|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 3 |
| Week | 03 |
| Due | 7/13/2025 |
| Student name | Toe Toe Aung |
| Student ID | 618090 |

**a) Class Diagram to add different views of the**

**counter value**A close-up of a diagram

AI-generated content may be incorrect.

**b) Sequence Diagram that shows the following scenario:**

1. The user clicks the increment button

2. The user clicks the decrement button

A diagram of a diagram

AI-generated content may be incorrect.

**c) Implementation of the observer pattern in the given code**

**Modification in Counter class**

**public** **class** Counter {

**private** **int** count=0;

List<Observer> observerList;

**public** Counter() {

observerList = **new** ArrayList<Observer>();

}

**public** **void** addObserver(Observer o) {

**this**.observerList.add(o);

}

**public** **void** increment(){

count++;

}

**public** **void** decrement(){

count--;

}

}

**Modification in JframeCounter Class**

**private** Counter counter;

**public** JFrameCounter() {

**try** {

jbInit();

counter = **new** Counter();

TextFrame textframe = **new** TextFrame();

textframe.setVisible(**true**);

RectFrame rectframe = **new** RectFrame();

rectframe.setVisible(**true**);

OvalFrame ovalframe = **new** OvalFrame();

ovalframe.setVisible(**true**);

counter.addObserver(textframe);

counter.addObserver(rectframe);

counter.addObserver(ovalframe);

} **catch** (Exception e) {

e.printStackTrace();

}

}

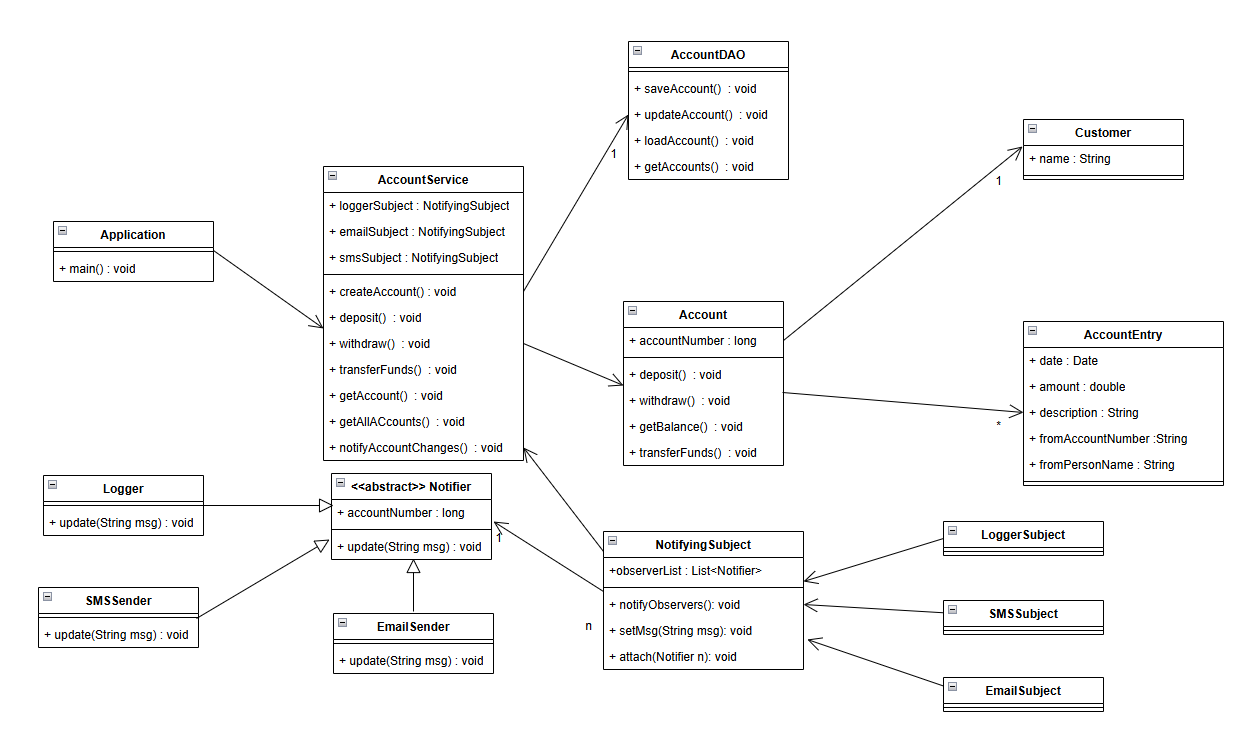
**Added Observer Interface**

**public** **interface** Observer {

**public** **abstract** **void** update(**int** counter);

}

**d and e) Modified class diagram with the observer pattern applied**

****

**f) The sequence diagram shows the following scenario:**

1. First create a new account

2. Then deposit $80 on this new account

A diagram of a project

AI-generated content may be incorrect.

**g) Implementation of the observer pattern in the given code**

**abstract** **class** Notifier {

NotifyingSubject subject;

**abstract** **void** update();

}

**public** **class** NotifyingSubject {

**private** List<Notifier> observerList;

String msg;

**public** **void** notifyObservers() {

**this**.observerList.stream().forEach(e -> e.update());

}

**public** **void** setMsg(String msg) {

**this**.msg = msg;

}

**public** **void** attach(Notifier n) {

**if** (**null** == observerList)

observerList = **new** ArrayList<Notifier>();

**this**.observerList.add(n);

}

}

**public** **class** Logger **extends** Notifier{

**public** Logger(NotifyingSubject ns) {

**this**.subject = ns;

**this**.subject.attach(**this**);

}

@Override

**public** **void** update() {

System.***out***.println("Logger: \t\t" + subject.msg);

}

}

**public** **class** LoggerSubject **extends** NotifyingSubject {

**public** LoggerSubject() {

**this**.msg = "Logger: " + msg;

}

}