Describe about pattern name

It is a handle we can use to illustrate a design problem, its solutions, and
consequences in a word or two.
Naming a pattern instantly increases our design vocabulary. It lets us design at a
higher level of abstraction.
It makes it easier to think about designs and to communicate them and their tradeoffs
to others.
Finding good names has been one of the hardest parts of developing our catalog

What do you mean by conseguences?

They are the results and trade-offs of applying the pattern.
Though consequences are often understood when we describe design decisions,
they are critical for evaluating design alternatives and for understanding the costs

and benefits of applying the pattern.

The consequences for software often concern space and time trade-offs.

They may address language and implementation issues as well.

Since reuse is often a factor in object-oriented design, the consequences of a

pattern include its impact on a system's flexibility, extensibility, or portability.

Listing these consequences explicitly helps you understand and evaluate them.

Point of view affects one's interpretation of what is and isn't a pattern. One person's

pattern can be another person's primitive building block.

The design patterns are descriptions of communicating objects and classes that are

How a Design pattern solves the design problem? Illustrate with an example.

1) Consider how design patterns solve design problems.
Discussions about how design patterns help you find appropriate objects, determine
object granularity, specify object interfaces, and several other ways in which design patterns
solve design problems. Referring to these discussions can help guide your search for the
right pattern.
2) Scan Intent sections.
Read through each pattern's intent to find one or more that are relevant to your problem.
You can use the classification scheme to reduce your search.
3) Study how patterns interrelate.
Studying the relationships between design patterns can help you to select right pattern or
group of patterns.
4) Study patterns of like purpose.
Each pattern concludes with a section that compares and contrasts with other patterns.

These sections help you to identify the similarities and differences between patterns of like purpose.

5) Examine a cause of redesign.

Study the causes of redesign to see if your problem involves one or more of them.

Then look at the patterns that help you avoid the causes of redesign.

6) Consider what should be variable in your design.

This approach is the opposite of focusing on the causes of redesign. Instead of considering what might force a change to a design, consider what you want to be able to change without redesign. The focus here is on encapsulating the concept that varies, a theme of many design patterns.

2(a) What is Gang of Four GOF?

2(b)How to select a design pattern?

1) Consider how design patterns solve design problems.
Discussions about how design patterns help you find appropriate objects, determine
object granularity, specify object interfaces, and several other ways in which design patterns
solve design problems. Referring to these discussions can help guide your search for the
right pattern.

2) Scan Intent sections.
Read through each pattern's intent to find one or more that are relevant to your problem.
You can use the classification scheme to reduce your search.

3) Study how patterns interrelate.
Studying the relationships between design patterns can help you to select right pattern or
group of patterns.

4) Study patterns of like purpose.

Each pattern concludes with a section that compares and contrasts with other patterns.
These sections help you to identify the similarities and differences between patterns of like
purpose.
5) Examine a cause of redesign.
Study the causes of redesign to see if your problem involves one or more of them.
Then look at the patterns that help you avoid the causes of redesign.
6) Consider what should be variable in your design.
This approach is the opposite of focusing on the causes of redesign. Instead of
considering what might force a change to a design, consider what you want to be able to
change without redesign. The focus here is on encapsulating the concept that varies, a theme
of many design patterns.

2(c) How to use design patterns? Explain in detail.

Once you've picked a design pattern, how do you use it? Here's a step-by-step approach to applying a design pattern effectively:

1. Read the pattern once through for an overview. Pay particular attention to the Applicability and Consequences sections to ensure the pattern is right for your problem.

2. Study the Structure, Participants, and Collaborations sections. Make sure you understand the classes and objects in the pattern and how they relate to one another.

3. Look at the Sample Code section to see a concrete example of the pattern in code. Studying the code helps you learn how to implement the pattern.

4. Choose names for pattern participants that are meaningful in the application context. The names for participants in design patterns are usually too abstract to appear directly in an

application. Nevertheless, it's useful to incorporate the participant name into the name
that appears in the application. That helps make the pattern more explicit in the
implementation.

5. Define the classes. Declare their interfaces, establish their inheritance relationships, and
define the instance variables that represent data and object references. Identify existing
classes in your application that the pattern will affect, and modify them accordingly.

6. Define application-specific names for operations in the pattern. Here again, the names
generally depend on the application. Use the responsibilities and collaborations
associated with each operation as a guide. Also, be consistent in your naming
conventions. For example, you might use the "Create-" prefix consistently to denote a
factory method.

7. Implement the operations to carry out the responsibilities and collaborations in the

pattern. The Implementation section offers hints to guide you in the implementation.
The examples in the Sample Code section can help as well.

2(d) Describe the consistent format for describing the design patterns.

3(a) What is design pattern?

Christopher Alexander says, "Each pattern describes a problem which occurs over and
over again in our environment, and then describes the core of the solution to that problem, in
such a way that you can use this solution a million times over, without ever doing it the same
way twice".

3(b) Explain design patterns in Smalltalk MVC.

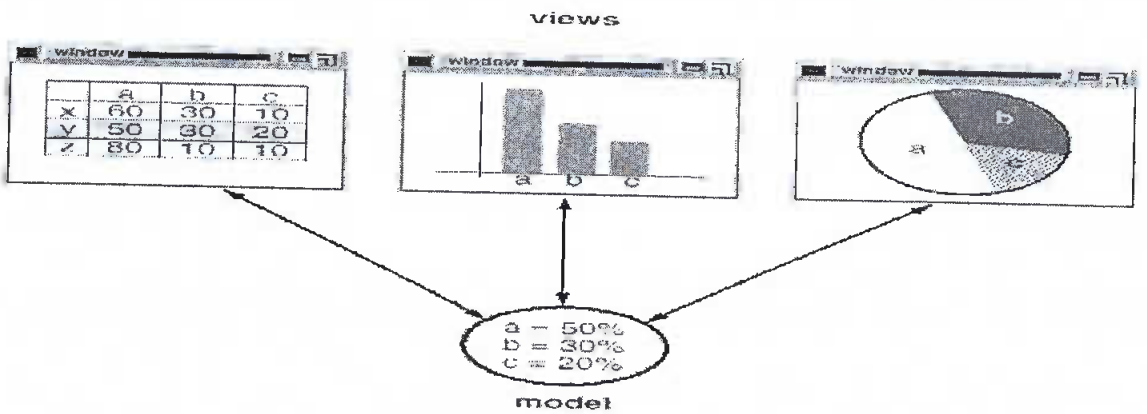
The Model/View/Controller (MVC) triad of classes is used to build user interfaces in
Smalltalk-80. MVC consists of three kinds of objects.
The Model is the application object,
the View is its screen presentation, and the Controller defines the way the user interface
reacts to user input. Before MVC, user interface designs tended to combine these objects
together. MVC decouples them to increase flexibility and reuse.
MVC decouples views and models by establishing a subscribe/notify protocol
between them.
A view must ensure that its appearance reflects the state of the model.
Whenever the model's data changes, the model notifies views that depend on it.

In response, each view gets an opportunity to update itself.

This approach lets you attach multiple views to a model to provide different
presentations.

We can also create new views for a model without rewriting it.

The following diagram shows a model and three views.



The model contains some data values, and the views defining a spreadsheet,
histogram, and pie chart display these data in various ways.

The model communicates with its views when its values change, and the views
communicate with the model to access these values.
This example reflects a design that decouples views from models. But the design is
applicable to a more general problem: decoupling objects so that changes to one can affect
any number of others without requiring the changed object to know details of the others. This
more general design is described by the Observer design pattern.
Another feature of MVC is that views can be nested. For example, a control panel of buttons
might be implemented as a complex view containing nested button views. MVC supports
nested views with the CompositeView class, a subclass of View.
But the design is applicable to a more general problem, which occurs whenever we want to

group objects and treat thegroup like an individual object. This more general design is described by the Composite design pattern.

MVC also lets you change the way a view responds to user input without changing its visual presentation. You might want to change the way it responds to the keyboard, for example, or have it use a pop-up menu instead of command keys. MVC encapsulates the response mechanism in a Controller object.

A view uses an instance of a Controller subclass to implement a particular response strategy; to implement a different strategy, simply replace the instance with a different kind of controller.

The View-Controller relationship is an example of the Strategy design pattern. A Strategy is an object that represents an algorithm.

MVC uses other design patterns, such as Factory Method to specify the default controller

class for a view and Decorator to add scrolling to a view. But the main relationships in MVC are given by the Observer, Composite, and Strategy design patterns.

3(c) What are the different ways in which patterns and frameworks share similarities and in which they differ? Discuss.

3(d) Explain about selection of a design pattern.

With more than 20 design patterns in the catalog to choose from, it might be hard to find the one that addresses a particular design problem, especially if the catalog is new and unfamiliar to you. Here are several different approaches to finding the design pattern that's right for your problem:

1) Consider how design patterns solve design problems.

Discussions about how design patterns help you find appropriate objects, determine
object granularity, specify object interfaces, and several other ways in which design patterns
solve design problems. Referring to these discussions can help guide your search for the
right pattern.

2) Scan Intent sections.

Read through each pattern's intent to find one or more that are relevant to your problem.
You can use the classification scheme to reduce your search.

3) Study how patterns interrelate.

Studying the relationships between design patterns can help you to select right pattern or
group of patterns.

4) Study patterns of like purpose.

Each pattern concludes with a section that compares and contrasts with other patterns.

These sections help you to identify the similarities and differences between patterns of like purpose.

5) Examine a cause of redesign.

Study the causes of redesign to see if your problem involves one or more of them.

Then look at the patterns that help you avoid the causes of redesign.

6) Consider what should be variable in your design.

This approach is the opposite of focusing on the causes of redesign. Instead of considering what might force a change to a design, consider what you want to be able to change without redesign. The focus here is on encapsulating the concept that varies, a theme of many design patterns.

4. a) List the four elements of design patterns. (2M)

Ans: 1. Pattern's Name 2. Problem
3. Solution 4. Consequences

b) Distinguish a class and an Object. (3M)

Ans: An Object's implementation defined by its class. The object is said to be an instance of the class. Objects are created by instantiating a class.

A class specifies the object's internal data and representation and defines the operations the object can perform.

c) Give the Step-by-step approach to apply a design pattern effectively. (5M)

Ans:

1. Read the pattern once through for an overview
2. Get back and study the structure, participants and collaboration sections.
3. Look at the sample code section to see a concrete example of the pattern in code
4. choose Names for pattern participants that are meaningful in application content.
5. Define the classes.
6. Define application specific Names for operations in the pattern

7. Implement the operation to carry out the responsibilities and collaborations in the pattern.

d) What is basis for classifying design patterns? categorize and tabulate the design patterns. (5m)

Ans: Design Pattern are classified by two criteria

1. Purpose 2. Scope.

Purpose: It reflects what a pattern does. Pattern can have either creational, structural or behavioral purpose.

creational patterns concern purpose of object creation.

structural patterns deal with the composition of classes or object.

Behavioral patterns characterise the way in which classes or objects interact and distribute.

Scope: Specifies whether the pattern applies primarly to classes or to object.

class patterns deal with relationship between classes and their subclasses. These relationship established through inheritance, so they are static and fixed at compile time.

Object patterns deal with object relationship, which can be changed at runtime and are most dynamic.

		Purpose		
		Creational	structural	Behavioral
Scope	class	Factory method	Adaptor (class)	interpreter Template method
	object	Abstract factory Builder Prototype Singleton	Adaptor (object) Bridge Composite Decorator Facade Flyweight Proxy	chain of responsibility command iterator mediator Memento Observer State strategy Visitor

5. a) State the meaning of solution. (2M)

Ans: Solution describes elements for the design that includes relationships, responsibilities and collaborations.

b) Define the two categories on which design patterns depends. (3M)

Ans: Design patterns depend on purpose and scope of design patterns.

Purpose means what does design pattern do. Pattern can have creational, structural or behavioral purpose.

Scope specifies whether pattern primarily applies to classes or object.

c) What are some common causes of redesign? (5M)

Ans: Common causes of Redesign:

1. Creating an object by specifying a class explicitly.
2. Dependance on specific operations.
3. Dependance on hardware and software platform
4. Dependance on object representations and Implementations.
5. Algorithmic Dependancies
6. Tight coupling
7. Expanding functionality by subclasses.
8. Inability to alter - classes conveniently.

d) Draw a diagram to describe design pattern relationships. (5M)

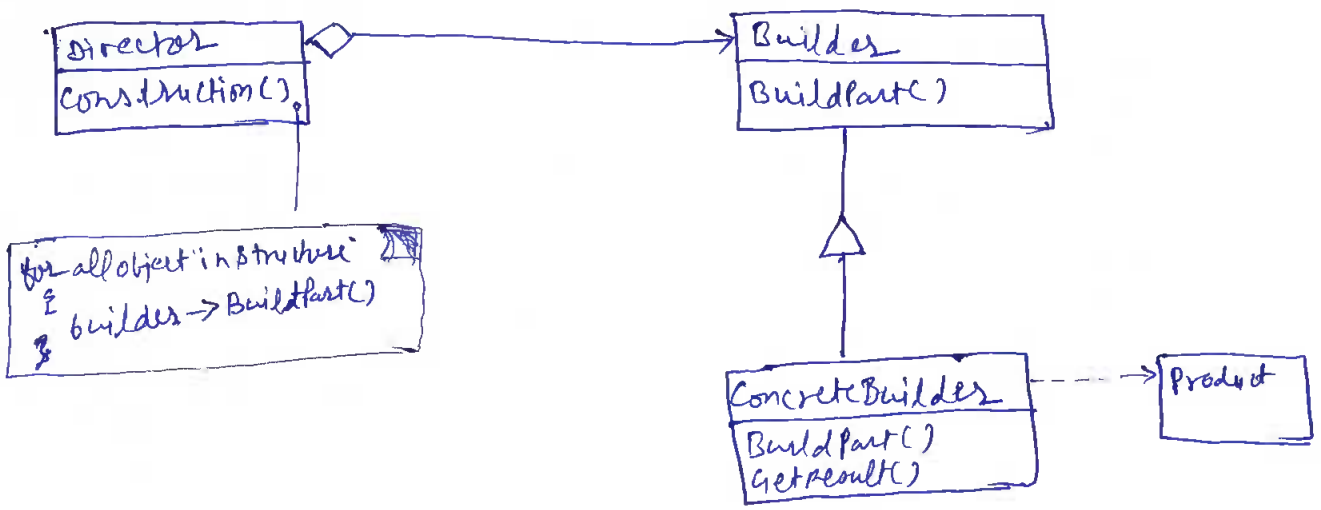
Ans: Another way to organize design patterns is according to how they reference each other in their "Related Patterns".

6 a) What is the meaning of Intent. (2M)

Ans: Intent section gives abstract description of design pattern, it include:
what does the design pattern do?
what particular design issues or problem does it address.

b) Give an Example of class structure. (3M)

Ans: Any class diagram of pattern is accepted.



Class Structure in Builder Design Pattern

c) Discuss about toolkits and frameworks.

Ans:

Toolkits:

An Application will incorporate classes from one or more ~~object~~ libraries of predefined classes called Toolkits. A toolkit is a set of related and reusable classes designed to provide useful and general purpose functionality. An example of toolkit is a set of collection classes for lists, associative tables, stacks and like. Toolkit emphasize on code reuse.

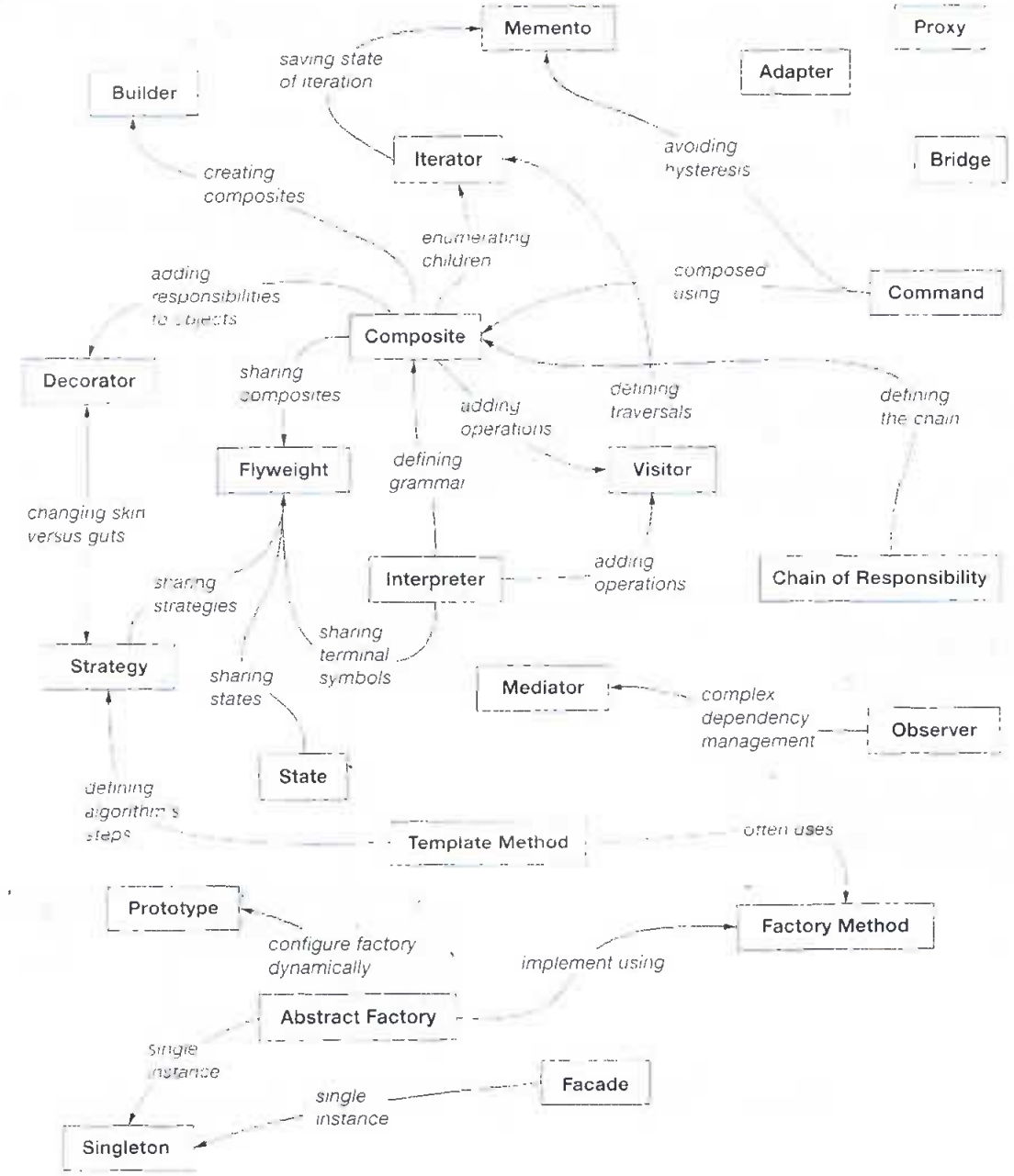


Figure 1.1: Design pattern relationships

Frameworks:

A framework is a set of cooperating classes that make up a reusable design for specific class of software. Framework can help to build compilers for different programming languages & target machines. Frameworks emphasize design reuse over code reuse.

d) compare runtime and compile time structure.

Ans: An object-oriented program's run-time structure often bear little resemblance to its code structure.

Aggregation:

- menifested at runtime
- one object own (having) or is responsible for another object (Every part)

Acquaintance:

- menifested at compilite time
- An object merely knows about another object.
- A weaker relationship than aggregation

A run-time structures are not clear from the code until you understand the pattern.