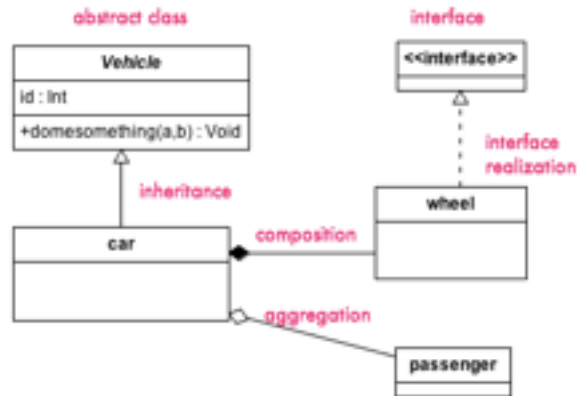


CS330 Design Patterns - Midterm 1

Name: _____

October 27th 2011

Please read all instructions carefully. The exam is closed book & no laptops / phones / computers shall be present nor be used. Please write your answers in the space provided. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper. Solutions will be graded on correctness and clarity. You can score a total of **100** points on this exam. For every question you may use class diagrams or java/C# code to illustrate your written answer, unless the question asks for a specific format. When you have to draw a class diagram, indicate the **methods** that make the pattern work also specify the type of **relation** (inheritance / composition) between classes and **type** of class (interface / abstract class / regular) as indicated in the sample UML class diagram. For each class & interface in your diagram also indicate what it does. Draw the right type of arrows in you diagram. You cannot ask any questions to the person who proctors this midterm. If you don't understand the question, write down what you don't understand. Solve the question the way you interpret it based on what you don't understand. Just don't simplify the question to something trivial as that will get you 0 points.



1: **(5 points)** What is one of the most important benefits that design patterns provide to **communities** of developers?

Design patterns give you a shared **vocabulary** with other developers. Once you've mastered the vocabulary you can more easily communicate with other developers and inspire those who don't know patterns to start learning them. Patterns let you communicate solutions very efficiently using their name. Patterns further let you think more abstractly about problems without getting lost into details (page 28).

2. **(5 points)** Why aren't there libraries of patterns? e.g. why isn't there an API for each programming language that lets you easily implement a number of patterns?

Design patterns are defined at a **higher** level of abstraction than a library which is typically specific to a particular problem (e.g. communication / rendering graphics, etc). Design patterns tell you how to structure classes and objects to solve certain problems and it is the programmer's job to adapt those designs to fit a particular application. (page 29)

CS330 Design Patterns - Midterm 1

3. (5 points) Name (at least) two **properties** of software designs that design patterns seek to improve through their application.

Software design can easily result in spaghetti code. Patterns tell you how to structure classes and objects so your software design becomes easier to:

- 1) **understandability**. Software designs are a common understanding of the intended system. If it is easier to understand this design less mistakes are made.
- 2) **maintainability**. Design patterns may increase the maintainability of your code for example when you encapsulate changes such as in the strategy pattern.
- 3) **flexibility**. Application of design patterns may allow for easily extending your software design with new functionality.

page 5 to 23

4. (10 points) Explain how the **factory method** pattern works. You may use an example to illustrate how this pattern works (but not the example from the book).

The factory method encapsulates object creation by letting a subclass decide what objects to create. The benefits of this is that it decouples clients from concrete classes and any change in creation of objects is localized to a single class. For example you could have an application that creates different types of pets (cats / dogs) using an abstract pet store. For specific types of pets (e.g. a dog you have concrete creators.

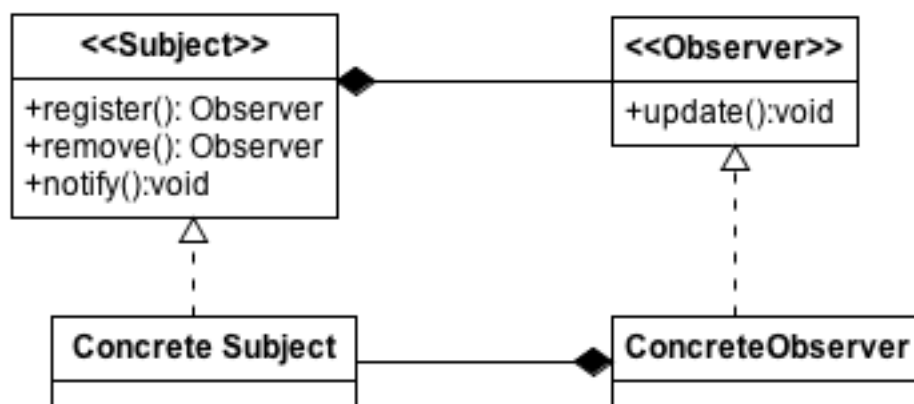
CS330 Design Patterns - Midterm 1

5. (10 points) Which three object oriented design principle(s) are involved in the **Strategy** design pattern? For each object oriented design principle explain the reasoning behind the principle in the context of using the strategy pattern.

The following Object oriented design principles are involved in the Strategy pattern:

1. **Encapsulates what varies** ~ take the parts of your code that vary and encapsulate these, so later you can easily **change** or extend the functionality in the parts that vary without affecting the parts of your code that do not vary (page 9)
2. **Program to an interface not to an implementation** ~ the idea is to use polymorphism so the actual runtime object isn't locked into the code which makes it hard to **change** later on. Instead you program to an interface or super type. (page 12)
3. **Favor composition over inheritance** ~ creating systems using composition gives you more **flexibility**. You encapsulate a family of algorithms into their own set of classes. and it lets you change their behavior at runtime. (page 23)

6. (5 points) Give a class diagram of the **Observer** design pattern. Follow the instructions for drawing and describing a class diagram on the first page of this midterm.



CS330 Design Patterns - Midterm 1

7. (5 points) What problems do the **command** and the **decorator** pattern address? You may illustrate your answer with code or a class diagram.

The decorator pattern addresses the problem that an overuse of inheritance leads to an explosion of classes which are difficult to maintain or modify. (page 81)

The command pattern addresses the problem that when your program has many different actions it can perform, implementing these would lead to huge if-elseif or switch blocks, which are difficult to extend or to maintain. (see slides command pattern)

8. (5 points) Which object oriented design principle(s) are involved in the **Decorator** design pattern? For each object oriented design principle discuss the motivation behind this principle in the context of the decorator pattern.

Classes should be open for extension but closed for modification ~ the idea is to close a class for modification but to open them for extension to incorporate new behavior. This will achieve a design that is resilient to change and flexible enough to take on new functionality to meet changing requirements.

9: (5 points) Assign each design pattern to the right category by placing an 'X' at the right spot. Each pattern belongs to one category.

pattern	structural	creational	behavioral
abstract factory		X	
command			X
decorator			X
observer			X
singleton		X	
strategy			X

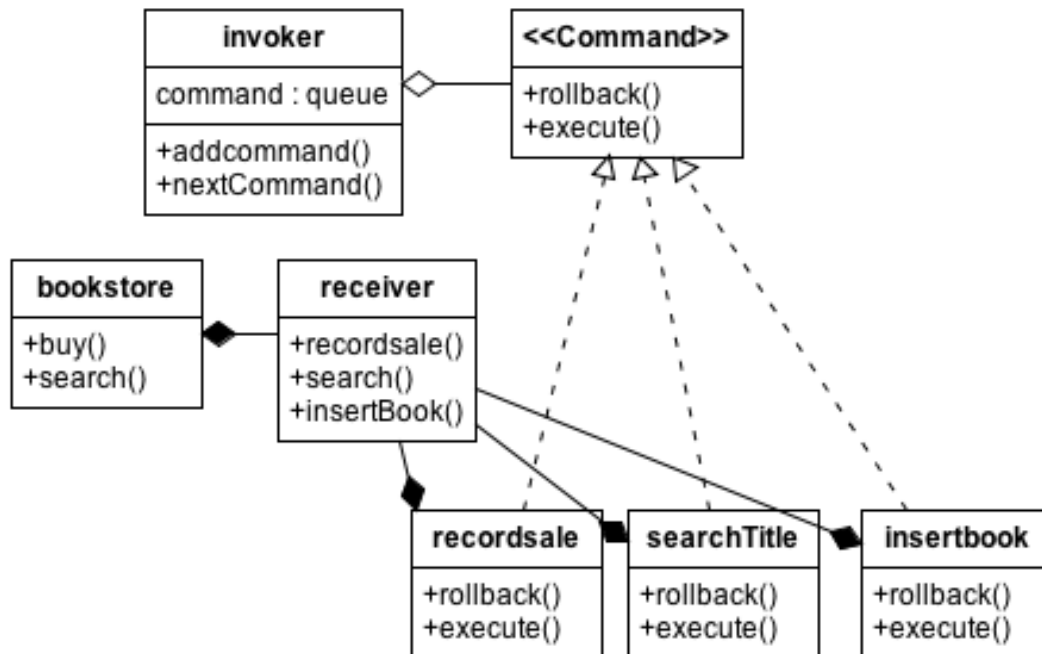
CS330 Design Patterns - Midterm 1

Design Problems

A number of design problems are listed below. For each problem illustrate which design pattern you should use and how it specifically addresses the problem or system requirement. Use a class diagram or source code to illustrate your solution.

10: **(15 points)**. For an online bookstore you need to design a database engine. This engine processes transactions that it receives from it's users through a web based interface. A database transaction could be the insertion of a new book, the recording of a sale, or searching for particular books on different fields (title/author). Your engine must keep a list of transactions that need to be executed and cues these as it may take a while for a transaction to complete. Transactions are performed sequentially. Should one of these transactions fail, all others can be **reverted** (called a rollback). For example, if two database tables which refer to each other must be updated, and the second update fails, the transaction can be rolled back, so that the first table does not now contain an invalid reference.

The solution here is to use the command pattern because of the rollback requirement which is a form of undo and the command pattern can provide a generic framework for implementing generic actions. Make all transactions implement your command interface and implement an rollback for each transaction. See the class diagram below.



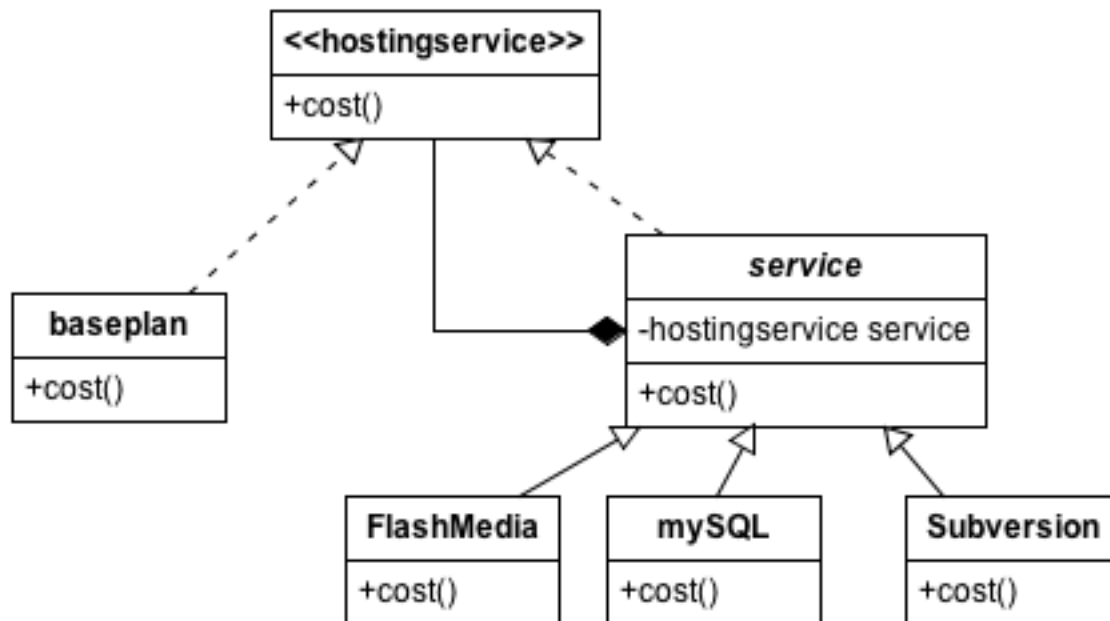
CS330 Design Patterns - Midterm 1

11: **(15 points)** You work for a webhosting company that offers a ton of hosting services (Flash Media, MySQL, Subversion) to its customers in addition to a base web hosting plan. You need to write an application that can easily compute the total monthly service fees for each plan. Your application must be able to easily support adding **new** types of hosting services (such as Ruby on Rails) when they become available.

Sample output could be:

> Basic Hosting, Subversion Hosting, Flash Media Server Hosting, MySQL Hosting(w/ 3 databases): 59.94 a month.

The Decorator pattern would have to be used to avoid an explosion of classes to support the different configurations (e.g. BasicSubversionFlashMySQL.class). Adding a new service only requires 1 new class. See the class diagram below:



12: **(15 points)** You work for a company that develops medical imaging equipment. You developed a medical imaging application which collects data from a computed axial tomography (CT) scanner via TCP/IP sockets. The imaging application was originally written to

CS330 Design Patterns - Midterm 1

communicate with one scanner but was expanded when your company started to develop a functional magnetic resonance imaging (FMRI) scanner. Unfortunately, this FMRA scanner does not speak the same language as the CT scanner. Marketing requires your application to support **variability** by supporting both types of scanners, since the imaging software is the same and hospitals may use the scanners in tandem. The R&D department of your company recently announced that they will launch a photo acoustic (PA) scanner in 2011 which use a different communication protocol than the CT and FMRI scanner. Because this is medical software it runs on custom operating system that is ultra reliable and consequently does not support unreliable hardware abstraction through drivers. This means the drivers for the scanners are part of your application. Your application works as follows: you place a patient in the scanner and press the scan button upon which the scanner will send a request for image transmittal to your application. Your software keeps track of which driver is required for which IP address.

Handle different versions with the factory or strategy pattern. Create an abstract class named *DataConnection*—an interface will work as well. This allows the application to speak to both versions of the hardware. It's used to define the signature for different versions of the connection objects. Create a specific *Scanner* connection object for each scanner version that communicates differently with the application. See the class diagram below:

