Homework 2: Design Patterns and Code Generation using Rational Software Development Platform

Laboratory Date: Monday, October 10, 2011 Take Home Due: Thursday, October 20, 2011

Name	:	
Student Number	:	
Laboratory Time	:	

Objectives

- Create a Class Diagrams using the Design Patterns feature in Rational Software Development Platform
- Generate Java code from a Class Diagram in Rational Software Development
 Platform
- Modify and run a Java application in Rational Software Development Platform

Preamble

In this lab, we will learn how to create a UML Class diagram from existing Java source code. We will also learn how to create Class Diagrams including Design Patterns in Rational Software Development Platform. Specifically, we will be using the Observer Design Pattern. Finally, we will learn how to generate Java code from a UML Diagram and how to run Java applications using the Rational tool.

Grading Checklist (30 points)

By the end of the laboratory session, you need to demonstrate to the TA that you can do the following tasks. The TA will check off the items below that you have completed and collect this cover page from you.

- □ Create a Class Diagram from Lunar Lander code
- □ Create a class Diagram showing the Observer Design Pattern, and all attributes and methods for the concrete classes
- □ Generate Java code from the Class Diagram
- □ Add Java code and run an application using the Observer pattern
- □ Implement the update method for the Observer that prints the employee's names

TA Initials: _____

Instructions for the Laboratory

Task 1: Familiarize yourself with the scenario.

INF111 Software Inc, a company which develops software for grocery stores, has recently started to implement an emergency preparedness plan for protecting its employees from emergencies such as fires or earthquakes. INF111 Software Inc needs to know the list of employees who are in the building at any moment, so that this list could be used to alert employees and take emergency actions accordingly.

INF111 Software Inc has provided a badge, which has an RFID tag located on the back, to each employee. RFID readers have been installed in the two entrances of the company. Every time an employee enters to the building the RFID reader will read the employee's RFID tag and it will add the employee's first and last name in the list of people who came to work.

Your task is to create an application that will receive the employee's name and add it to a list. Each time and employee's name is added, you should print the number of employees and the list of employee's names that are in the building.

While thinking about the design of your implementation, you have determined that you should keep track of the list of people and trigger two events when a new element is added. You have decided that you can use the Observer Pattern to create this application, where the list of employees will have the Subject role and the classes that handle the counting of employees and the printing of names will have the Observer role.

Knowing that Rational Software Development Platform provides support to design applications using Design Patterns and also provides support to generate code, you decided to model your system with this tool and then add your customized code to meet the requirements.

Task 2: Set up a Project in Rational Software Development Platform

For this task, you will create a class diagram to include the Observer Design Pattern.

- a) Make sure you have the directory "UMLDiagrams" in your H directory. If not, create it.
- b) Start Rational Software Development Platform.



c) Change the Workspace location to H:\UMLDiagrams\workspace

Workspace Launcher	×
Select a workspace IBM Rational Software Development Platform stores your projects in a director Select the workspace directory to use for this session.	ory called a workspace.
Workspace: H:\UMLDiagrams\workspace	Browse
Less this as the default and do not ask again	OK Cancel

d) Create a new Project. Go to File -> New -> Project

©	lodel	ling - IBM	Ration	al Softw	are Deve	lopm	ent Plati	iorm
File	Edit	Navigate	Search	Project	Modeling	Run	Window	Help
1	Vew		Alt+Si	nift+N	🕨 对 Proj	ect		

e) Select the UML Project wizard. Click Next.

🛛 New Project 🛛 🗙
Select a wizard Create a new UML modeling project
Wizards: Managed Make C++ Project WiLl Project Will Project Modeling
< Back Next > Einish Cancel

f) In the Project name field, enter "ClassDiagramWithPatterns". Click Next.

🔍 UML Modeling Project 🛛 🗙
UML Modeling Project A new UML modeling project with a readied empty model well suited for modeling.
Project name: ClassDiagramWithPatterns Project contents ✓ ✓ Use default Directory: H:\UMLDiagrams\workspace\ClassDiagramWithPatterns Browse
< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

g) In the File name field, enter "ClassDiagramWithObserver". Click Finish.

🔘 New UML Model		
Create UML Model Create a new UML model		
File types:	Templates: Bank Model Corba Template Model Enterprise IT Design Mod Sulse Case Model Sa XSD Model	6
Description: Create a blank UML model. File name:		<
ClassDiagramWithObserver		_
Destination folder:		
ClassDiagramWithPatterns		Browse
Default diagram	model.	
	Einish	Cancel

Task 3: Create a Design Pattern in Rational Software Development Platform

a) Open the Pattern Explorer View. To do that, go to Window -> Show View -> Pattern Explorer.

Window Help					
New Window					
Open Perspective					
Show View 🔹 🕨	💷 Bookmarks				
Customize Perspective	📃 Console				
Save Perspective As	🔝 Diagram Navigator				
Reset Perspective	Eg Inheritance Explorer				
Close All Perspectives	Rega Model Explorer				
Navigation •	🔚 Model Query Results				
New Model Explorer	e Outline Alt+Shift+Q, O				
Droforoncoc	🚯 Palette				
Freierences	🌍 Pattern Authoring				
	😪 Pattern Explorer				

b) In the Pattern Explorer, expand Design Patterns, expand Behavioral, and expand Observer. Select ConcreteSubject. Right click on it. Select Apply Pattern.

🗣 Pattern Explorer 🗴 🛛 🙂 💅 🛷 👻 🗖 🗖	
🖃 🛷 Design Patterns	
🚊 🛷 Behavioral	
🗄 🚽 Chain of Responsibility	
🗄 🐗 Command	
🕂 🚭 Interpreter	
🕂 🚭 Iterator	
🕀 🚚 Mediator	
🕀 🐗 Memento	
🖻 📲 Observer	
ConcreteSubject	
G Concre Show Pattern Documentation	
🕂 🚑 State 💦 Apply Battern	
🗄 😅 Strategy	

c) In the Pattern Instance Target, select ClassDiagramWithObserver as the Location. Click Next.

Apply Pattern "Observer"	
Pattern Instance Target Select the location where the pattern inst	ance will be created.
Location: ClassDiagramWithObserver Selected location: ClassDiagramWithO ClassDiagramWithO Create pattern instance on diagram:	Observer ClassDiagramWithObserver::Main
	Mew Pattern Documentation
< <u>B</u> ack	Next > Einish Cancel

d) In the Pattern Parameters window, specify the Value "ListOfEmployees" for the ConreteSubject parameter and the Value "EmployeesCounter" for the ConcreteObserver parameter. Click on Add Value and specify the Value "EmployeesNamesPreview." Click Finish.

App	ply Pattern "Observe	r"	
atteri Specif	n Parameters y values for pattern parar	neters.	
Datte	rn narameter values:		
<u>F</u> atte	Parameter Name	Туре	Multiplicity Value
	ConcreteSubject	G Class	1 G ListOfEmployees
	ConcreteObserver	G Class	0* G EmployeesCounter
		🖸 Class	0* G EmployeesNamesPreview
			Ì
<u> </u>		1111	
			Add Value Delete Value
Conc	reteObserver parameter (description	
Cond	creteObserver maintains a	reference to a ConcreteSubject	object. It stores state that should stay consistent with
the s	subject's. It implements th	e Observer updating interface to	b keep its state consistent with the subject's.
			View Pattern Documentation

e) You will see the following pattern in your class diagram.

	«Pattern Instance»	
Observer	Pattern Parameters	
ConcreteSubject [1] :	O ⊡ListOfEmployees	÷
ConcreteObserver [*] :	ComplexesCounter EmployeesNamesPreview	ź

f) In the Model Explorer, you will see that 3 classes have been created. Drag and drop each of the three classes to your diagram.



g) Your will see the Observer Pattern Instance and also one ConcreteSubject and two ConcreteObserver classes in your diagram.





h) Add all the attributes and methods to each class as shown in the following diagram.

Note: To see the complete signature of the methods. Select one class. Right click. Select Filters -> Show Signature.

**** NOTE:** Notice that in ListOfEmployees class, you have two variables of type java.util.ArrayList. To create an attribute of a primitive type ArrayList, click on the Class name until you see a pop up menu to add new Attribute. Click on "Add mew Attribute" button. Once a textbox appears for you, type in the attribute, keep typing both name and the type as seen in the picture below. Once you do this, you will have an ArrayList as a type in a primitive type group which you can use when creating attributes with other methods. Do the same for java.util.Observer.



Task 4: Generate Java code from the Class Diagram using Design Patterns in Rational Software Development Platform

Now that you designed your application, you can generate the Java code for it using the Rational tool.

a) Select the model ClassDiagramWithObserver. Right click and select Transform -> Run Transformation -> UML to Java.

ClassDiagramWithPatternsTest ClassDiagramWithObserver1 ClassDiagramWithObserver1 ClassDiagramWithObser PrimitiveTypes ClassDiagramWithObserver1 Main ClassDiagramWithObserver1 ClassDiagramWithObserver	Fest.emx * verTed Add UML Add Diagram Add Shortcut		•		
⊕ «ConcreteObserver: ■ ⊕ ■ ⊕ ■ ■ ⊕ ■ ■ ⊕ ■	Visualize		٠l		
⊕ ···· «Pattern Instance» 	撞 Rename	F2			
🕀 😥 UseCase_ClassDiagrams	Refactor		١		
⊕ ⊕ UseCase_ClassDiagramsTest	Find/Replace				
	Import Model Library				
E Outline 🛛 Inheritance Explorer	🖆 Run Validation				
	📑 Model Query				_
	Transform		Þ	🔡 UML to Java	
	Show Properties View			Run Transformation 🕨	E UML to C++
	Properties			-	IML to CORBA
					🏬 UML to Java
					💵 UML to XSD

b) Click on "Create new Target Container." In the Create a Java project window, enter **ClassDiagram_CodeGenerated**. Click Finish.

Section 1 Run Transformation	
Run this Transformation Select a Java project as the t	🔍 New Java Project. 🗙
Name: UML to Java	Create a Java project Create a Java project in the workspace or in an external location.
Source: ClassDiagramWith Use the selected source Make the selected source Target Properties Collection UML Element Create new Target Container.	Project name: ClassDiagram_CodeGenerated Location C Create project in workspace C Create project at external location Directory: H:\UMLDiagrams\workspace Project layout Project layout C Use project folder as root for sources and class files C Create separate source and output folders Configure Defaults

c) In the Run Transformation window, click Apply and Run.

🔡 Run Tra	nsformation 🛛 🗙
Run this Tra	nsformation
<u>N</u> ame: <u>S</u> ource: ☐ Use ☐ Maka	UML to Java ClassDiagramWithPatterns the selected source, not the saved source the selected source the saved source
Target Pro UML Elemen	sperties Collections Mapping Common t ssDiagram_CodeGenerated
(Create per	w Tarnet Container

d) Notice that the Java code has been generated for the classes you had in your diagram. Inspect the code that has been generated in each class.



Task 5: Add the Java code to run an application using the Observer Pattern

As you can see in each class, the generated Java code matches with the properties and method signatures you specified in you class diagram. The code generation is a very helpful feature because you already have the code with all the structure of your classes. However, you need to add the specific functionality that you want each method to execute. Now, you will add the code that is needed to run the application based on the created structure.

Add source code to Classes

Using the Model explorer make the following changes in the Java classes.

- a) Class: ListOfEmployees.java
 - The declaration of the private properties listOfPeople and observers is already in the source code. You should modify this declaration to add the initialization of the ArrayList in each case. The properties should look like:

```
private ArrayList listOfPeople = new ArrayList();
.....
private ArrayList observers = new ArrayList();
```

• Method: add. Add the following lines:

listOfPeople.add(s); notifyObservers();

• Method: iterator. Replace return null for:

return listOfPeople.iterator();

• Method: addObserver. Add the following line:

observers.add(o);

• Method: removeObserver. Add the following lines:

observers.remove(o);

• Method: notifyObservers. Add the following lines:

```
// loop through and notify each observer
Iterator i = observers.iterator();
while( i.hasNext() ) {
    Observer o = ( Observer ) i.next();
    o.update( this,o );
}
```

- b) Class: EmployeesCounter.java.
 - Import java.util.Iterator in the class.
 - Method: EmployeesCounter. Add the following lines:

```
this.list = list;
list.addObserver( this );
```

```
• Method: update. Add the following lines:
if( o == list ) {
    System.out.println( "The contents of the list of employees have changed." );
    int counter = 0;
    Iterator i = list.iterator();
    while( i.hasNext() ) {
        String line = ( String ) i.next();
        counter++;
    }
    System.out.println( "The total number of employees in the building is: " + counter);
    }
```

c) Class: EmployeesNamesPreview.java

Method: EmployeesNamesPreview. Add the following lines:

this.list = list; list.addObserver(this);

Create class to run the application using the Observer Pattern

d) Create a Class called TestObserverPattern. Right click on the project ClassDiagram_CodeGenerated. Select New -> Other. Select Java -> Class. Click Next.

New	X
Select a wizard Create a Java class	
<u>W</u> izards:	
 Example EMF Model Creation Wizards Java Tuterface Java Project Package Source Folder Java Run/Debug Java Run/Debug Modeling Class Diagram Static Method Sequence Diagram Static Method Sequence Diagram Topic Diagram 	
☐ Show All Wizards.	0
< <u>B</u> ack Next > Einish	Cancel

e) Enter the name of the Class: TestObserverPattern. Click Finish.

🥝 New Java Clas	s		×
Java Class	fault package is discouraged.		C
Source Folder:	ObserverPattern_CodeGenerated		Browse
Package:		(default)	Bro <u>w</u> se
Enclosing type:			Bro <u>w</u> se
Name: Modifiers:	TestObserverPattern ☞ gublic C default C prig □ abstract □ final □ stat	ate C protected	
Superclass:	java.lang.Object		Browse
Interfaces:			Add
Which method stubs	 would you like to create? _ public static void main(String[] args) 		
	< <u>B</u> ack Mext	: > Einish	Cancel

f) Add the following method to the class.

public static void main(String [] args) {
 String employee1 = "Peter Anteater";
 String employee2 = "Bucky Badger";
 String employee3 = "Buster Bronco";
 String employee4 = "Donald Duck";
 String employee5 = "Herbie Husker";

String employee6 = "Blizzard Husky"; ListOfEmployees list = new ListOfEmployees(); list.add(employee1); list.add(employee2); list.add(employee3); list.add(employee4); //counter and preview add themselves to the listOfPeople EmployeesCounter counter = new EmployeesCounter(list); EmployeesNamesPreview preview = new EmployeesNamesPreview (list); System.out.println("An employee has entered to the building:"); list.add(employee5); System.out.println(""); System.out.println("An employee has entered to the building:"); list.add(employee6); System.out.println("");

Run the application using the Observer Pattern

}

g) Select the class TestObserverPattern and go to Run -> Run as -> Java Application

Run Window Help		
O Internal Tools		▶ 🗸
🗞 Run Last Launched	Ctrl+F11	🕖 StringPreview.j
🍬 Debug Last Launched	F11	
Run History		•
Run As		🗾 🛃 1 Java Applet
Run		🗊 2 Java Application
Debug History		Ju 3 JUnit Test
		•

h) You should see the following output:



Task 6: Implement the update method for the Observer EmployeesNamesPreview

- a) Implement the update method in the EmployeesNamesPreview Class, so that you will print in the screen the names of the people who are in the building each time one employee is added to the list. Hint: You should use the method update in EmployeesCounter.java as a reference.
- b) After you finished with the implementation, run the program. You should have an output similar to the one showed in the following screen.

Properties Tasks 📮 Console 🛛 Bookmarks
<terminated> TestObserverPattern (2) [Java Application] C:\Program Files\IBM\Ration</terminated>
An employee has entered to the building:
The contents of the list of employees have changed.
The total number of employees in the building is: 5
The contents of the list of employees have changed.
The employees who are in the building are:
Peter Anteater
Bucky Badger
Buster Bronco
Donald Duck
Herbie Husker
An employee has entered to the building:
The contents of the list of employees have changed.
The total number of employees in the building is: 6
The contents of the list of employees have changed.
The employees who are in the building are:
Peter Anteater
Bucky Badger
Buster Bronco
Donald Duck
Herbie Husker
Blizzard Husky

Take Home

1. (40 points) Identify two design patterns used in DVDVendor. For each design pattern: i) Create a class diagram containing classes and interfaces involved in the pattern.

- Use a class diagram of each design pattern in the lecture slides as a template for arranging the classes and relationships.
- Use either Note or Stereotype to show the role of each class or template in the design pattern.
- For each classes and interface, you don't need to include all attributes and methods. However you need to include attributes and methods that are involved in the pattern.

ii) Explain how the pattern in the DVDVendor software works. What system or user event causes them to be triggered?