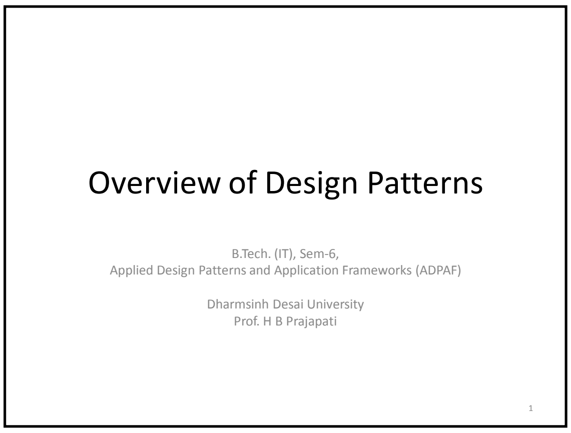
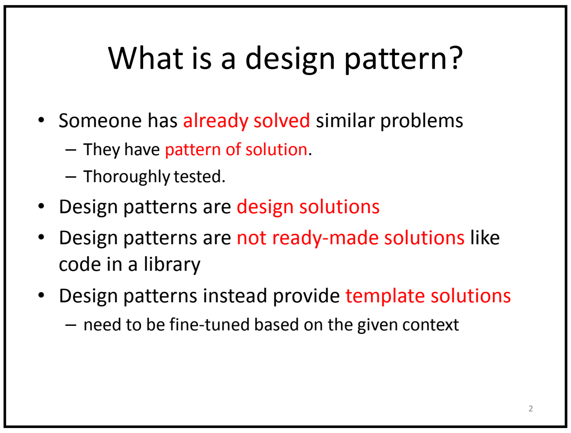


Overview of Design Patterns

B.Tech. (IT), Sem-6,
Applied Design Patterns and Application Frameworks (ADPAF)

Dharmsinh Desai University
Prof. H B Prajapati

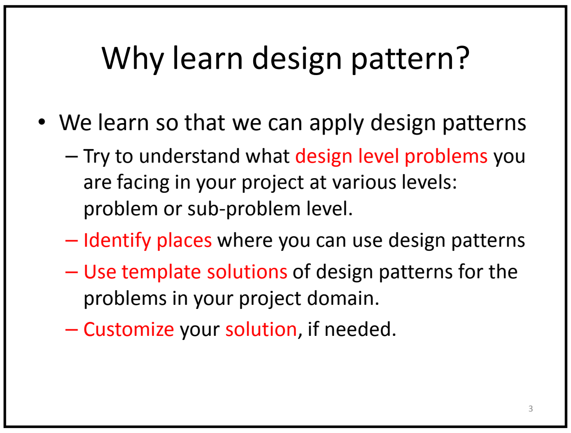
1



What is a design pattern?

- Someone has already solved similar problems
  - They have pattern of solution.
  - Thoroughly tested.
- Design patterns are design solutions
- Design patterns are not ready-made solutions like code in a library
- Design patterns instead provide template solutions
  - need to be fine-tuned based on the given context

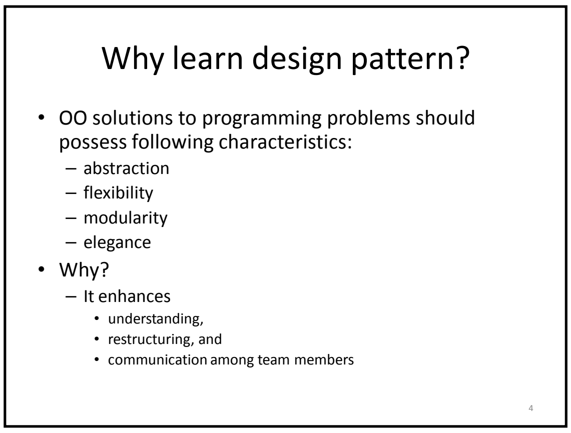
2



Why learn design pattern?

- We learn so that we can apply design patterns
  - Try to understand what design level problems you are facing in your project at various levels: problem or sub-problem level.
  - Identify places where you can use design patterns
  - Use template solutions of design patterns for the problems in your project domain.
  - Customize your solution, if needed.

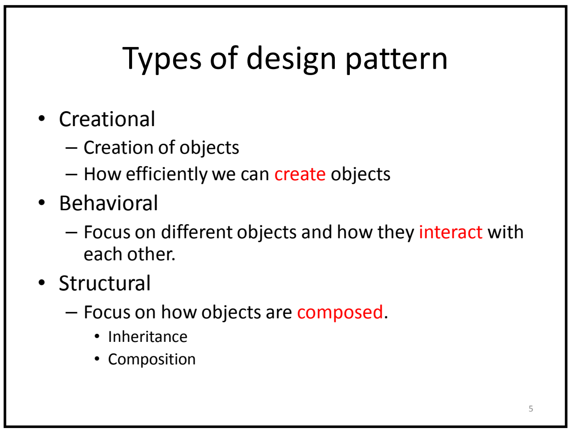
3



Why learn design pattern?

- OO solutions to programming problems should possess following characteristics:
  - abstraction
  - flexibility
  - modularity
  - elegance
- Why?
  - It enhances
    - understanding,
    - restructuring, and
    - communication among team members

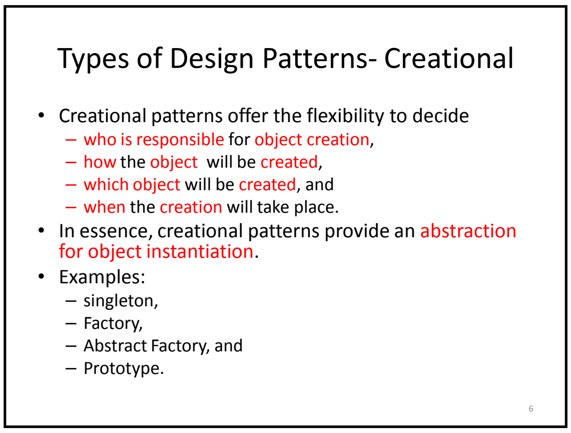
4



Types of design pattern

- Creational
  - Creation of objects
  - How efficiently we can create objects
- Behavioral
  - Focus on different objects and how they interact with each other.
- Structural
  - Focus on how objects are composed.
    - Inheritance
    - Composition

5



Types of Design Patterns- Creational

- Creational patterns offer the flexibility to decide
  - who is responsible for object creation,
  - how the object will be created,
  - which object will be created, and
  - when the creation will take place.
- In essence, creational patterns provide an abstraction for object instantiation.
- Examples:
  - singleton,
  - Factory,
  - Abstract Factory, and
  - Prototype.

6

Types of Design Patterns- Structural

- Structural patterns are focused on how related classes (and objects) are composed together to form a larger structure.
  - Examples:
    - Composite,
    - Decorator,
    - Proxy, and
    - Façade.

7

Types of Design Patterns- Behavioral

- Behavioral patterns define
  - the communication among the objects and
  - control the flow within the participating objects.
- Examples:
  - Mediator,
  - Chain of responsibility,
  - Observer,
  - State, and
  - Strategy.

8

Widely used creational design pattern

- Prototype
- Builder
- Singleton
- Factory

9

Widely used creational design pattern

- Prototype
  - Copy or clone a fully initialized instance.
  - E.g., Preparing initial setup of chess game, or carom game, or snooker game.
- Builder
  - Used when there is a complex object structure
  - Separates object construction from its representation.
  - E.g. Multicourse dinner

10

Widely used creational design pattern

- Singleton
  - Only one instance of a class can exist
  - For example Prime Minister of a country
  - E.g., java.lang.System
  - Important things
    - Constructor has to be private
    - @Singleton annotation is available
- Factory
  - Creates objects of a single family
  - E.g., manufacturing car (petrol, diesel, and variants)

11

Widely used Structural design pattern

- Proxy
- Decorator
- Façade (pronounced as fu saad)
- Adapter
- Flyweight

12

Widely used Structural design pattern

- Proxy
  - An object representing another object
  - E.g., Debit card is a proxy for our bank account
  - Represent a situation where an actual object is at a remote location, but we access it using local proxy object.
  - E.g., Java Remote Method Invocation uses proxy design pattern.
  - E.g., in Enterprise Java Bean (EJB) Remote and Home objects are proxy objects

Widely used Structural design pattern

- Decorator
  - Add responsibilities to objects dynamically
  - E.g., without using decorator design pattern, if we have 5 flavors of cake with 10 types of toppings, then we need to create 50 classes.
  - For example in Java IO
    - FileInputStream passed to BufferedInputStream passed to LineNumberInputStream

Widely used Structural design pattern

- Façade
  - A single class that represents entire subsystem.
  - For example, an event manger for Food, Decoration, Music, Dance, Invitation
  - E.g., Make online order
    - Check availability of product
    - Place order
    - Generate invoice
  - Advantages
    - Reduces network calls for various operations
    - Reduce coupling between web layer and data layer

Widely used Structural design pattern

- Adapter
  - Match interfaces of different classes
  - For example, connecting mobile for charging to 230 V socket, we use adapter which matches two different interfaces (230V and 5V)
  - E.g., Adapters are available for different event listener interfaces

Widely used Structural design pattern

- Flyweight
  - A fine-grained instance used for efficient sharing
  - We reuse object
  - For example, in old days, telephone switching office, cities interconnection line was reused.

Widely used Behavioral design pattern

- Chain of Responsibility
- Iterator
- State
- Strategy
- Observer
- Visitor
- Template method
- Command
- Memento
- Mediator

Widely used Behavioral design pattern

- Chain of Responsibility
  - A way of passing a request between a chain of objects
  - For example, leave approval process
  - Exception handling in Java

Widely used Behavioral design pattern

- Iterator
  - Sequentially access the elements of a collection.
  - Iterating on songs of an album using Next and Previous buttons
  - In Java, we iterate different types of collection object using a single interface Iterator
    - We do not need to know internal representation of the object (Queue, List, HashMap, etc.)

Widely used Behavioral design pattern

- State
  - To change an object’s behavior when its state changes.
  - Fan Speed Controller
    - We can represent each speed level as a State

Widely used Behavioral design pattern

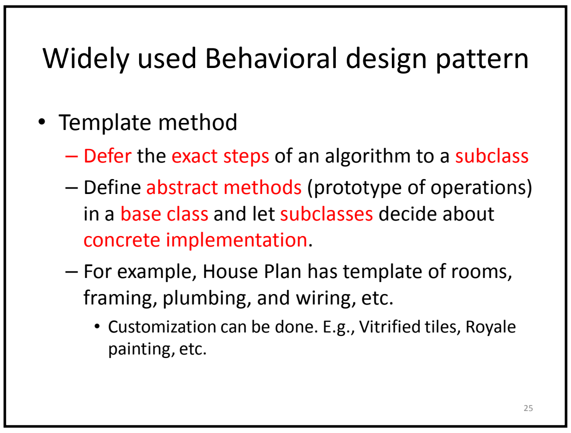
- Strategy
  - Encapsulates an algorithm inside a class
  - We create interface for strategy (e.g., sorting algorithm), and implement that interface.
  - We can change the implementation, i.e., from bubble sort to quick sort.
  - E.g., java.util.Comparator interface and compare() method (strategy)

Widely used Behavioral design pattern

- Observer
  - A way of notifying about changes to a number of objects
  - Cricket score is notified to Radio, TV, Internet, etc.
  - Online bidding
    - When someone places a bid, this change is notified to all participants
  - It is available in Java
    - Subject: object for which other objects are interested in changes. Subject extends Observable
    - Other objects implements Observer interface and
    - in via its update method changed object can be retrieved.

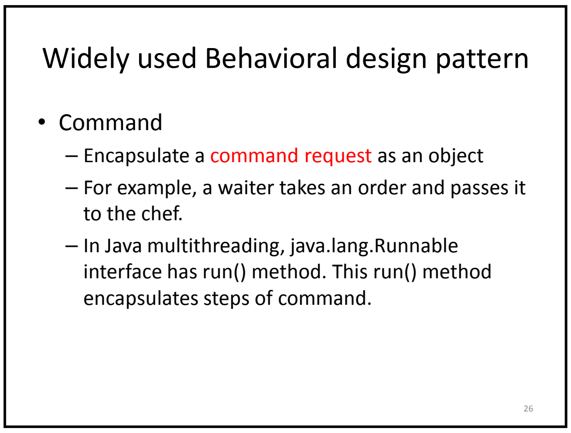
Widely used Behavioral design pattern

- Visitor
  - Visitor object defines a new operation to a class without change.
  - A person (visitor) hires a cab. Now visitor uses hired transportation as per his will/requirements.
  - Visitor object (person) changes the executing algorithm of an element object (car).
  - Design pattern
    - An element object has to accept visitor object
    - Visitor object handles the operation on the element object



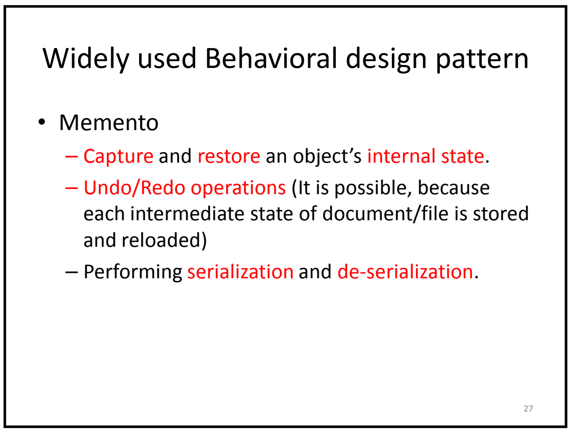
Widely used Behavioral design pattern

- Template method
  - Defer the exact steps of an algorithm to a subclass
  - Define abstract methods (prototype of operations) in a base class and let subclasses decide about concrete implementation.
  - For example, House Plan has template of rooms, framing, plumbing, and wiring, etc.
    - Customization can be done. E.g., Vitrified tiles, Royale painting, etc.



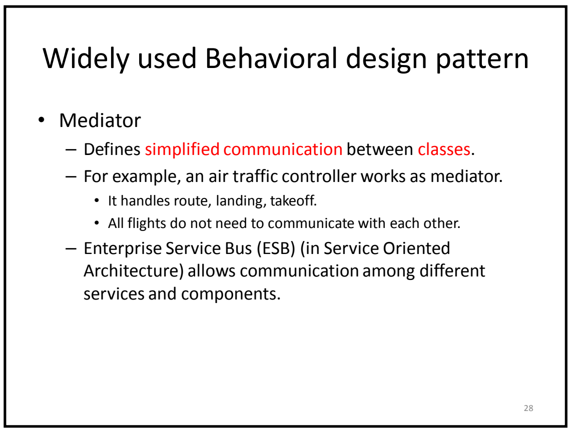
Widely used Behavioral design pattern

- Command
  - Encapsulate a command request as an object
  - For example, a waiter takes an order and passes it to the chef.
  - In Java multithreading, java.lang.Runnable interface has run() method. This run() method encapsulates steps of command.



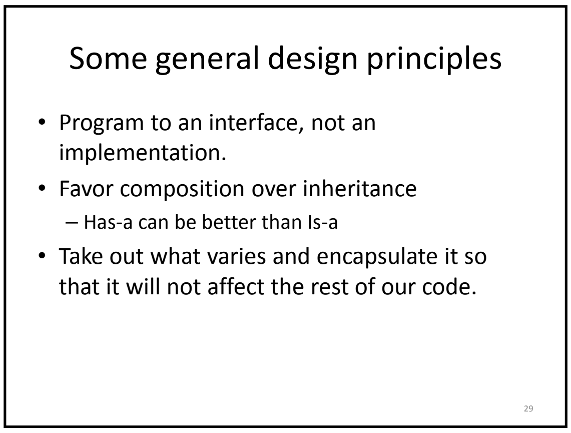
Widely used Behavioral design pattern

- Memento
  - Capture and restore an object's internal state.
  - Undo/Redo operations (It is possible, because each intermediate state of document/file is stored and reloaded)
  - Performing serialization and de-serialization.



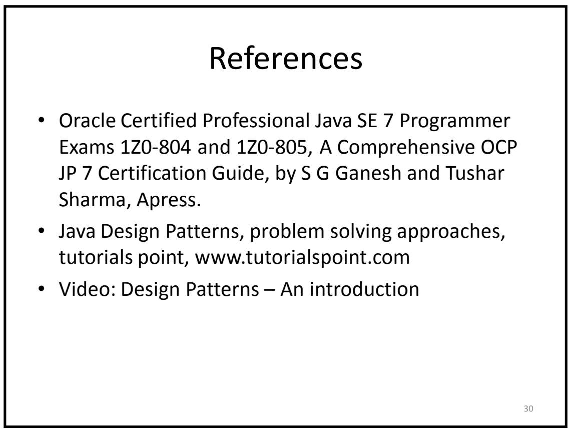
Widely used Behavioral design pattern

- Mediator
  - Defines simplified communication between classes.
  - For example, an air traffic controller works as mediator.
    - It handles route, landing, takeoff.
    - All flights do not need to communicate with each other.
  - Enterprise Service Bus (ESB) (in Service Oriented Architecture) allows communication among different services and components.



Some general design principles

- Program to an interface, not an implementation.
- Favor composition over inheritance
  - Has-a can be better than Is-a
- Take out what varies and encapsulate it so that it will not affect the rest of our code.



References

- Oracle Certified Professional Java SE 7 Programmer Exams 1Z0-804 and 1Z0-805, A Comprehensive OCP JP 7 Certification Guide, by S G Ganesh and Tushar Sharma, Apress.
- Java Design Patterns, problem solving approaches, tutorials point, www.tutorialspoint.com
- Video: Design Patterns – An introduction