Design Patterns

-Yun Cho

3 different categories of Design Patterns

Creational - concerned with the way in which objects are created

Structural - simplifying the design of large object structures by intensifying relationships between them

Behavioral - focus on how objects interact with each other or how the user interacts with them

Creational Patterns

- Singleton
- Builder
- Factory

Singleton Pattern

- A class that has only one instance and provides a global point of access. Also it ensures that only single instance should be created and single object can be used by all other classes

- What is interesting about it?
  - It saves memory because it only creates one instance then it is reused over and over again.

- Usage?
  - Mainly used for database and multi-threaded applications, such as logging, configuration settings

Builder Pattern

- Builds a complex object from simpler objects using step-by-step approach.

- What is interesting about it?
  - Using this pattern, it creates a clear separation between the representation of the object and the construction of it.

- Usage?
  - Main usage is when objects cannot be created in a single step.

Design Patterns

-Yun Cho

3 different categories of Design Patterns

Creational - concerned with the way in which objects are created

Structural - simplifying the design of large object structures by intensifying relationships between them

Behavioral - focus on how objects interact with each other or how the user interacts with them

Creational Patterns

- Singleton
- Builder
- Factory

Singleton Pattern

- A class that has only one instance and provides a global point of access. Also it ensures that only single instance should be created and single object can be used by all other classes

- What is interesting about it?
  - It saves memory because it only creates one instance then it is reused over and over again.

- Usage?
  - Mainly used for database and multi-threaded applications, such as logging, configuration settings

Builder Pattern

- Builds a complex object from simpler objects using step-by-step approach.

- What is interesting about it?
  - Using this pattern, it creates a clear separation between the representation of the object and the construction of it.

- Usage?
  - Main usage is when objects cannot be created in a single step.

Factory Method Pattern

- Defines an abstract or interface class for creating an object but then it lets the subclasses define which class to instantiate.

- What's interesting about it?
  - It allows the subclasses to choose the type of objects to create
  - It promotes loose-coupling
    - Code interacts solely with the resultant interface/abstract class so it will work with any class that implements it.
- Usage?
  - When a class wants its subclass to specify the objects to be created

Structural Patterns

- Decorator
- Adapter
- Proxy
- Facade

Decorator Pattern

-Uses a composition instead of inheritance to extend the functionality of an object when running (also known as the “Wrapper”)

-What’s interesting about it?

-Simplifies code by allowing you to develop a series of functionality from targeted classes instead of coding all the behavior into the object

-Usage?

-Used when wanting to add responsibilities to object without affecting other objects.

Adapter Pattern

- Converts the interface of a class into another interface that a client wants. (also known as a “Wrapper’

- What is interesting about it?
  - It allows two or more incompatible objects to interact.

- Usage?
  - When objects needs to utilize an incompatible interface.

Proxy Pattern

-Known as the “Placeholder”. It provides the control for accessing the original object.

-What is interesting about it?

-It provides the protection to the original object from the outside world.

-Usage?

-Used in Protective proxy scenarios where it acts as an authorization layer to verify whether the user has access to the appropriate content.

Facade Pattern

-Provides simplified interface to a set of interfaces in a subsystem while it hides the complexities of the subsystem from the client.

-What’s interesting about it?

-Shields the user from the sub-system complexities.

-Usages

-When several dependencies exist between clients and the implementation classes of an abstraction.

Behavioral Patterns

- Observer
- Strategy
- Template
- Command

Observer Pattern

-A one to one dependency so that when one object changes states all of its dependents are updated and notified.

-What's interesting about it?

-It provides the support for broadcast-type communication.

Usage?

-When the framework that is written needs to be enhanced in the future with enw observers with minimal changes.

Strategy Pattern

-A family of functionality that encapsulate each one and make them interchangeable.

-What’s interesting about it?

-It provides a substitute to subclassing.

Usage?

-When you need different variations of an algorithm.

Template Pattern

-The Skeleton of a function in an operation that deferes some steps to its subclasses

-What’s interesting about it?

- Common technique for reusing the code.

Usage?

-When a common behavior among subclasses should be moved to a single common class by avoiding the duplication.

Command Pattern

-Known as a Transaction, it encapsulated a request under an object as a command then passes it to the invoker object. The invoker looks for the right object which can handle this command then passes the command to the corresponding object that will execute the command

-What's interesting about it?

-Makes it easy to add commands because existing classes remain unchanged.

-Usage?

- When you need to create and execute requests at different times.

Factory Method Pattern

- Defines an abstract or interface class for creating an object but then it lets the subclasses define which class to instantiate.

- What's interesting about it?
  - It allows the subclasses to choose the type of objects to create
  - It promotes loose-coupling
    - Code interacts solely with the resultant interface/abstract class so it will work with any class that implements it.
- Usage?
  - When a class wants its subclass to specify the objects to be created

Structural Patterns

- Decorator
- Adapter
- Proxy
- Facade

Decorator Pattern

-Uses a composition instead of inheritance to extend the functionality of an object when running (also known as the “Wrapper”)

-What’s interesting about it?

-Simplifies code by allowing you to develop a series of functionality from targeted classes instead of coding all the behavior into the object

-Usage?

-Used when wanting to add responsibilities to object without affecting other objects.

Adapter Pattern

- Converts the interface of a class into another interface that a client wants. (also known as a “Wrapper’

- What is interesting about it?
  - It allows two or more incompatible objects to interact.

- Usage?
  - When objects needs to utilize an incompatible interface.

Proxy Pattern

-Known as the “Placeholder”. It provides the control for accessing the original object.

-What is interesting about it?

-It provides the protection to the original object from the outside world.

-Usage?

-Used in Protective proxy scenarios where it acts as an authorization layer to verify whether the user has access to the appropriate content.

Facade Pattern

-Provides simplified interface to a set of interfaces in a subsystem while it hides the complexities of the subsystem from the client.

-What’s interesting about it?

-Shields the user from the sub-system complexities.

-Usages

-When several dependencies exist between clients and the implementation classes of an abstraction.

Behavioral Patterns

- Observer
- Strategy
- Template
- Command

Observer Pattern

-A one to one dependency so that when one object changes states all of its dependents are updated and notified.

-What's interesting about it?

-It provides the support for broadcast-type communication.

Usage?

-When the framework that is written needs to be enhanced in the future with enw observers with minimal changes.

Strategy Pattern

-A family of functionality that encapsulate each one and make them interchangeable.

-What’s interesting about it?

-It provides a substitute to subclassing.

Usage?

-When you need different variations of an algorithm.

Template Pattern

-The Skeleton of a function in an operation that deferes some steps to its subclasses

-What’s interesting about it?

- Common technique for reusing the code.

Usage?

-When a common behavior among subclasses should be moved to a single common class by avoiding the duplication.

Command Pattern

-Known as a Transaction, it encapsulated a request under an object as a command then passes it to the invoker object. The invoker looks for the right object which can handle this command then passes the command to the corresponding object that will execute the command

-What's interesting about it?

-Makes it easy to add commands because existing classes remain unchanged.

-Usage?

- When you need to create and execute requests at different times.