COMP/EECE 4882

Final Exam

Spring 2012

Name: Answer Key

Instructions:

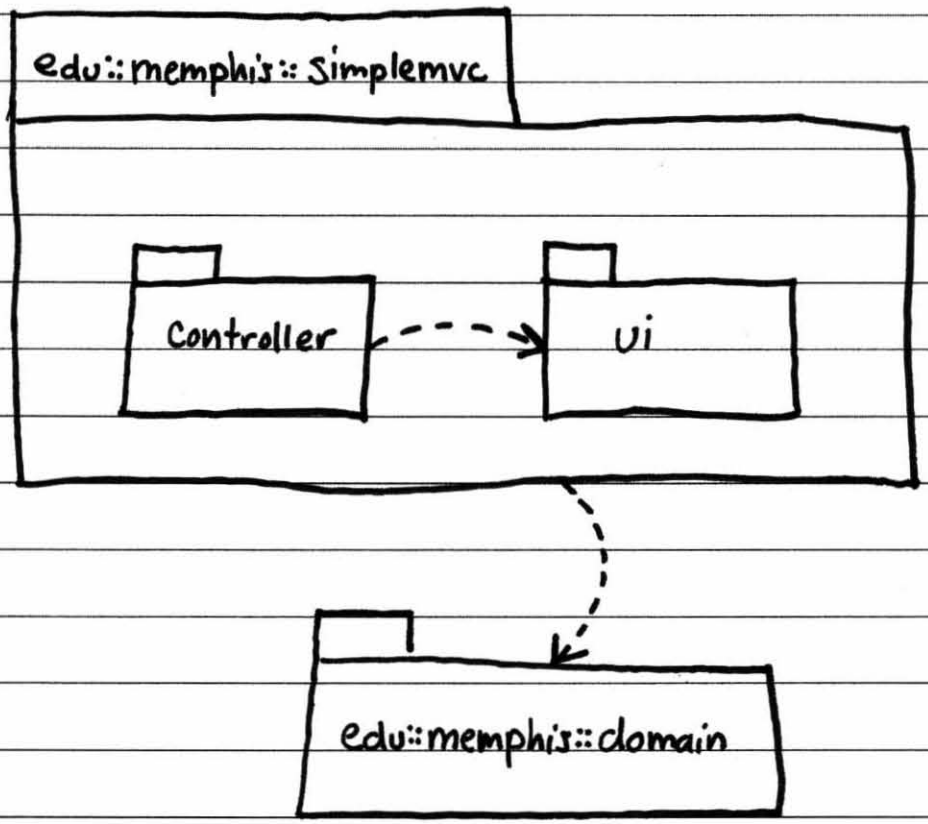
- No bathroom breaks.

- No calculators.
- No cellphones—turn them off.
- No other devices.

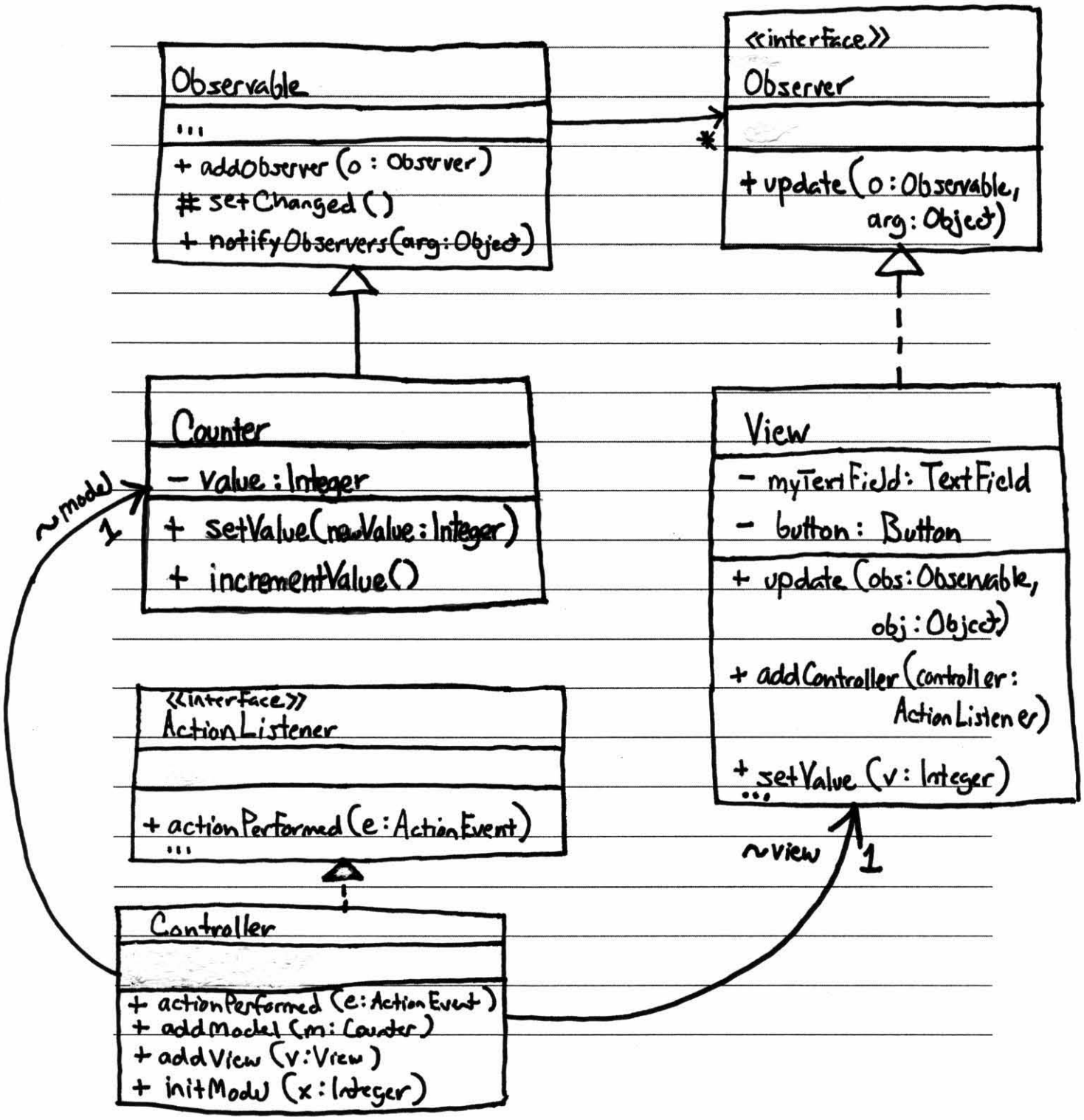
- Closed book.
- Closed note.
- Closed neighbor.

- Verify that you have all the pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

1. [16pts] Draw a package diagram representing the logical architecture of the SimpleMVC application.
   - Exclude all Java API packages (i.e., all packages that begin with "java.").
   - Do not explicitly model the "edu" and "edu.memphis" packages (because they just aren't interesting).
   - Model all dependencies.



2. [18pts] Draw a Design Class Diagram (DCD) that models the SimpleMVC application.
   - Model only the following classes: Counter, View, Controller, Observable, ActionListener, and Observer.
   - Don't forget to include all generalizations and associations among these classes.
   - Don't forget to include all attributes and operations of these classes, including types and visibilities.



3. [3pts] Which of the following best characterizes the Law of Demeter?
   a. Thou shalt not kill.
   b. Don't talk to strangers.
   c. Build bridges, not walls.
   d. A bird in the hand is worth two in the bush.
   e. A chain is as strong as its weakest link.

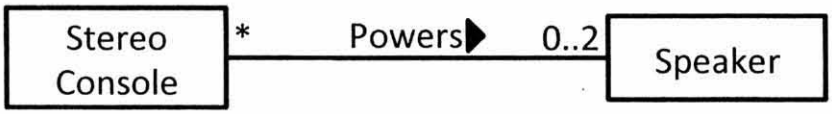
4. [3pts] What is the expected relationship between coupling and cohesion?
   a. As coupling increases, cohesion decreases.
   b. As coupling increases, so does cohesion.
   c. Making the classes within a package more tightly coupled increases the packages cohesion.
   d. Only classes have cohesion, and only packages have coupling.
   e. None of the above. Coupling and cohesion are independent.

5. [3pts] How are the Domain Model and the Domain Layer of a layered logical architecture related?
   a. They have exactly the same classes.
   b. Classes in the Domain Layer inspire classes in the Domain Model.
   c. Classes in the Domain Model inspire classes in the Domain Layer.
   d. Classes in the Domain Model become packages in the Domain Layer.
   e. They both contain only non-fabricated classes.

6. [3pts] According to the Creator Pattern, a class B should be responsible for creating instances of A if _____(fill in the blank)_____.
   a. B "contains" A
   b. B records A
   c. B closely uses A
   d. B has initializing data for A
   e. Any of the above.

7. [3pts] According to the Information Expert pattern, you should assign a knowing responsibility to a class that _____(fill in the blank)_____.
   a. is from the Domain Model
   b. is a fabricated class
   c. has the information necessary to fulfill the responsibility
   d. has polymorphic operations and implements a general interface
   e. Any of the above.

8. [3pts] Which of the following statements correctly expresses the semantics of following class diagram?



a. A stereo console powers between 0 and 2 speakers
b. A speaker is powered by 0 or more stereo consoles
c. A stereo console powers 0 or more speakers
d. A speaker is powered by between 0 and 2 stereo consoles
e. Both a and b
f. Both c and d
g. a, b, c, and d
h. None of the above

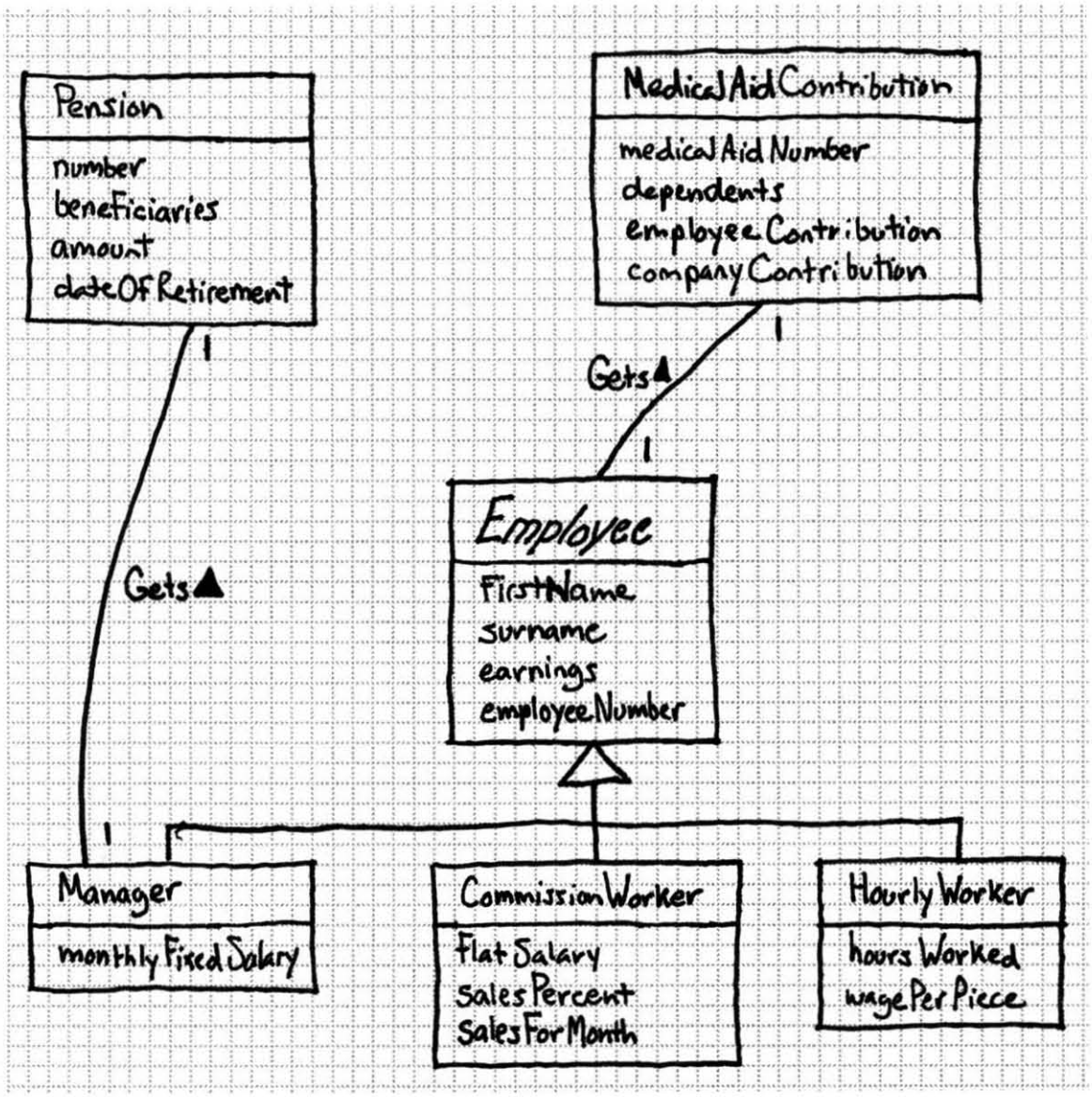
9. [3pts] Which of the following is true of coupling?
   a. Less coupling is generally better.
   b. Java class C is coupled to class D if C invokes operations of D.
   c. Java class C is coupled to class D if C inherits from D.
   d. Coupling is a measure of how strongly one element is connected to others, has knowledge of others, and/or relies on others.
   e. All of the above.

10. [18pts] Create a Domain Model in the form of a class diagram based on the following description. Model only things that are specifically described. Do not model "the system."

    A company requires a payroll system. The company has three types of employees, namely, a manager, a commission worker, and an hourly worker. Information that needs to be stored for each employee is the employee's first name, surname, earnings and employee number. Additional information that must be stored for a manager is his/her monthly fixed salary; for a commission worker his/her flat salary, sales percent and the sales made for the month; for an hourly worker the number of hours worked and the wage per piece. All employees get a medical aid contribution. In addition to this managers get a pension as a fringe benefit. Each medical aid instance specifies the medical aid number, dependents, employee contribution and company contribution, while each pension instance specifies the pension number, beneficiaries, amount and date of retirement.

Draw your diagram on the next page.

Answer question 10 here.



11. [18pts] Unscramble the JSP version of the SimpleMVC program. You may write your answer as just the letters (a–r) or as both the letters and the actual lines of code.

j <html>

o <head> ... </head>

i <body>

k <%

b if (session.isNew()) {

f session.setAttribute("myCounter", new Integer(10));

h } else {

n session.setAttribute(...);

l }

q %>

m Counter:

e <%=

p session.getAttribute("myCounter")

q %>

c

d PressMe

g </body>

a </html>

12. [3pts] What was the stimulus that led to "net neutrality" legislation?
    a. Internet service providers' suggestion that they may begin a tiered service, charging more for higher-priority routing of packets.
    b. Internet censuring by foreign countries.
    c. The proliferation of pornography on the Web.
    d. The Arab Spring.
    e. None of the above.

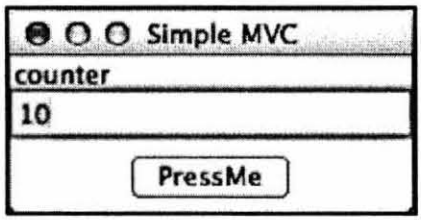
13. [3pts] Which one of the following is an argument for globalization?
    a. American workers should not be forced to compete with foreign workers who do not receive decent pay and working conditions.
    b. Both manufacturing jobs and white-collar jobs overseas are being lost at an accelerating rate.
    c. The removal of trade barriers hurts workers in foreign countries, too.
    d. People in poorer countries deserve jobs, too.
    e. None of the above.

14. [3pts] If package **ui** contains package **web** contains package **jsp**, then what is the fully qualified name for **jsp** in UML syntax?

ui :: web :: jsp

SimpleMVC Program

In this exam, we will use as a running example a small program that purports to use the MVC pattern. The UI for the program looks like this:



Initially, the counter is set to 10. The user can press the "PressMe" button to increment the counter by one.

Relevant Source Code

MyMain.java

```
package edu.memphis.simplemvc;

import edu.memphis.domain.Counter;
import edu.memphis.simplemvc.controller.Controller;
import edu.memphis.simplemvc.ui.View;

public class MyMain {

	private static int start_value = 10;

	public static void main(String[] args) {
		Counter myModel = new Counter();
		View myView = new View();
		Controller myController = new Controller();

		myModel.addObserver(myView);
		myController.addModel(myModel);
		myController.addView(myView);
		myController.initModel(start_value);
		myView.addController(myController);
	}

}
```

Counter.java

```
package edu.memphis.domain;

import java.util.Observable;

public class Counter extends Observable {

	private int value = 0;

	public void setValue(int newValue) {
		this.value = newValue;
		setChanged();
		notifyObservers(newValue);
	}

	public void incrementValue() {
		++value;
		setChanged();
		notifyObservers(value);
	}

}
```

Controller.java

```
package edu.memphis.simplemvc.controller;

import java.awt.event.*;
import edu.memphis.domain.Counter;
import edu.memphis.simplemvc.ui.View;

public class Controller implements ActionListener {

	Counter model;
	View view;

	public void actionPerformed(ActionEvent e) {
		model.incrementValue();
	}

	public void addModel(Counter m) { this.model = m; }
	public void addView(View v) { this.view = v; }

	public void initModel(int x) { model.setValue(x); }

}
```

View.java

```
package edu.memphis.simplemvc.ui;

import java.awt.*;
import java.awt.event.*;
import java.lang.Integer;
import java.util.*;


public class View implements Observer {

      private TextField myTextField;
      private Button button;

      public View() {
            Frame frame = new Frame("Simple MVC");
            frame.add("North", new Label("counter"));

            myTextField = new TextField();
            frame.add("Center", myTextField);

            Panel panel = new Panel();
            button = new Button("PressMe");
            panel.add(button);
            frame.add("South", panel);

            ...
            frame.setSize(200, 100);
            frame.setLocation(100, 100);
            frame.setVisible(true);

      }

      public void update(Observable obs, Object obj) {
            myTextField.setText("" + ((Integer)obj).intValue());
      }

      public void addController(ActionListener controller) {
            button.addActionListener(controller);
      }

      public void setValue(int v) { myTextField.setText("" + v); }

      ...

}
```

Relevant Excerpts from the Java API

public class Observable

An observable object can have one or more observers. An observer may be any object that implements interface `Observer`. After an observable instance changes, an application calling the `Observable` object's `notifyObservers` method causes all of its observers to be notified of the change by a call to their `update` method.

- `public void addObserver(Observer o)`
  - Adds an observer to the set of observers for this object, provided that it is not the same as some observer already in the set.
- `protected void setChanged()`
  - Marks this `Observable` object as having been changed
- `public void notifyObservers(Object arg)`
  - If this object has changed, as indicated by the `hasChanged` method, then notify all of its observers, and then call the `clearChanged` method (not shown) to indicate that this object has no longer changed. Each observer has its `update` method called with two arguments: this observable object and the `arg` argument.

public interface Observer

A class can implement the `Observer` interface when it wants to be informed of changes in observable objects.

- `void update(Observable o, Object arg)`
  - This method is called whenever the observed object is changed. An application calls an `Observable` object's `notifyObservers` method to have all the object's observers notified of the change.

public interface ActionListener

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's `addActionListener` method. When the action event occurs, that object's `actionPerformed` method is invoked.

- `void actionPerformed(ActionEvent e)`
  - Invoked when an action occurs.

Scrambled JSP Version of SimpleMVC

The following is a JSP implementation of the SimpleMVC program—only it's scrambled!

a.
```
</html>
```

b.
```
if (session.isNew()) {
```

c.
```
</p>
```

d.
```
<a href=". ">PressMe</a>
```

e.
```
<%=
```

f.
```
session.setAttribute("myCounter", new Integer(10));
```

g.
```
</body>
```

h.
```
} else {
```

i.
```
<body>
```

j.
```
<html>
```

k.
```
<%
```

l.
```
}
```

m.
```
<p>
counter:
```

n.
```
session.setAttribute("myCounter",
  new Integer(((Integer)session.getAttribute("myCounter")).intValue()+1));
```

o.
```
<head>
<title>Counter</title>
</head>
```

p.
```
session.getAttribute("myCounter")
```

q.
```
%>
```

q.
```
%>
```

} Because these two are the same, they have the same letter

The JSP looks like this in a web browser: