Structural design patterns

Structural design patterns are concerned with how classes and objects can be composed, to form larger structures.

The structural design patterns simplifies the structure by identifying the relationships.

These patterns focus on, how the classes inherit from each other and how they are composed from other classes.

1.ADAPTER PATTREN:

An Adapter Pattern says that just "converts the interface of a class into another interface that a client wants".

In other words, to provide the interface according to client requirement while using the services of a class with a different interface.

The Adapter Pattern is also known as Wrapper.

Advantage of Adapter Pattern

It allows two or more previously incompatible objects to interact.

It allows reusability of existing functionality.

EXAMPLE:

* Target Interface: This is the desired interface class which will be used by the clients.
* Adapter class: This class is a wrapper class which implements the desired target interface and modifies the specific request available from the Adaptee class.
* Adaptee class: This is the class which is used by the Adapter class to reuse the existing functionality and modify them for desired use.
* Client: This class will interact with the Adapter class.

.................................................

Step 1

Create a CreditCard interface (Target interface).

Step 2

Create a BankDetails class (Adaptee class).

Step 3

Create a BankCustomer class (Adapter class).

Step 4

Create a AdapterPatternDemo class (client class).

...........................................................

BRIDGE PATTREN:

A Bridge Pattern says that just "decouple the functional abstraction from the implementation so that the two can vary independently".

The Bridge Pattern is also known as Handle or Body.

Advantage of Bridge Pattern:

It enables the separation of implementation from the interface.

It improves the extensibility.

It allows the hiding of implementation details from the client.

........................................................................



Step 1

Create a Question interface that provides the navigation from one question to another or vice-versa.

Step 2

Create a JavaQuestions implementation class that will implement Question interface.

Step 3

Create a QuestionManager class that will use Question interface which will act as a bridge..

Step 4

Create a QuestionFormat class that will extend the QuestionManager class

Step 5

Create a BridgePatternDemo class.