Table of Contents

[2016 Lu - CS525 Quiz1 2](#_Toc479206698)

[CS525 Quiz 2 4](#_Toc479206699)

[CS525 Quiz 3 11](#_Toc479206700)

[CS525 Quiz 4 11](#_Toc479206701)

[CS525 Quiz 5 12](#_Toc479206702)

[CS525 Quiz 6 12](#_Toc479206703)

[CS525 Quiz 7 15](#_Toc479206704)

[CS525 Quiz 11 15](#_Toc479206705)

[CS525 Quiz 12 16](#_Toc479206706)

[CS525 Quiz 13 16](#_Toc479206707)

[CS525 Quiz 14 17](#_Toc479206708)

[2016 Jan Salek - CS525 Exam 1 18](#_Toc479206709)

[2016 Feb - CS525 Mock Exam 20](#_Toc479206710)

[2016 Feb Salek - CS525 Final Exam 21](#_Toc479206711)

[CS525 Quiz 1 22](#_Toc479206712)

[CS525 Quiz 2 22](#_Toc479206713)

[CS525 Quiz 3 23](#_Toc479206714)

[CS525 Quiz 4 23](#_Toc479206715)

[CS525 Quiz 5 23](#_Toc479206716)

[CS525 Quiz 6 23](#_Toc479206717)

[CS525 Quiz 7 23](#_Toc479206718)

[CS525 Quiz 8 24](#_Toc479206719)

[CS525 Quiz 9 24](#_Toc479206720)

[CS525 Quiz 10 24](#_Toc479206721)

[Question 1 [15 points] [15 minutes] 24](#_Toc479206722)

# 2016 Lu - CS525 Quiz1

1. *What is a design pattern?*

A design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.

1. *Why are design patterns important to software designers?*

They help us solve recurring design.

1. *What is a software framework?*

Frameworks are semi-completed applications that need to be customized according to user requirements to become an application for a particular organization.

1. *Why do we need software frameworks?*

**Frameworks reduce the software development effort:** Software applications = code for business logic + code for the infrastructure that holds the application components together.

**Frameworks provide generic services**: Examples include request processing, caching, logging, configuration and database access. They simplify development for event handling, user interface management, data exchange and job processing. This infrastructure code is usually the most detailed and tedious code to write, requiring deep technical skills.

**Frameworks reduce complexity**: Many of the design decisions necessary when starting development from scratch are already built into the framework, like the architecture, object creation, performance optimization, etc. therefore, developers do not spend time reinventing the wheel. Less experienced developers can still build better quality software based on framework.

**Frameworks improve productivity, quality and consistency**: Frameworks make the software development cycle a more predictable process by providing a standardized architecture and a standardized development approach. For example, developers work with pre-defined rules for coding and proven methodologies for building parts of an application. They also use a given set of development tools.

1. *Explain the 3 categories of GoF design patterns.*

**Behavioral**: patterns that are concerned with communication between objects in a design solution.

**Creational:** patterns used to create objects.

**Structural:** patterns that are concerned with class or object composition to put in place a certain capability or perform a certain task.

1. *Name 3 design patterns for each category.*

**Behavioral**: Factory Method, Abstract Factory, Builder, Prototype, Singleton

**Creational:** Adapter, Bridge, Composite, Decorator, Façade, Proxy

**Structural:** Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Flyweight, Observer, State, Strategy, Visitor

1. *What is the intent of the Singleton pattern?*

**Intent**

Ensure a class only has one instance, and provide a global point of access to it.

**Motivation**

* 1. Sometimes we want just a single instance of a class to exist in the system
  2. For example, we want jus one factory for a family of products
  3. We need to have that one instance easily accessible
  4. And we want to ensure that additional instances of the class cannot be created

**Structure**



1. *What four essential elements has a pattern?*
2. The **pattern name**
3. The **problem**: explains the problem and its context and when to apply the pattern
4. The **solution**: a template that describes the elements that make up the design – their relationships, responsibilities, and collaborations.
5. The **consequences**: are the results and trade-offs of applying the pattern.
6. *Conceptually how to implement a Singleton? Please explain with necessary code snippets.*

public class Singleton {

public static Singleton instance = new Singleton();

private Singleton() {};

public static Singleton getInstance() {

if (instance == null) {

synchronized(Singleton.class) {

if (instance == null) {

instance = new Singleton();

}

}

}

return instance;

}

public void message() {

System.out.println("hello");

}

public static void main(String[] args) {

Singleton demo = Singleton.getInstance();

demo.message();

}

}

1. *What is a software foundation?*

A software foundation is a complete application which was built with a fully-developed ‘core’ plus well-designed/documented extension mechanisms provided so other developers can easily add functions/modules to the foundation for their own application.

# CS525 Quiz 2

1. *What is the intent of the Adapter pattern?*

**Intent**

Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn’t otherwise because of incompatible interfaces.

**Motivation**

* 1. Sometimes a toolkit or class library cannot be used because its interface is incompatible with the interface required by an application
  2. We cannot change the library interface
  3. Even if we did have the source code, we should not change the library

1. *Explain how the Adapter pattern works (you need to talk about the problem it solves, and the solution that includes structure, participants and applicability, etc.)*

**Structure**

An adapter relies on object composition



**Participants**

* 1. Target: defines the domain-specific interface that client uses.
  2. Adaptee: defines an existing interface that needs adapting.
  3. Adapter: adapts the interface of Adaptee to the Target interface.
  4. Client: collaborates with objects conforming to the Target interface.

**Applicability**

Use the Adapter pattern when: you want to use an existing class that performs similar functions but with a different interface. For example, a third-party report generator based on xml input.

**Implement**

Target interface

public interface Target {

public void push(String str);

public String pop();

public boolean isEmpty();

}

Adapter

public class Adapter implements Target {

private Adaptee adaptee;

@Override

public void push(String str) {

adaptee.add(str);

}

@Override

public String pop() {

int end = adaptee.getEnd();

String str = adaptee.get(end);

adaptee.remove(end);

return str;

}

@Override

public boolean isEmpty() {

return adaptee.empty();

}

}

Adaptee

public class Adaptee {

private String[] data;

private int start;

private int end;

public String startString() {

return data[start];

}

public String endString() {

return data[end];

}

public boolean empty() {

return (end == -1);

}

public void add(String str) {

data[end] = str;

end++;

}

public void remove(int pos) {

// remove the String object at position 'pos' and bring //forward all items after it

for (int i = pos; i < end; i++) {

data[i] = data[i + 1];

}

}

public String get(int pos) {

return data[pos];

}

 ...

}

1. *True/False: The reason we cannot change the interface of the Adaptee is because we have no access to its source code.*
2. *What is the intent of the Proxy pattern?*

Provide a surrogate or placeholder for another object to control access to it.

1. *Explain how the Proxy pattern works (same as how the Adapter works.)*

Proxy is applicable whenever there is a need for a more versatile or sophisticated reference to an object than a simple pointer. Here are several situations where the Proxy pattern is applicable:

* 1. Remote Proxy – provides a reference to an object located in a different address space on the same or different machine.
  2. Protection (access) proxy – provides different clients with different levels of access to a target object.
  3. Cache Proxy – Provides temporary storage of the results of expensive target operations so that multiple clients can share the results.

**Structure**





**Participants**

1. Proxy

* Maintains a reference that allows the proxy to access the real subject.
* Provides an interface identical to Subject’s so that a proxy can be substituted for the real subject.
* Controls access to the real subject and may be responsible for creating and deleting it.
* Other responsibilities depend on the kind of proxy:
  + **Remote proxies** are responsible for encoding a request and its arguments and for sending the encoded request to the real subject in a different address space;
  + **Virtual proxies** may cache additional information about the real subject so that can postpone accessing it;
  + **Protection proxies** check that the caller has the access permissions required to perform a request.

1. Subject

* Defines the common interface for RealSubject and Proxy so that a Proxy can be used anywhere a RealSubject is expected.

1. RealSubject

* Defines the real object that the proxy represents.

**Implement**

Subject interface

public interface ICommandExecutor {

public void runCommand(String cmd) throws Exception;

}

Proxy class that implements the subject interface

public class CommandExecutorProtectionProxy implements ICommandExecutor {

private boolean isAdmin;

private ICommandExecutor executor;

public CommandExecutorProtectionProxy(String user, String pwd) {

if ("username".equals(user) && "password".equals(pwd))

isAdmin = true;

executor = new CommandExecutor();

}

@Override

public void runCommand(String cmd) throws Exception {

if (isAdmin) {

executor.runCommand(cmd);

} else {

if (cmd.trim().startsWith("del")) {

throw new Exception("'delete' command is not allowed for non-admin users.");

} else {

executor.runCommand(cmd);

}

}

}

}

Real subject class

class CommandExecutor implements ICommandExecutor {

@Override

public void runCommand(String cmd) throws IOException {

Runtime.getRuntime().exec(cmd);

System.out.println("'" + cmd + "' command executed.");

}

}

1. *For the Java Applet scenario we talked about during class, how would you implement it? Just talk about ideas. (If you need more space, use the back side)*
2. *Provide 3 additional scenarios you know of where the Proxy pattern is used.*

**Scenario 1**: Suppose we have a large collection object, such as a hash table, which multiple clients want to access concurrently. One of the clients wants to perform a series of consecutive fetch operations while not letting any other client add or remove elements. What can we do to improve the situation?

**Solution 3**: It would be nice if we could actually clone the collection only when we need to, that is when some other client has modified the collection. For example, it would be great if the client that wants to do a series of fetches could invoke the clone() method, but no actual copy of the collection would be made until some other client modifies the collection. This is a copy-on-write cloning operation. We can implement this solution using proxies. The proxy is the class LargeHashtable. When the proxy's clone() method is invoked, it returns a copy of the proxy and both proxies refer to the same hash table. When one of the proxies modifies the hash table, the hash table itself is deep-cloned.

**Scenario 2**: An Internet Service Provider notices that many of its clients are frequently accessing the same web pages, resulting in multiple copies of the web documents being transmitted through its server. What can the ISP do to improve this situation?

**Solution**: Use a Cache Proxy. The ISP's server can cache recently accessed pages and when a client request arrives, the server can check to see if the document is already in the cache and then return the cached copy. The ISP's server accesses the target web server only if the requested document is not in the cache or is out of date.

**Scenario 3:** A class library provides a Table class, but does not provide a capability to allow clients to lock individual table rows. We do not have the source code for this class library, but we have complete documentation and know the interface of the Table class. How can we provide a row locking capability for the Table class?

**Solution**: A Synchronization Proxy, which uses a locking mechanism to control the number of clients that simultaneously access the server or the real object.

1. *Compare the Adapter and the Proxy pattern.*

Similarities:

* Both use a ‘wrapper’ object for the real object that offers services of interest to the client;
* Both provide an interface that the client expects/needs;
* Both rely on object composition (or delegation) to implement it;
* Both are structural patterns;

Differences:

* Their intent is different; (this is always true. If you say this, you need to explain how as below)
* Adapter is to convert an interface that a client would not otherwise be able to use.
* Proxy is to provide a surrogate or placeholder for the client to use the real subject’s services.

# CS525 Quiz 3

1. *What is the intent of the Template Method pattern?*

Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

1. *You are asked to design a framework that other developers can use to quickly build a GUI application in Java. You should allow different approaches to create components on the GUI. For example, by providing code themselves, by reading from an XML file, or by loading components from a remote server, etc. But what is common are the steps to build the GUI –*
   1. *Initialize (setRootContainer, setBackground, enableEventHandling)*
   2. *Build components (create or load GUI components)*
   3. *Load data (populate the data-displaying components, like textbox, dropdown, etc)*
   4. *Apply layout (lay out the components)*
   5. *Apply styling (css styling)*
   6. *Display (show it on screen)*

*Implement the framework with a Template method – createGUI().*

1. *What is the intent of the Prototype pattern?*
2. *What are some of the consequences of the Prototype pattern? (give at least 3)*

# CS525 Quiz 4

1. *What is the intent of the Factory Method pattern?*
2. *What are the benefits of using a Factory Method to create objects (compared with constructors)?*
3. *A marketing company asks you to design a document generator based on a web page (in html format). For now, they want to generate pdf files, image files, or PowerPoint files. Later there can be more formats to support by your generator. For all the current 3 document types, the company can choose to print them on A4 or A5 paper either as “portrait” or “landscape”. In the future, they may also want to print them on paper of other sizes (including custom size). Design your generator and show your structure.*

# CS525 Quiz 5

1. *List 4 motivations of the Iterator pattern.*
2. *Explain how you implement an Iterator based on a given Aggregate object.*
3. *Explain the participants of the Composite pattern?*
4. *What is the main benefit of using the Composite pattern? When do you use the pattern?*
5. *True/False. An iterator will guarantee thread-safe concurrent traversals of aggregate objects.*
6. *True/False. In the Composite pattern, a Composite should know all its immediate children but its children do not know their parent.*

# CS525 Quiz 6

1. *When would you use the Prototype design pattern? Please choose all that apply:*
   1. *To abstract steps of construction of complex objects*
   2. *To create objects without knowing their type or any details of how to create them*
   3. *To avoid proliferation of the class hierarchy*
   4. *To restrict class instantiation to one object*
   5. *When creating a large/complex object that is resource intensive*
2. *What are the consequences of applying the Prototype pattern? Please choose all that apply:*
   1. *Each concrete prototype class must implement the clone method*
   2. *It makes it easier for a certain family of objects to work together*
   3. *It enables the client code to register a new concrete prototype instance at run time*
   4. *It reduces the number of classes in the class hierarchy as compared to the factory method design pattern*
3. *Which of the following are participants in the Adapter design pattern? Please choose all that apply:*
   1. *Creator*
   2. *Product*
   3. *Target*
   4. *Caretaker*
   5. *Adapter*
   6. *Converter*
4. *Which design pattern can be used to implement Lazy Loading? Please choose the best one:*
   1. *Adapter*
   2. *Prototype*
   3. *Proxy*
   4. *Singleton*
5. *You are trying to add a class already written in another application to serve clients, beside other classes, in your system. All other classes have the same interface, the incoming class has a totally different interface from what the clients expect, but contains all required functionalities. What kind of refactoring is needed to make this class fit in with minimum changes in your system? Please choose the best one:*
   1. *Apply the Proxy Pattern*
   2. *Apply the Adapter Pattern*
   3. *Create a new class which implements the expected interface and copy and paste*
   4. *The code from the class in the other application to this new class*
6. *You are building an online makeup website, which provides beside text articles, makeup tutorials as video files. Which pattern is recommended to use in such a website to deliver these media resources? Please choose only one answer:*
   1. *Adapter*
   2. *Proxy*
   3. *Prototype*
   4. *Singleton*
7. *Which of the following are participants in the Composite design pattern? Please choose all that apply:*
   1. *Aggregate*
   2. *Component*
   3. *Leaf*
   4. *Iterator*
8. *Which of the following are participants in the Factory Method design pattern? Please choose all the answers that apply:*
   1. *Creator*
   2. *Product*
   3. *Refined Abstraction*
   4. *Abstract factory*
9. *When to use the Iterator Pattern? Please choose all that apply:*
   1. *To support traversals of aggregate objects without exposing their internal representation.*
   2. *To guarantee thread-safe, concurrent traversals of aggregate objects.*
   3. *To provide a uniform interface for traversing different aggregate structures to support polymorphic iteration.*
   4. *To allow different traversal methods depending on what the client needs*
10. *What is true about using the Template Method Pattern? Please choose all that apply:*
    1. *It reduces code duplication.*
    2. *It favors inheritance over composition.*
    3. *It takes advantage of polymorphism by having the correct methods in subclasses called automatically.*
    4. *To allow runtime change of behavior by taking different steps of operations of the template method.*
    5. *The template method in the abstract class must be declared as ‘final’.*
11. *Which of the following uses Chain of Responsibility to implement itself? Choose the best 2.*
    1. *java.awt.event.ComponentAdapter*
    2. *javax.jms.QueueConnectionFactory*
    3. *The Java Servlet filter framework*
    4. *java.awt.Toolkit*
    5. *Java exception handling*
12. *When will you consider using the Command pattern?*
    1. *When you need to provide system logging for an application*
    2. *When you need to provide undo or system recovery capability for an application*
    3. *When you need to provide run-time change of behavior for an application*
    4. *When you want to avoid hardcoded associations between Widgets and actions to perform for a GUI application*
    5. *When you want to schedule execution of requests at different times*
13. *What is the intent of the Chain of Responsibility Pattern?*
14. *Draw the Structure of CoR Pattern*
15. *True/False: With the CoR pattern, it is guaranteed that all requests from the client will be processed.*
16. *True/False: In the CoR Pattern, the chain of handlers can be dynamically changed by adding new handlers.*
17. *You are designing an order processing system for a web-based business. All normal domestic orders will have to be shipped by FedEx. All large bulk orders are handled separately within the warehouse and might go to different shipping carriers. Most Favored Customers (your largest customers) may have special requirements for shipping and tracking, and they’re going to get what they want. All other international orders are handled in a generic manner. Draw both the class and object structures of your design and provide some skeleton code that shows how it works.*
18. *What is the motivation of the Command pattern?*
19. *Explain how Command pattern works? (remember we talked about how to explain)*
20. *Which methods in the Command pattern are considered “callback” methods?*

# CS525 Quiz 7

1. *Explain the responsibilities of each participant in the Builder pattern.*
2. *Template Method is to provide a skeleton algorithm for subclasses to redefine without changing the structure of the algorithm. Builder is to provide a construction process that each concrete builder follows with its own implementation of the steps. So both use a step-by-step approach to perform some functions. How do you compare the two?*
3. *With the Flyweight pattern, what does it mean by intrinsic and extrinsic state information?*
4. *Describe when the Flyweight pattern would be applicable. (5 conditions)*

# CS525 Quiz 11

1. What does the Context participant do in the State pattern?
2. What does the Context participant do in the Strategy pattern?
3. When will you consider using the State pattern?
   1. When an object’s behavior depends on its state, and it must change its behavior at run-time depending on that state.
   2. When we want to model vending machines or implement computer video games.
   3. When you want to treat each state of an object as an object that can vary independently from others.
   4. When the number of states can change at runtime without affecting the client.
   5. When you want to localize state-specific behavior in each concrete state.
   6. When you want to break a multi-conditional statement and move related conditional branches into their own class.
4. When will you consider using the Strategy pattern?
   1. When you want the client to decouple from a strategy it uses to perform a function
   2. When you want to make the change of strategy easier at runtime.
   3. When you need to provide run-time change of behavior for an application
   4. When you want your application to use different strategies for different situation during runtime.
   5. When you want to break a multi-conditional statement and move related conditional branches into their own class.
5. Compare the State and Strategy patterns. List as many similarities and differences as you can.

# CS525 Quiz 12

1. The intent of the Bridge pattern is “Decouple and abstraction from its implementation so that the two can vary independently.” What does it mean by “the two can vary independently”?
2. The Bridge pattern relies on composition to decouple abstraction from implementation. True or False?
3. We can use the Bridge pattern to make an existing implementation of an API work in our business logic. True or False?
4. What is the main difference between the Implementation and Abstraction interface?
5. When will you consider using the Decorator pattern?
   1. When you want to add responsibilities to individual objects dynamically and transparently.
   2. When you want to avoid subclassing because of a large number of independent extensions are possible and would produce an explosion of subclasses.
   3. When you need to provide run-time change of behavior for an application.
   4. When we want to change the behavior of objects of the class but not the class itself.
6. What are the advantages and disadvantages of the Decorator pattern?

# CS525 Quiz 13

1. The Java event handling model is an implementation of the Observer or Mediator pattern? Explain why.
2. “Drag and drop” is considered a UI design pattern. Can you implement it with a Mediator or Observer pattern? Explain why or why not.
3. Let’s think about the Reversi board game again. We said we could implement it with a Mediator by having 2 human players modeled as 2 instances of the Colleague class. But another opinion is to model the ‘positions’ on the board as instances of the Colleague class. Do you agree? Please explain.
4. What are the advantages and disadvantages of the Mediator pattern?

# CS525 Quiz 14

True/False Questions 1-11

1. In Strategy Pattern, strategy objects are stateful.
2. Strategy Pattern helps us eliminate conditional statements from client by encapsulating the behavior in separate Strategy classes.
3. In Observer Pattern, a mediator simplifies the way colleagues communicate with each other by replacing many-to many interactions with one-to-many interactions.
4. Mediator centralizes control which can make the mediator itself a monolith that’s easy to maintain.
5. Observer pattern supports layering of software applications.
6. Visitor makes adding new operations hard but adding a new Concrete Element class easy.
7. The MVC pattern decouples among the model, view, and controller.
8. In the state pattern, you usually set the state only once.
9. In the strategy pattern, the strategy is independent of the surrounding context.
10. In the strategy pattern, if a strategy does not work, there is a fallback to the next strategy (if any)
11. In the flyweight pattern, you always need both shared and unshared flyweight objects.
12. Explain Double Dispatch.
13. When will you consider using the Visitor pattern?
14. Considering the shopping cart example we saw yesterday, why it implemented the following methods, instead of visit (OrderItem orderItem);

public void visit(Book book);

public void visit(DVD dvd);

public void visit(Toy toy);

1. What is decoupled from what in each of the 8 patterns (after the midterm)?
2. Which patterns help us remove if/else statements? (all we have learned)
3. Which patterns help us avoid a lot of subclassing? (all we have learned)
4. What have you learned from the GoF patterns so far? (or what are the design ‘tricks’ that you have learned from all these patterns)

# 2016 Jan Salek - CS525 Exam 1

1. [5 points] – Explain **Liskov Substitution Principle**.
2. [5 points] – Explain **Single Responsibility Principle**.
3. [10 points] – For each of the following descriptions, name a pattern that best matches the description:
   1. Simplifies the interface of a group of classes
   2. Provides a way to sequentially traverse a collection of objects without exposing the collections implementation
   3. Clients treat collection of objects and individual objects uniformly
   4. Encapsulates interchangeable behaviors and uses delegation to decide which one to use
   5. Allows a group of objects to be notified when some state changes
4. [15 points] – Draw a UML diagram for the **Command Pattern** and explain the role of each class. What are the software engineering/OO benefits of this pattern?
5. [15 points] – Draw a UML diagram for the **Decorator Pattern** and explain the role of each class. What are the software engineering/OO benefits of this pattern?
6. [10 points] – Explain the **Observer Pattern** and mention at least one or two main benefits of using this pattern.
7. [5 points] – **Decorators** have the same super-type as the objects they decorate. Why?
8. [5 points] – Can **Decorators** be chained? How?
9. [5 points] – What OO principle(s) do the **Factory** patterns (all three of them) promote?
10. [5 points] – Can you subclass from a **Singleton** class? Why?
11. [5 points] – What are the similarities of **Façade** and **Adapter**?
12. [5 points] – When we are creating a **Template Method**, how do we know when to use abstract methods and when to use hooks?
13. [10 points] – What patterns do you see in the code listed below?

public class SimpleLoggerImpl implements SimpleLogger{

private static final Map<Class<?>, SimpleLogger> INSTANCE\_MAP = Collections.synchronizedMap(new HashMap<Class<?>>, SimpleLogger>());

private Logger logger;

private SimpleLoggerImpl(Logger logger) {

super();

this.logger = logger;

}

public static SimpleLogger getInstance(Class<?> clazz) {

SimpleLogger instance = INSTANCE\_MAP.get(clazz);

if (null == instance) {

synchronized(SimpleLoggerImpl.class) {

if (null == instance) {

instance = new SimpleLoggerImpl(LogManager.getLogger(clazz));

INSTANCE\_MAP.put(clazz, instance);

}

}

return instance;

}

public void log (LogLevel logLevel, String message) {

Level level;

switch(logLevel) {

case ERROR:

level = Level.ERROR;

break;

case WARNING:

level = Level.WARNING;

case INFO:

level = Level.INFO;

default:

level = Level.INFO;

break;

}

logger.log(level, message);

}

}

# 2016 Feb - CS525 Mock Exam

**Note**: Please use Eclipse to solve the following problem.

**Preparation**

1. Before you start reading the problem statement, make sure your environment is ready (JDK and STS/Eclipse). If not download from [\\CS5\Public\Courses\CS544\Salek\Software](file://CS5/Public/Courses/CS544/Salek/Software)
2. Download the code skeleton from the following and import as Maven <https://delta.cs.mum.edu/svn/salek/trunk/courses/cs525/examples/project/>

**Problem Statement**

Assume that you have been given the task of writing a program to automate certain aspects of Human Resources for a large corporation. This company consists of teams of employees and each team has a manager. Managers in turns belong to higher level teams that have their own managers and so on. There are many layers of management across company. An entry level employee can have as many as ten levels of management above him, but this number is not fixed and can change in future. Managers have all the attributes of an employee, plus a few extra. Managers have a bonus which is added to their annual salary. Managers have an annual **teamBudget** that they can spend on anything they choose (from equipment purchase to parties, etc.).

Here are the use cases that your design needs to cover:

1. Employee class must implement a method called getManagers() which should return the list of managers in order (manager, manager of manager, etc.). the list should have at least one Manager in it (in case of the CEO of the company return an empty list!).
2. Manager class must implement a method called getEmployees() which should return the list of employees that work for this manager (including the employees of employees, etc.)
3. Manager class must implement a method called getTotalSalary() which should return a Double representing the total salary of all employees under this manager plus the manager himself/herself.
4. Manager class must implement a method called getTotalAnnualSalary() which should return a Double representing the total annual salary of all employees under this manager plus the manager himself/herself (remember that managers annual salary is 12 \* salary + bonus).
5. Manager class must implement a method called getTotalAnnualBudget() which should return a Double representing the total annual salary of all employees under this manager plus the manager himself/herself plus teamBudget.

As you can see in numbers 3 – 5 there is a high demand for operations to be added to Manager class over time. Do you know of a better way of solving the problem so that operations can be added from outside rather than modifying the Manager class?

In general, how do you make this more extensible so that in future any operation can be applied to all members of the aggregate?

# 2016 Feb Salek - CS525 Final Exam

**Existing System**

Assume that you are given the HR system from the Mock Exam yesterday (we have modified it a bit to fit our needs). This is a fully functional system. In this new program **Manager** extends from Employee and Employee extends from **Person**. Each HR personnel member is represented by a User who also extends from Person.

**Person** has SSN, name, gender (Male or Female), birthdate, email address. **User** is a **Person** who has username and role. **Role** is either **Admin** or **Regular** **User**. **Employee** is a **Person** who has the following additional fields: salary, position and start date. Also, every **Employee** has exactly one **Manager**. **Manager** is an **Employee** who manages other Employees (has a list of employees) and has the following additional fields: bonus and team budget. **Manager** is a composite of employees. Also, **Manager** has an internal iterator that accepts a visitor.

**Company** is a class consisting of one **Manager** (the CEO). **CompanyService** is the interface for interacting with this HR system. To use the system, user needs to call the login() method of **CompanyService** and then other methods can be called. See **Application** for an example usage of this system.

**Question**

List (on the back of this page) all patterns that are used in this application and explain where they are applied in one or two sentences.

**Coding Challenge**

Now assume that we decide to add role-based access control to this system. HR personnel (represented by objects of type User) use a service layer object called **CompanyService** to access different customer services provided by this system (review **CompanyService** to see what types of services we are talking about).

Only users of type “ADMIN” are allowed to access salary or budget information. How can you implement this access control mechanism without changing a single line of code in any of the classes except for **ServiceFactory**? Remember open-closed principle. You are allowed to extend the system, but no changes to any of the existing classes except for **ServiceFactory**.

What if you wish to add logging to this system? Let’s say, we wish to print some information to the console (System.out.println(…)) each time we enter a method in the service layer.

**Extra Credit**

Even if you block access to methods such as **getTotalSalary**() and **getAnnualBudget**(), user can still call getEmployees() and then call those methods directly on Employee or Manager. Using the same method above (minimal impact to code), how can you extend access control to patch this security hole?

Also, only ADMIN role should be able to call the accept() method of Employee or Manager!!! This is a dangerous method! Opens up a huge security hole!!! As one student once called it, it is like a backdoor for a virus!!!

# CS525 Quiz 1

10 points, 10 minutes. Answer all of the questions as clearly and briefly as possible. Show all work and answers directly on this sheet.

1. What is the intent of strategy pattern?
2. Where is VOD used in the strategy pattern?
3. Where is polymorphism used in the strategy pattern? Why – for what benefit?
4. What is the importance (role) of the context object in the strategy pattern?
5. Name three OO principles gained from using the strategy pattern, and how they are achieved

# CS525 Quiz 2

1. What is the main intent(s) of the observer pattern, and how does it accomplish it?
2. Describe the push model for an observer? What are its benefit(s), cost(s)?
3. Describe the pull model for an observer? What are its benefit(s), cost(s)?
4. In the Java observer model, which of the two base types in the observer model is not an interface? Why, and what is the impact of this?
5. What is the issue of coupling in the observer pattern, and how does the patter solve it?

# CS525 Quiz 3

1. What is the main intent(s) of the decorator pattern, and how does it accomplish it?
2. Are decorators static or dynamic (meaning can they add runtime/dynamic behavior)? How?
3. Why do decorators have the same super-types as the objects they are decorating?
4. In decorator pattern, we use inheritance and composition? Explain the role of each one.
5. What is the down side of the decorator pattern?

# CS525 Quiz 4

1. What is the intent of factory method?
2. What is the intent of abstract factory?
3. How does (does?) the factory method pattern use polymorphism?
4. How does the abstract factory pattern use polymorphism?
5. What is the difference in factory method and abstract factory patterns?

# CS525 Quiz 5

1. What the intent of the command pattern?
2. Does every Concrete Command need to compose of a Receiver? Give an example if you can.
3. What is the role of polymorphism in the command pattern?
4. Draw the UML diagram for the command pattern (use the other side of the page)
5. Which class creates the Command objects? Which one executes the command?

# CS525 Quiz 6

1. What the intent of the adapter pattern?
2. What the intent of the façade pattern?
3. Where would you use the adapter pattern?
4. Where would you use the façade pattern?
5. Draw a UML diagram for the adapter pattern?

# CS525 Quiz 7

1. What is the intent of the template method pattern?
2. Where is overriding used in the template pattern, and why (for what benefit)?
3. Where is polymorphism used in the template pattern? Why – for what benefit?
4. Name three SE benefits gained from using the template pattern, and how they are achieved
5. What is the role of refactoring in the template pattern?

# CS525 Quiz 8

1. Compare external and internal iterators;
   1. What are the advantages of an internal iterator (if any)?
   2. What are the possible disadvantages (if any)?
2. What is the intent of the composite pattern? How well does it realize that goal?
3. What was the design issue for deciding if all methods should be in the top level component interface, or specific to child nodes?
4. In what place(s) we use polymorphism in the composite pattern?
5. What is the intent of the Iterator pattern?

# CS525 Quiz 9

1. What is the role of the context class in the state pattern?
2. What are the VOD components in the state pattern (what varies, what stays the same)?
3. The state and strategy patterns have very similar design UML diagrams; how are they different?
4. What is the role of polymorphism in the state pattern?
5. What is the coupling between the concrete states and the context? How is it resolved (handled)?

# CS525 Quiz 10

1. What is the benefit of the proxy pattern?
2. What is the benefit and general structure of a protection proxy?
3. Give three applications of the proxy pattern.
4. What is the role of polymorphism in the proxy pattern?
5. How does a proxy pattern provide abstraction, and over what?

# Question 1 [15 points] [15 minutes]

Below you find a list of design problems that can be solved by applying one of the patterns we used. Give the name of the pattern you would use to solve the given problem. Only write down the name of the pattern.

|  |  |
| --- | --- |
| **Design problem** | **Name of the design pattern that solves this problem** |
| We want to be able to sort a list in different ways like using bubble sort, insertion sort or even your own sorting algorithm |  |
| We want to be able to record and play back all actions we do when we play a game |  |
| We want to store the content of a XML file in memory and then do operations on this XML content like adding a new element, or changing an attribute |  |
| We want to draw certain shapes in a paint application. We first have to select which shape we want to draw (circle, line, etc). Then we click the mouse on a canvas at position 1, followed by moving the mouse to position 2 while holding the mouse down, and then releasing the mouse at position 2. The application should now draw the selected shape using the 2 positions. |  |
| We want to implement the model-view-controller pattern that separates the model (data we want to show) from the view (user interface that shows the actual data). When the model changes, the views have to be updated. You can have as many views as you want. |  |
| We receive different kind of orders from all of our clients. Every order has its own format and we have many clients. So whenever we receive an order, we first have to know who in the client that has sent this order, end then we know how to process it. We expect to add many new clients in the future that will sent their orders in their own format. It should be easy to add many new clients. |  |
| We want to loop over a collection without being dependent on the structure of that collection. |  |
| We want clients of a component to be independent of the internals of that component |  |