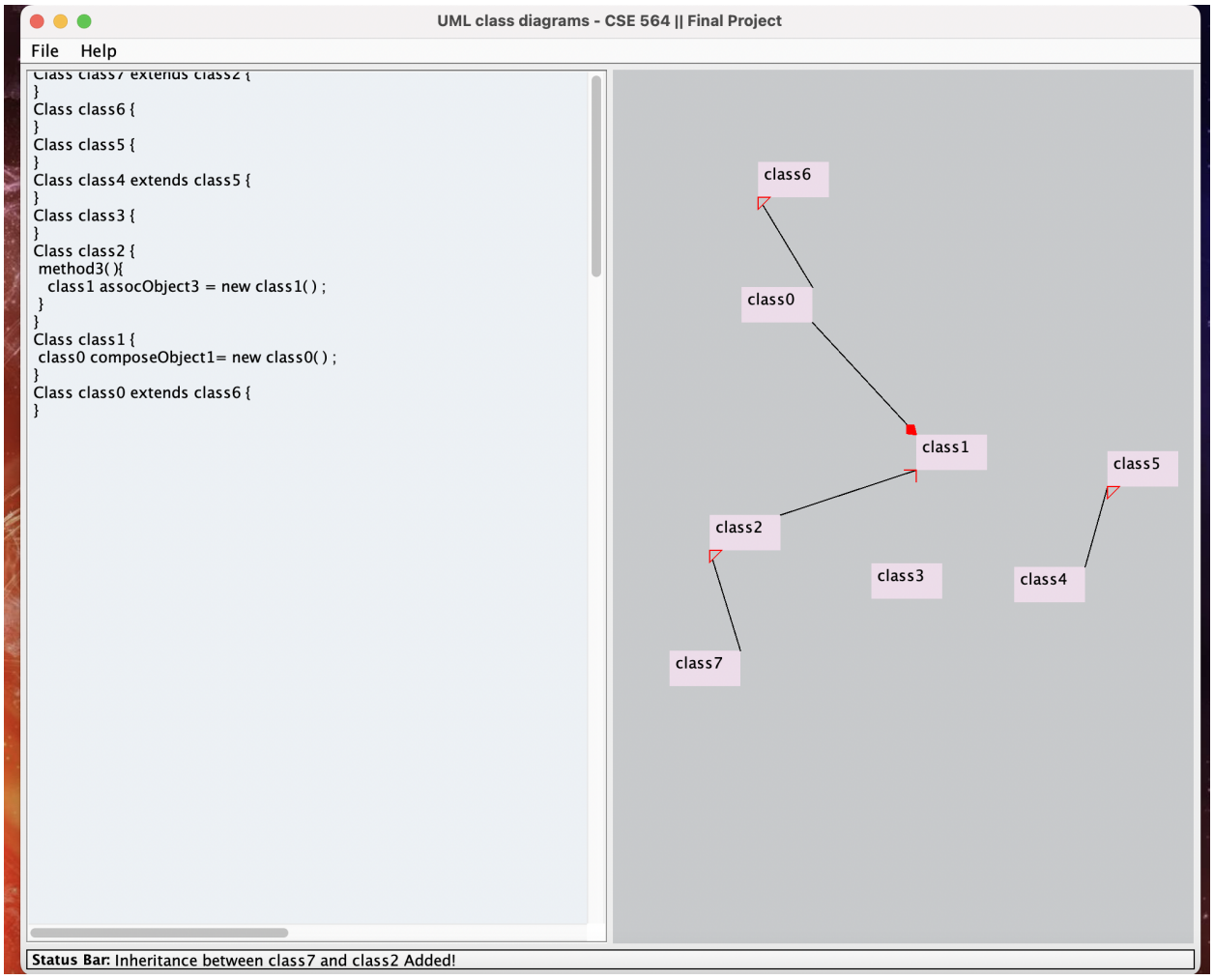

class0
class1
Input
Enter Class Name
Cancel
OK

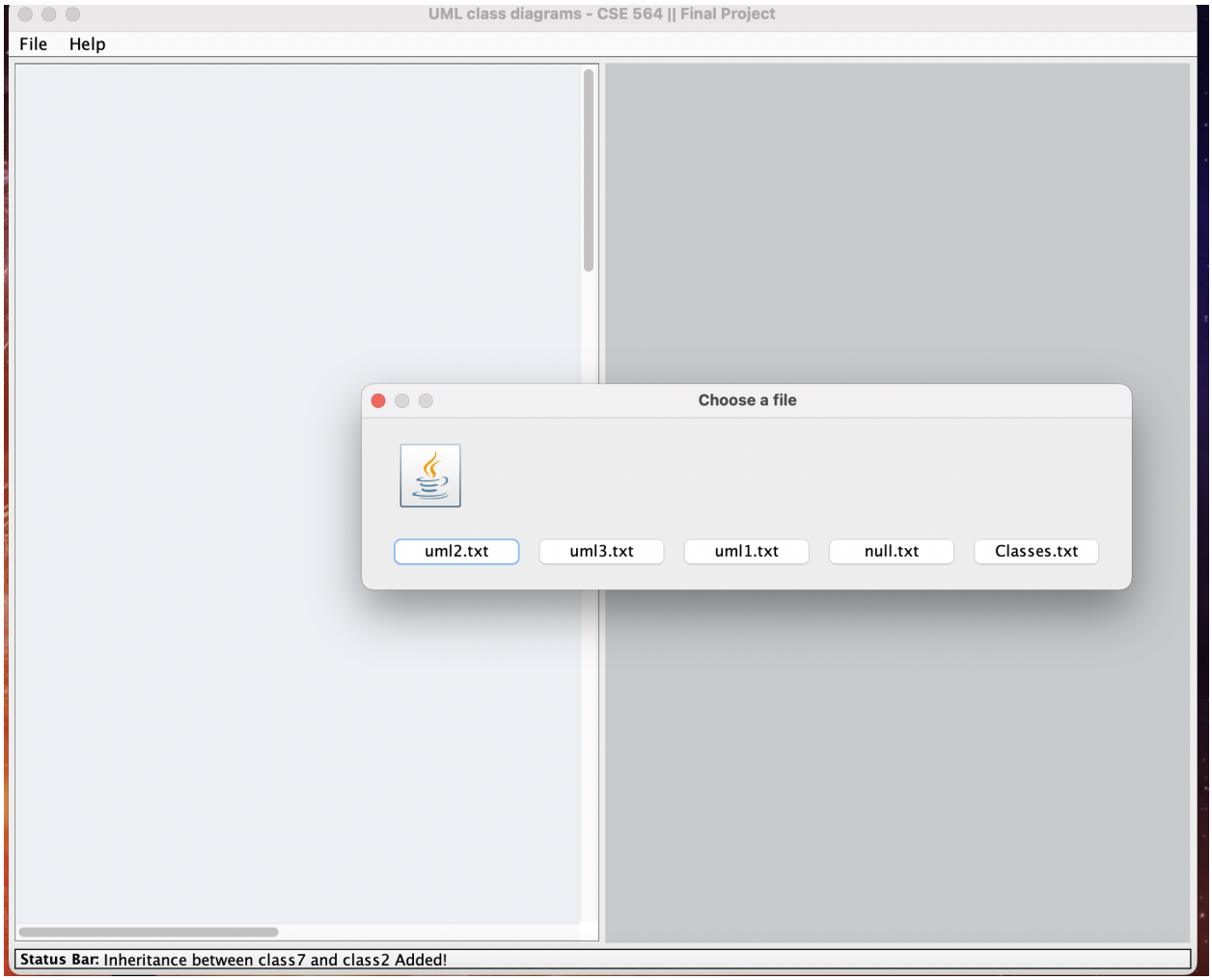


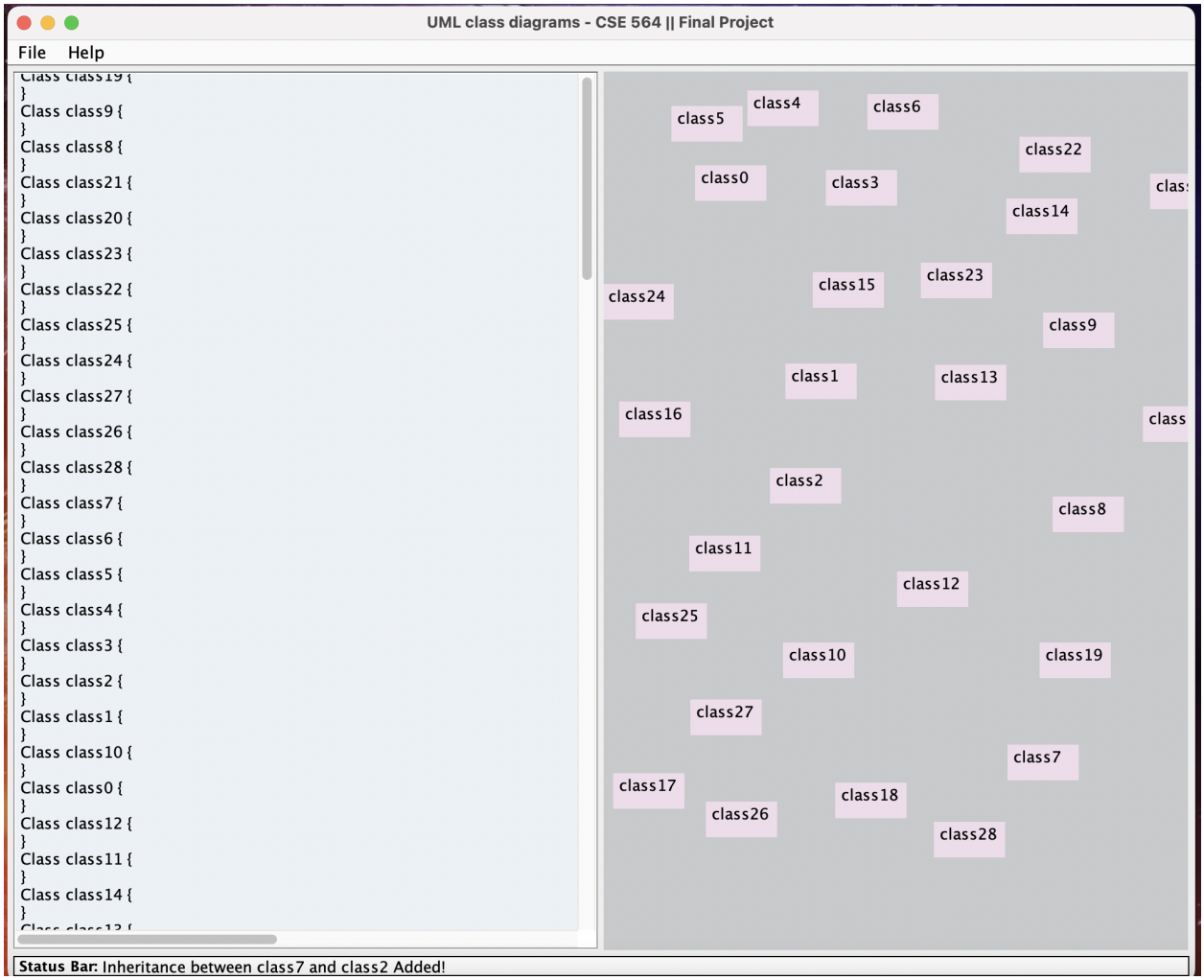

```
Class class1 {
 class0 composeObject1= new class0( ) ;
}
Class class0 {
}
```




File
Help
Class class6 {
}
Class class5 {
}
Class class4 extends class5 {
}
Class class3 {
}
Class class2 {
method3( ){
class1 assocObject3 = new class1( ) ;
}
}
Class class1 {
class0 composeObject1= new class0( ) ;
}
Class class0 extends class6 {
}
class6
class0
class1
class2
class3
class4
Status Bar: Inheritance between class0 and class6 Added!


UML class diagrams - CSE 564 || Final Project
File
Help
}
Class class6 {
}
Class class5 {
}
Class class4 extends class5 {
}
Class class3 {
}
Class class2 {
method3( ){
class1 assocObject3 = new class1( ) ;
}
}
Class class1 {
class0 composeObject1= new class0( ) ;
}
Class class0 extends class6 {
}
class6
class0
class1
class5
class2
class3
class4
class7
Status Bar: Inheritance between class7 and class2 Added!


UML class diagrams - CSE 564 || Final Project
File
Help
Choose a file
uml2.txt
uml3.txt
uml1.txt
null.txt
Classes.txt
Status Bar: Inheritance between class7 and class2 Added!


UML class diagrams - CSE 564 || Final Project
File
Help
Class class19 {
}
Class class9 {
}
Class class8 {
}
Class class21 {
}
Class class20 {
}
Class class23 {
}
Class class22 {
}
Class class25 {
}
Class class24 {
}
Class class27 {
}
Class class26 {
}
Class class28 {
}
Class class7 {
}
Class class6 {
}
Class class5 {
}
Class class4 {
}
Class class3 {
}
Class class2 {
}
Class class1 {
}
Class class10 {
}
Class class0 {
}
Class class12 {
}
Class class11 {
}
Class class14 {
}
class4
class6
class5
class22
class0
class3
class14
class15
class23
class24
class9
class1
class13
class16
class2
class8
class11
class12
class25
class10
class19
class27
class7
class17
class18
class26
class28
Status Bar: Inheritance between class7 and class2 Added!


UML class diagrams - CSE 564 || Final Project
File
Help
Status Bar: Inheritance between class7 and class2 Added!