Objective - Patterns - Test

Review of attempt 1

Finish review

Started on	Tuesday, 5 November 2013, 01:42 AM
Completed on	Tuesday, 5 November 2013, 02:53 AM
Time taken	1 hour 10 mins
Grade	**18** out of a maximum of 38 (**47**%)
Feedback	FAIL

Show **All** / Correct / In-correct

1. The current application accesses multiple databases for implementing its functionality. You have observed that the business objects are using vendor-specific database API calls for storage & retrieval.

Which of the following design patterns must have been used to avoid this situation?

- a. Transfer Object ✗
- b. Data Access Object ✓
- c. Domain Store ✗
- d. JDBC ✗

JDBC is a technology but not a design pattern. Option B is correct.

Transfer Object - The Transfer Object pattern provides the best techniques and strategies to exchange data across tiers (that is, across system boundaries) to reduce the network overhead by minimizing the number of calls to get data from another tier.

Data Access Object - Data Access Object enables loose coupling between the business and resource tiers. Data Access Object encapsulates all the data access logic to create, retrieve, delete, and update data from a persistent store. Data Access Object uses Transfer Object to send and receive data.

Domain Store - Domain Store provides a powerful mechanism to implement transparent persistence for your object model. It combines and links several other patterns including Data Access Objects.

Refer: http://www.corej2eepatterns.com/Patterns2ndEd/PatternRelationships.htm

Incorrect
Marks for this submission: 0/1.

Feedback to Author

2. An enterprise has multiple J2EE-based Credit card applications. The business is considering to sell some of the existing services such as credit card statements etc. (built for these applications). These services are built as EJBs, POJOs and WebServices.

Which of the following design patterns best fit this situation?

- a. Session Facade ✗
- b. Domain Store ✗
- c. Web Service Broker ✓
- d. Service To Worker. ✗

Option C is correct.

Using Web Service Broker pattern(and suitable products), you can manage(add services, security, configure clients

etc..) all the exposed services of your enterprise.

Web Service Broker - Web Service Broker exposes and brokers one or more services in your application to external clients as a web service using XML and standard web protocols. A Web Service Broker can interact with Application Service and Session Facade. A Web Service Broker uses one or more Service Activators to perform asynchronous processing of a request.

Session Facade - Session Facade provides coarse-grained services to the clients by hiding the complexities of the business service interactions. A Session Facade might invoke several Application Service implementations or Business Objects. A Session Facade can also encapsulate a Value List Handler.

Domain Store - Domain Store provides a powerful mechanism to implement transparent persistence for your object model. It combines and links several other patterns including Data Access Objects.

The Service to Worker pattern, like the Dispatcher View pattern, describes a common combination of other patterns from the catalog. Both of these macro patterns describe the combination of a controller and dispatcher with views and helpers. While describing this common structure, they emphasize related but different usage of patterns.

Both of these patterns differ in division of labour among components(Controller, Dispatcher and View). In Dispatcher View content retrieval is done by View and in case of Service To worker content retrieval is done by controller.

Correct
Marks for this submission: 1/1.

Feedback to Author

3. Julia Fractals Inc. is building a J2EE based application for Order Entry and management of their fractal software. Once the order is taken, it is submitted to a relational database. A provisioning system then queries data and makes appropriate calls to various subsystems using JMS on MQ Series.

What design pattern is JMS an example of here?

- a. Observer ✗
- b. Mediator ✗
- c. Adapter ✗
- d. Bridge ✓
- e. Visitor ✗

Choice D is correct.

Bridge (GOF 151) "Decouple an abstraction from its implementation so that the two can vary independently." In this case, JMS is the abstraction. The implementation could be MQ Series, TIBCO Rendezvous and Vitria Businessware. Hence, choice D is correct.

Observer (GOF 293) "Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically." Hence, choice A is incorrect.

Mediator (GOF 273) "Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and lets you vary their interaction independently." Hence, choice B is incorrect.

Adapter (GOF 139) "Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces." Hence, choice C is incorrect.

Visitor (GOF 331) "Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates." Hence, choice E is incorrect.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

4. What design pattern best explains the use of the stub and the skeleton in CORBA based RPC applications?

- a. Factory Method ✗
- b. Singleton ✗
- c. Strategy ✗
- d. Proxy ✓

Choice D is correct.

Proxy (GOF 207) "Provide a surrogate or placeholder for another object to control access to it." Hence, choice E is incorrect. The applicability section (GOF 208) defines 'remote proxy' as "A remote proxy provides a local representative for an object in a different address space." Using the stub and the skeleton, CORBA based applications provide local representatives for distributed objects. Hence, choice D is correct.

Factory Method (GOF 107)"Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory method lets a class defer instantiation to subclasses." Hence, choice A is incorrect.

Singleton (GOF 127) "Ensure a class only has one instance, and provide a global point of access to it." Hence, choice B is incorrect. Strategy (GOF 315) "Define a family of algorithms, encapsulate each one and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it." Hence, choice C is incorrect.

Decorator (GOF 175) "Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality." Hence, choice E is incorrect.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

5. Which of the statements below describe the Strategy design pattern and its benefits? Select two choices.

- a. The client code will contain if-then-else statements used to select the appropriate algorithm. ✗
- b. It offers an alternative to sub classing. ✓
- c. Easy to add new behavior without the need to edit any code. ✓
- d. Allows subclasses to override parts of an algorithm without the need to rewrite the whole algorithm. ✗
- e. Allows the developer to dynamically add extra functionality to an object. ✗

Choices B and C are the correct answers.

The Strategy design pattern allows you to change behavior without recompiling any client code. Basically you define a method in an abstract class;the variations of this behavior are then defined in other classes that inherit from the abstract class. The client can then use any of these classes without the need to recompile.

Choice A is incorrect because the whole algorithm could be changed without the client being aware; so there would be no conditional statement within the client code.

Choice D describes a benefit of using the Template Method pattern.

Choice E describes a benefit of the Decorator pattern.

Strategy - "Define a family of algorithms, encapsulates each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

6. Which of the following statements describe the Observer pattern and the benefits of using it?

- a. Provides loose coupling between components ✓
- b. Should increase application performance ✗
- c. Based on point-to-point messaging ✗
- d. Allows messages to be queued ✗
- e. An object will appear to change its class when it's internal state changes ✗

Choice A is the correct answer.

The Observer pattern provides a way for an object to broadcast messages (one-to-many).

Observer - "Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically."

Choice C is incorrect because the observer pattern has a one-to-many messaging relationship.

Choice B is incorrect because the Observer pattern will have no affect on the performance of an application.

Choice D describes the Command design pattern.

Choice E describes the State pattern.Command - "Encapsulates a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and supports undoable operations."

Strategy - "Define a family of algorithms, encapsulates each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Correct
Marks for this submission: 1/1.

Feedback to Author

7. Which of the following are benefits of using Data Access Object pattern?

- a. You want to implement parent-child relationships efficiently when implementing business objects as entity beans. ✗
- b. Reduces layers in the application. Lesser coding. ✗
- c. Provides a uniform data access API for a persistent mechanism to various types of data sources, such as RDBMS, LDAP, OODB, XML repositories, flat files ✓
- d. You want to separate persistence from your object model. ✗

Option D is incorrect because it describes Domain Object.

Option A is incorrect because it describes Composite Entity.

Option C correctly describes benefit of DAO pattern.

Option B is incorrect as the DAO introduces a new layer in the application.

Correct
Marks for this submission: 1/1.

Feedback to Author

8. Which of the following are benefits of using the Service Activator pattern?

- a. Improves reusability of business logic ✗
- b. Provides asynchronous processing for any business-tier component ✓
- c. Centralizes reusable business and workflow logic ✗
- d. You want to provide access to one or more services using XML and web protocols. ✗

Options A and C describe benefit of Application Service Design Pattern.

Option D describes Web Service Broker Pattern.

Option B is the benefit of Service Activator Pattern - "Use a Service Activator to receive asynchronous client requests and messages. On receiving a message, the Service Activator locates and invokes the necessary business methods on the business service components to fulfill the request asynchronously."

Correct
Marks for this submission: 1/1.

Feedback to Author

9. In which of the following situations, would you use the Observer pattern? Select two choices.

- [x] a. When you need to have objects notified of events but you don't know which objects would have such needs, or if you will need to add more objects to receive such notification, at a later date. ✓
- [ ] b. You want one object to monitor the state of another object but you don't want the object being monitored to need to send any messages regarding its state. ✗

Choices A and D are correct.

The observer pattern is used to notify dependents of an object when that object changes state.

Choice E is a description of the Mediator pattern.

Choice C is incorrect because the Observer pattern does not reduce the number of instances you need to create.

Choice B is almost correct except the messages are sent when the object being monitored changes state.

Observer - (GOF 293): "Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notifies and updated automatically."

Mediator - (GOF 273): "Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Correct
Marks for this submission: 1/1.

Feedback to Author

10. You need to access a complex object in a recursive way, building the object from other objects. This is an example of which pattern?

- a. Abstract Factory ✗
- b. Factory Method ✗
- c. Builder ✗
- d. Composite ✓
- e. Recursive Builder ✗

Choice D is correct.

Normally, you would assume that you would need to use a creational pattern such as the Builder or Abstract Factory to do this. (Prototype could be applicable as well, if we are not building a family or an aggregate object, but simply prototyping existing objects). However, the important aspect is that you need to recursively build a composite object from other objects. This is an example of the Composite pattern.

There is no such pattern as the Recursive Builder. Therefore, choice E is incorrect.

Composite - (GOF 163): "Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly."

The other patterns were:

- Abstract Factory - (GOF 87): "Provide an interface for creating families of related or dependent objects without specifying their concrete classes."
- Factory Method - (GOF 107): "Define an interface for creating an object, but let subclasses decide, which class to instantiate. Factory Method lets a class defer instantiation to subclasses."
- Builder - (GOF 97): "Separate the construction of a complex object from its representation so that the same construction process can create different representations."

Correct
Marks for this submission: 1/1.

Feedback to Author

11. You are currently designing your own Desktop Publishing application, as you have not found any existing application that does exactly what you want. As part of the design, you are using a Controller to which you send all GUI requests. Not all objects can process the same commands.

For example, you cannot select the spell check tool when an image has the focus. To stop any possible errors, you would like to filter out some of the messages as they are passed from these objects to the Controller object. What pattern could you use?

- a. Firewall ✗
- b. Proxy ✓
- c. Adapter ✗

Choice B is correct.

Firewall and Filter are not design patterns. In this scenario, what you are essentially trying to do is filter all packets that do not meet a certain set of requirements. This behavior is just like a Proxy server dropping packets from certain IP address etc.

Proxy - (GOF 207): "Provide a surrogate or placeholder for another object to control access to it."

The other patterns:

Adapter - (GOF 139):"Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."

Observer - (GOF 293):"Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically."

Chain of Responsibility - (GOF 223):"Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

12. You are designing a complex set of classes that provides a secure framework for other programmers to use. The idea behind this framework is that it will allow other programmers to write secure programs without the usual complexities of writing secure applications.

What sort of design pattern is being used here?

- a. Composite ✗
- b. Facade ✓
- c. Decorator ✗
- d. Adapter ✗
- e. Mediator ✗

Choice B is correct.

Facade - (GOF 185): "Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use."

The other patterns are described below:

- Adapter - (GOF 139): "Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."
- Composite - (GOF 163): "Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly."
- Decorator - (GOF 175): "Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality."
- Mediator - (GOF 273): "Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

13. ABC Company has an existing complaint management system which is used only by its employees through a desktop application which updates the database. Company also has very good messaging infrastructure. Now, it would like to expose this system to worldwide users as a service.

Which of the following design patterns could help solve this requirement? Choose two options.

- a. Session Facade ✗
- b. Service Activator. ✓
- c. Web Service Broker ✓

Options B and C are correct.

The service can be exposed through WebService Broker which will put messages in Queue to be picked by Service Activator. On receiving a message,the Service Activator updates the database. See descriptions of patterns. Service To Worker is a presentation tier pattern and Session Facade is a business tier pattern.

Patterns from integration tier - Service Activator and Web Service Broker best suits above requirements.

Session Facade - Session Facade provides coarse-grained services to the clients by hiding the complexities of the business service interactions. A Session Facade might invoke several Application Service implementations or Business Objects. A Session Facade can also encapsulate a Value List Handler.

Service Activator - Service Activator enables asynchronous processing in your enterprise applications using JMS. A Service Activator can invoke Application Service, Session Facade or Business Objects. You can also use several Service Activators to provide parallel asynchronous processing for long running tasks.

Web Service Broker - Web Service Broker exposes and brokers one or more services in your application to external clients as a web service using XML and standard web protocols. A Web Service Broker can interact with Application Service and Session Facade. A Web Service Broker uses one or more Service Activators to perform asynchronous processing of a request.

The Service to Worker pattern, like the Dispatcher View pattern, describes a common combination of other patterns from the catalog. Both of these macro patterns describe the combination of a controller and dispatcher with views and helpers. While describing this common structure, they emphasize related but different usage of patterns. Both of these patterns differ in division of labour among components(Controller, Dispatcher and View).

In Dispatcher View content retrieval is done by View and in case of Service To worker content retrieval is done by controller.

Correct
Marks for this submission: 1/1.

Feedback to Author

14. ABC Bank has an internet Banking application. This application has been built with very-well defined Services. Bank would like to expose these services through mobile channel. Bank has partnered with XYZ mobile solutions company. It has been decided that when the XYZ company receives request through customer mobile, it makes a web service call to your application.

Which of the following design patterns best fit this situation?

- a. Session Facade ✗
- b. Domain Store ✗
- c. Web Service Broker ✓
- d. Service To Worker. ✗

Option C is correct.

See descriptions of patterns.

Correct
Marks for this submission: 1/1.

Feedback to Author

15. You are creating a web application for an online product ordering system. You have developed an application level object to maintain connection pooling mechanism for ordering system. Your lead instructed that the resources should be efficiently utilized and should provide a global point of access to that instance.

What pattern should you use?

- A. Abstract factory ✗
- B. Factory Method ✗
- C. Builder ✗
- D. Prototype ✗
- E. Singleton ✓

Choice E is correct.

The Singleton pattern ensures that a class has only one instance, and provides global point of access to that instance. It can also be used to create a variable number of instances of a class. The Abstract factory pattern is used for creating many objects that are dependent on each other so choice A is incorrect.

The Factory Method pattern provides an interface for creating an object, it defers creation to either sub classes or helper classes. Hence, choice B is incorrect.

The Builder pattern separates the construction and representation of an object. The client is shielded from the objects construction, only needing to specify its content and type.

Choice C is hence incorrect. The Prototype pattern is used to create new objects by copying its prototype.

Choice D is therefore incorrect.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

16. You have had enough with all the UML tools on the market as none do exactly what you want them to. Therefore, you have decided to design your own. However when designing it you realize that certain parts will be really complicated for example you have a Diagram object that is made up of lots of other objects. This diagram object can be used for a variety of different diagrams including class and sequence diagrams. When you create it you only want to specify its type and content.

What pattern should you use?

- a. Abstract Factory ✗
- b. Factory Method ✗
- c. Builder ✓
- d. Decorator ✗

Choice C is correct.

The builder pattern separates the construction and representation of an object. The client is shielded from the objects construction only needing to specify it's content and type. The Abstract Factory pattern is used for creating many objects that are dependent on each other. Hence, choice A is incorrect.

The Factory Method pattern provides an interface for creating an object that allows either sub classes or helper classes to create that object. Hence, choice B is also incorrect.

The Decorator pattern is not used to build objects. It adds extra functionality to existing objects. Hence, choice D is incorrect.

Correct
Marks for this submission: 1/1.

Feedback to Author

17. Which of the following options describe the Visitor pattern and the benefits of using it? Select two choices.

- a. Easy to add new operations ✓
- b. Simple interface for a complex subsystem ✗
- c. Separates abstraction and implementation ✗
- d. You need to group related operations ✓
- e. Easy to add functionality dynamically ✗

Choices A and D are the correct answers.

As stated in the GOF definition below, the Visitor pattern provides a way to add new operations without changing the class of the element on which it operates. The Visitor pattern should be used when the class defining the object structure rarely changes, but you need to add extra operations. It is a handy pattern when you need to upgrade old applications or when trying to save a design that didn't satisfy all the requirements, it should have.

Choice B describes the Facade pattern.

Choice C describes the Bridge pattern.

Choice E describes the Decorator pattern.

Visitor - "Represents an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates."

Correct
Marks for this submission: 1/1.

Feedback to Author

18. You are working on the design for a desktop publishing application. When the user selects a new tool from a menu, i.e. the paintbrush, you need to inform multiple objects of this new selection. (This is for them to change how they respond to events).

You recognize the problem of sending one message to multiple objects, but is there a GOF design pattern to handle this situation? If so, what is it called?

- a. Chain of Responsibility ✗
- b. Command ✗
- c. Mediator ✗
- d. Observer ✓
- e. Proxy ✗
- f. Strategy ✗

Choice D is the correct answer.

The Observer pattern provides a way for an object to broadcast messages (one-to-many).

Observer - "Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically."

Chain of Responsibility - "Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."

Command - "Encapsulates a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and supports undoable operations."

Mediator - "Defines an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Proxy - "Provides a surrogate or placeholder for another object to control access to it."

Strategy - "Defines a family of algorithms, encapsulates each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

19. Which of the following are benefits of using Application Service pattern? Select two choices.

- a. Improves reusability of business logic ✓
- b. Provides asynchronous processing for any business-tier component ✗
- c. Centralizes reusable business and workflow logic ✓
- d. You want to provide access to one or more services using XML and web protocols. ✗

Options A and C describe the benefit of Application Service Design Pattern. So, Options A and C are correct.

Option D describes WebService Broker Pattern.

Option B is the benefit of Service Activator Pattern.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

20. Which of the following are benefits of using Composite Entity pattern?

- a. You want to implement parent-child relationships efficiently when implementing business objects as entity beans. ✓
- b. Improves Security. ✗
- c. Provides a uniform data access API for a persistent mechanism to various types of data sources, such as RDBMS, LDAP, OODB, XML repositories, flat files ✗
- d. You want to separate persistence from your object model. ✗

Option D is incorrect because it describes Domain Object.

Option A is correct because it describes Composite Entity.

Option C is incorrect because it describes benefit of DAO pattern.

Option B is incorrect as the Composite Entity pattern is not related security.

Correct
Marks for this submission: 1/1.

Feedback to Author

21. What is the difference between the abstract factory pattern and the factory method pattern?

- a. The factory method makes objects that should be used together. This is not the case for the abstract factory. ✗
- b. The abstract factory pattern provides an interface for creating a family of objects whereas factory method provides an interface for creating one object. ✓
- c. In the abstract factory pattern, the objects that the factory makes are to be used together. This is not necessarily true in the factory method pattern. ✗
- d. The factory method pattern is used when the class does not know the class of the object it must create. But in the abstract factory this is known in advance. ✗
- e. The factory method and abstract factory are essentially the same pattern but two different names are used to describe them depending on the circumstances when they are implemented. ✗

Choice B is correct.

Both the Abstract Factory and Factory Method are Creational patterns.

Abstract Factory - (GOF 87): "Provide an interface for creating families of related or dependent objects without specifying their concrete classes."

Factory Method - (GOF 107): "Define an interface for creating an object, but let subclasses decide, which class to instantiate. Factory Method lets a class defer instantiation to subclasses."

Choice A offers a description of the abstract factory pattern not the factory method.

The descriptions in choices C and D are the wrong way round.

Choice E is incorrect, as the two patterns are different.

Correct
Marks for this submission: 1/1.

Feedback to Author

22. You are designing a paint application and as part of the user interface, you have a toolbar along the left-hand side of the screen. Each of the icons on the toolbar has different actions when you are using different tools. The way you have structured it, the application is required to pass commands from one object to another. When the appropriate object receives the command, it handles the request.

This is an example of which pattern?

- a. Command ✗

Choice B is correct.

Chain of Responsibility - (GOF 223):"Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."

Below are descriptions of the other patterns:

Command - (GOF 233):"Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations"

Interpreter - (GOF 243):"Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language."

Adapter - (GOF 139):"Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."

Strategy - (GOF 315):"Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

23. You have had enough of you existing IDE (Integrated Development Environment) and have started work on designing your own. Now you are working on the undo part of the application and you need to record an object internal state without violating encapsulation and reclaim it later without knowledge of the original object.

What pattern would you use to do this?

- A. Memento ✓
- B. State ✗
- C. Mediator ✗
- D. Rollback ✗
- E. Transaction ✗
- F. ACID ✗

Choice A is correct.

Memento - (GOF 283):"Without violating encapsulation, Capture and externalize an object's internal state so that the object can be restored to this state later."

Choices D, E and F are not names of patterns. Rollback is something you would do if a transaction were terminated halfway through. Transaction is self-explanatory. ACID stands for Atomic, Consistent, Isolatable and Durable. All transactions must adhere to this.

Below are descriptions of the other patterns:

State - (GOF 305):"Allow an object to alter its behavior when it's internal state changes. The object will appear to change its class."

Mediator - (GOF 273):"Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

24. One of the advantages of using stateless session beans is that they are lightweight objects and do not have conversational state overheads. Further, the container swaps these bean instances in and out of the bean pool to appropriately manage resources. This allows the container to use fewer instances of the bean to service a larger number of clients.

What design pattern is being illustrated here?

Choice D is correct.

- a. Decorator ✗
- b. Factory ✗
- c. Facade ✗
- d. Flyweight ✓
- e. Visitor ✗

Flyweight (GOF 195) "Use sharing to support large numbers of fine-grained object efficiently." Here the container uses fewer instances of Stateless Session Beans to service a larger number of clients. Hence, choice D is correct.

Decorator (GOF 175) "Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality." Hence, choice A is incorrect.

Factory Method (GOF 107) " Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory method lets a class defer instantiation to subclasses." Hence, choice B is incorrect. Facade (GOF 185)"Provide a unified interface to a set of interfaces in a subsystem." Hence, choice C is incorrect.

Visitor (GOF 331) "Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates." Hence, choice E is incorrect.

Correct
Marks for this submission: 1/1.

Feedback to Author

---

25. Which of the following is not a Integration-Tier pattern?

- a. Composite Entity ✓
- b. Service Activator ✗
- c. Domain Store ✗
- d. Data Access Object ✗

Option A is correct.

Presentation Tier Patterns :- Intercepting Filter, Context Object, Front Controller, Application Controller, View Helper, Composite View, Dispatcher View, Service To Worker

Business Tier Patterns:- Business Delegate, Service Locator, Session Facade, Application Service, Business Object, Composite Entity, Transfer Object, Transfer Object Assembler, Value List Handler

Integration Tier Patterns:- Data Access Object, Service Activator, Domain Store, Web Service Broker

Correct
Marks for this submission: 1/1.

Feedback to Author

---

26. The current application has been built using JSF & a custom persistence framework. You have been approached to expose some of the data as a EJB to another J2EE application. You may need to access multiple business objects to provide the data.

Which of the following design patterns best fits the situation?

- a. Application service ✗
- b. Session Facade ✓
- c. Service To worker ✗
- d. Business Delegate ✗

Option B is correct.

See description of patterns.
Application Service - Application Service centralizes and aggregates behavior to provide a uniform service layer to the business tier services. An Application Service might interact with other services or Business Objects. An Application Service can invoke other Application Services and thus create a layer of services in your application.

Session Facade - Session Facade provides coarse-grained services to the clients by hiding the complexities of the business service interactions. A Session Facade might invoke several Application Service implementations or Business Objects. A Session Facade can also encapsulate a Value List Handler.

The Service to Worker pattern, like the Dispatcher View pattern, describes a common combination of other patterns from the catalog. Both of these macro patterns describe the combination of a controller and dispatcher with views and helpers. While describing this common structure, they emphasize related but different usage of patterns. Both of these patterns differ in division of labour among components(Controller, Dispatcher and View).

In Dispatcher View content retrieval is done by View and in case of Service To worker content retrieval is done by controller.

Business Delegate - Business Delegate reduces coupling between remote tiers and provides an entry point for accessing remote services in the business tier. A Business Delegate might also cache data as necessary to improve performance. A Business Delegate encapsulates a Session Facade and maintains a one-to-one relationship with that Session Facade. An Application Service uses a Business Delegate to invoke a Session Facade.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

27. You need to interface with an existing application. You have full access to the source code and UML diagrams from the existing application. Part of the requirements implies that you will need to connect unrelated objects together. You want to know whether the Bridge pattern or the Adapter pattern will be suitable.

Which of the following are true about the Bridge and Adapter patterns? Select two choices.

- a. The Adapter pattern implements an interface known to its clients and provides an instance of a class not known to its clients. ✓
- b. The Bridge pattern implements an interface known to its clients and provides an instance of a class not known to its clients. ✗
- c. The Adapter pattern creates a separation between abstractions and classes that implement those abstractions. ✗
- d. The Bridge pattern creates a separation between abstractions and classes that implement those abstractions. ✓

Choices A and D are correct.

As the answers state the Adapter pattern implements an interface known to its clients and provides an instance of a class not known to its clients. The Bridge pattern creates a separation between abstractions and classes that implement those abstractions.

Choices B and C are incorrect because the descriptions are the other way around.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

28. In a project, low on performance, you have analyzed that a lot of time is spent on accessing data using fine grained remote method calls. After several method calls the required data is assembled into object and sent out for display. Which design pattern can be used to improve this performance limitation?

- a. Service Locater ✗
- b. Transfer Object ✓
- c. Singleton ✗
- d. Business Delegate ✗

Choise B is correct.

Choice B is correct because Transfer Object lets programmer assemble all data remotely and send it back to front layers. It reduces number of remote calls.

Choice A is incorrect because Service Locater pattern is used to centralize resource lookup code.

Choice C is incorrect because Singleton pattern is used to control number of instances not method calls.

Choice D is incorrect because Business Delegate pattern is used to encapsulate remote access code and remote exceptions behind itself.

Further Reference: Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition) Authored By Deepak Alur, Dan Malks and John Crupi (Publisher: Prentice Hall)

Correct
Marks for this submission: 1/1.

Feedback to Author

29. Which of the following statements describe the Flyweight pattern and the benefits of using it?

- a. The Flyweight pattern should be used with application that has a low number of objects and increased performance is desired. ✗
- b. The Flyweight pattern should be used when you need to control access to an object. ✗
- c. You need to coordinate access for a group of objects. ✗
- d. May reduce the amount of memory an application uses. ✓
- e. The Flyweight pattern should be used when an application is dependent upon object identity." ✗

Choice D is the correct answer.

The Flyweight pattern will help increase the performance of the application by sharing objects as opposed to creating unique objects for each task. Flyweight - "Uses sharing to support large numbers of fine-grained objects efficiently".

Choice B describes the Proxy pattern.

Choices A and E are incorrect because an application needs to have a high number of objects that are not dependent upon object identity for the Flyweight pattern to be of any benefit.

Choice C describes the Mediator pattern.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

30. Which of the following choices describe the Template Method pattern and the benefits of using it? Select two choices.

- a. Promotes code reuse ✓
- b. An alternative to sub classing ✗
- c. Coordinates communication between groups of objects ✗
- d. Easy to add new operations to existing objects ✗
- e. Helps reduce code volume when dealing with localized versions of applications ✓

Choices A and E are the correct answers.

The Template Method pattern allows you to modify an algorithm without rewriting all of the algorithm code. This works by moving certain parts of the algorithm into separate methods and implementing these in sub classes.

Choice B is incorrect because the Template Method pattern uses subclasses to implement specific behavior (the Strategy pattern is an alternative to sub classing).

Choice C describes the Mediator pattern.

Choice D describes the Visitor pattern.

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

Mediator - "Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Visitor - "Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes the elements on which it operates."

Correct
Marks for this submission: 1/1.

Feedback to Author

31.

Which of the following is a benefit of the Application Controller pattern?

- a. You want to intercept and manipulate a request and a response before and after the request is processed. ✗
- b. You want to apply common logic to multiple requests ✗
- c. You want to centralize controlled access points into your system ✗
- d. You want to centralize and modularize action and view management. ✓

Option A describes the benefit of Intercepting Filter Pattern.

Options B and C describes benefits of the FrontController Pattern.

Option D describes the benefit of the Application Controller pattern. So Option D is correct.

Incorrect
Marks for this submission: 0/1.

Feedback to Author

32. Which of the following are benefits of using the Composite pattern? Select two choices.

- a. Decouples an abstraction and implementation ✗
- b. Minimizes the complexity of an object that consists of many different objects ✓
- c. Convert the interface of a class into that of another ✗
- d. Can easily add new types of component ✓
- e. Dynamically attach additional responsibilities to an object ✗

Choices B and D are the correct answers.

The Composite pattern allows you to easily add new types of component and it minimizes the complexity of an object that consists of many different objects.

Choice A describes the Bridge pattern.

Choice C describes the Adapter pattern.

Choice E describes the Decorator pattern.

Composite - "Composes objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

33. You are working on application that will become the next generation digital multimedia service. The logic for playing each type of multimedia file is contained with a class. The classes that form this logic are ordered in two hierarchies.

Firstly, two abstractions exist - one for video files and one for audio files. The second abstraction hierarchy is for playing logic such as MP3, MPEG, and AVI etc. (For each of these hierarchies there are separate implementation classes.)As we have seen in recent years new multimedia file types are always being developed, the application should be able to handle these new types without a need to recompile. You propose to develop the application WITHOUT combining abstractions and implementations (i.e. no distinct classes). This will allow you to extend abstractions without affecting existing implementations.

Is there a GOF design pattern to handle this problem and if so, what is it called?

- a. Adapter ✗
- b. Builder ✗
- c. Bridge ✓
- d. Composite ✗
- e. Decorator ✗

Choice C is the correct answer.

The requirements that the scenario describes are: the ability to extend abstractions without affecting existing implementations (maintain a separation between abstraction and implementation). The Bridge pattern meets this requirement.

Bridge - "Decouples an abstraction from its implementation so that the two can vary independently."

Adapter - "Converts the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."

Builder - "Separates the construction of a complex object from its representation so that the same construction process can create different representations."

Composite - "Composes objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly."

Decorator - "Attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

34. You are involved in the design of an email application. You have recognized a problem and know using a GOF pattern can solve it. The problem is described below, what is the name of the pattern that will solve this?

The construction of an email object can be extremely complex, setting email type, embedding multimedia objects etc. Therefore, you would like to specify the email object type and content and have the email object constructed automatically.

- a. Abstract Factory ✗
- b. Builder ✓
- c. Factory Method ✗
- d. Prototype ✗
- e. Singleton ✗

Choice B is the correct answer.

The builder pattern requires the client to only specify content and type and shields the client from the details of an objects construction.

Builder - "Separates the construction of a complex object from its representation so that the same construction process can create different representations."

Abstract Factory - "Provides an interface for creating families of related or dependent objects without specifying their concrete classes."

Factory Method - "Defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses."

Prototype - "Specifies the kinds of objects to create using a prototypical instance, and creates new objects by copying this prototype."

Singleton - "Ensures a class only has one instance, and provides a global point of access to it."

Correct
Marks for this submission: 1/1.

Feedback to Author

35. You need to simplify the communication between objects. You propose to introduce a single object that will handle the message distribution. Is there a GOF design pattern to handle this situation and if so, what is it called?

- a. Command ✗
- b. Mediator ✓
- c. Observer ✗
- d. Proxy ✗
- e. Strategy ✗

Choice B is the correct answer.

The Mediator pattern controls how a set of objects interact (The objects refer to each other through one object).

Mediator - "Defines an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."

Command - "Encapsulates a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and supports undoable operations."

Observer - "Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically."

Proxy - "Provides a surrogate or placeholder for another object to control access to it."

Strategy - "Defines a family of algorithms, encapsulates each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Correct
Marks for this submission: 1/1.

Feedback to Author

36. You are working on a presentation application that will become the next Microsoft PowerPoint. Consider the following part of the design:When the user selects a slide an empty object is created. When movies are dragged onto the slide movie playing functionality is added to the slide object dynamically. This design will allow for further expansion.

Is there a GOF design pattern to handle this problem and if so what is it called?

- a. Adapter ✗
- b. Builder ✗
- c. Decorator ✓
- d. State ✗
- e. Strategy ✗
- f. There is no such GOF design pattern to handle this ✗

Choice C is the correct answer.

The key to this question is that additional functionality needs to be added to objects dynamically, and the Decorator allows this (please see the GOF definition below).

The definition of the Decorator pattern in the GOF is: "Attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality."

Adapter - "Converts the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces."

Builder - "Separates the construction of a complex object from its representation so that the same construction process can create different representations."

State - "Allows an object to alter its behavior when its internal state changes. The object will appear to change its class."

Strategy - "Defines a family of algorithms, encapsulates each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

37. You are working on a multimedia application. When the user selects new from the pull-down menu, they are presented with the choice to create a variety of different types of file (objects). The class that handles the new requests will not be able to anticipate the class type of the objects it must create.

Is there a GOF design pattern to handle these requirements and if so, what is it called?

- a. Abstract Factory ✗
- b. Builder ✗
- c. Factory Method ✓
- d. Prototype ✗
- e. Singleton ✗
- f. There is no GOF design pattern that is suitable for use in the above scenario ✗

Choice C is the correct answer.

The Factory Method pattern allows sub classes to decide which class to instantiate. The scenario says that the class that processes the 'new' requests will not know in advance the type of class it must instantiate. The Factory Method pattern fits this.

Factory Method - "Defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses."

Abstract Factory - "Provides an interface for creating families of related or dependent objects without specifying their concrete classes."

Builder - "Separates the construction of a complex object from its representation so that the same construction process can create different representations."

Prototype - "Specifies the kinds of objects to create using a prototypical instance, and creates new objects by copying this prototype."

Singleton - "Ensures a class only has one instance, and provides a global point of access to it."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

38. You are designing the next generation multimedia player. As part of the design, you need to ensure that only two instances of the 'Speaker' class are ever created (the speaker class is used to control the volume, bass, and tone). Is there a GOF design pattern to handle this situation and if so, what is it called?

- a. Abstract Factory ✗
- b. Builder ✗
- c. Factory Method ✗
- d. Prototype ✗
- e. Singleton ✓
- f. There is no GOF design pattern that is suitable for use in the above scenario ✗

Choice E is the correct answer.

The Singleton pattern can ensure either only one instance of a class is ever created or that a fixed variable number of instances are created. In code, the constructor is made private and static getInstance() methods are used.

Singleton -"Ensures a class only has one instance, and provides a global point of access to it."

Abstract Factory - "Provides an interface for creating families of related or dependent objects without specifying their concrete classes."

Builder - "Separates the construction of a complex object from its representation so that the same construction process can create different representations."

Factory Method - "Defines an interface for creating an object, but lets subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses."

Prototype - "Specifies the kinds of objects to create using a prototypical instance, and creates new objects by copying this prototype."

Incorrect
Marks for this submission: 0/1.

Feedback to Author

Finish review