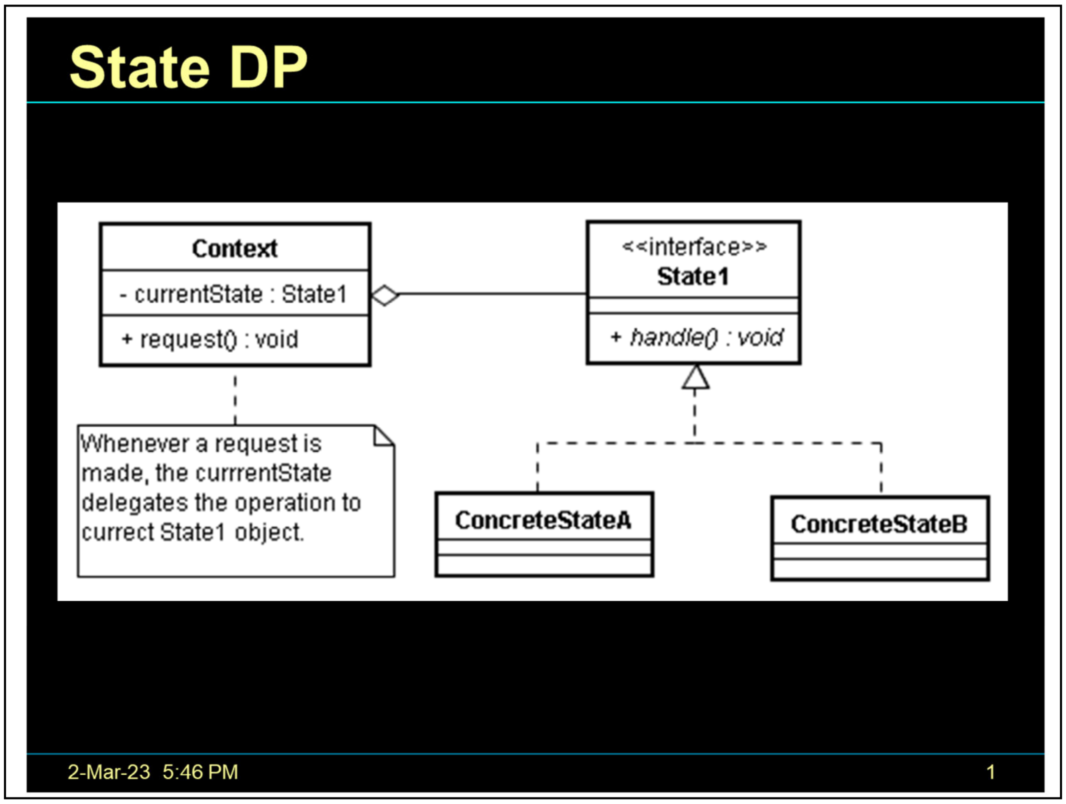
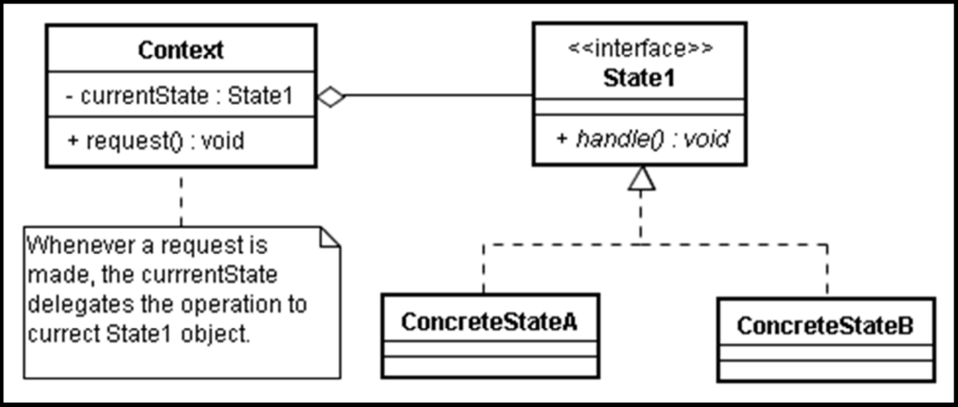


State DP



2-Mar-23 5:46 PM 1

The context

is of interest to Clients

maintains an instance of a concreteState subclass that defines the current state.

It switches from one state to another during its lifetime, in order to produce different behavior from the same method call(s).

The State

defines an interface for encapsulating the behavior associated with a particular state of the Context.

The ConcreteState subclasses

Each subclass implements a behavior associated with a state of the context.

==================================

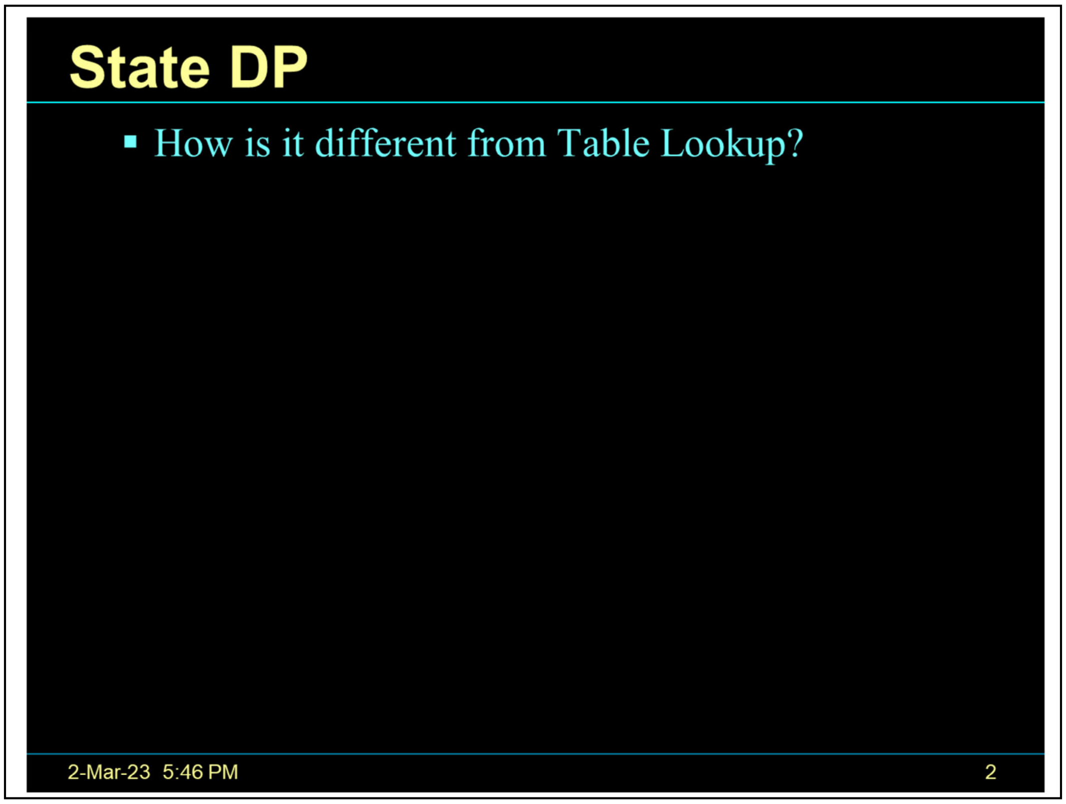
It allows an object to have many different behaviors that are based on its current internal state.

Clients never interact directly with the State objects.

It eliminates the necessity for a set of long, look-alike conditional statements scattered through the program’s code.

It makes the state transitions explicit.

Without this DP, some variable value implicitly determines the state of object.


State DP

- How is it different from Table Lookup?

2-Mar-23 5:46 PM 2

We can use a table to map state transitions.

The table maps every possible input to the succeeding state.

It converts a conditional code / virtual functions into table lookup.

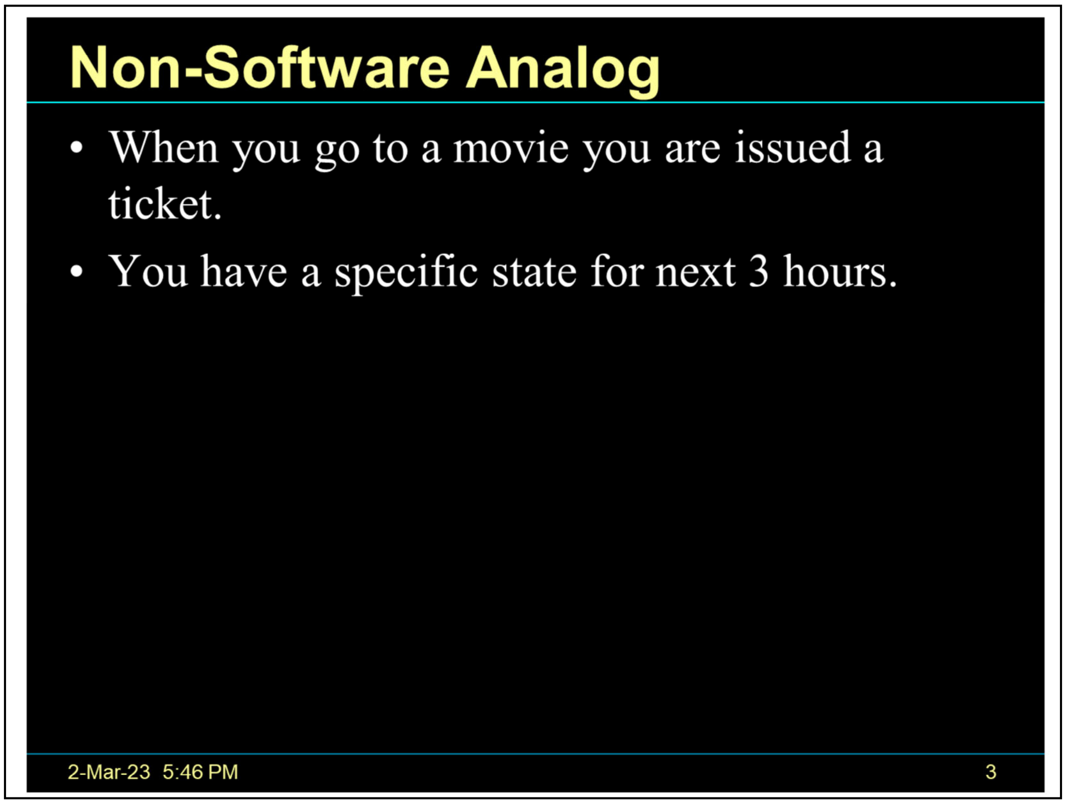
The State DP focuses on state-specific behavior whereas the table-driven approach focuses on defining state transitions.

=========================

We use the State DP when

An object's behavior depends upon its state and it must change its behavior at runtime depending on that state.

Operations have large, multipart conditional statements that depend on the object's state.

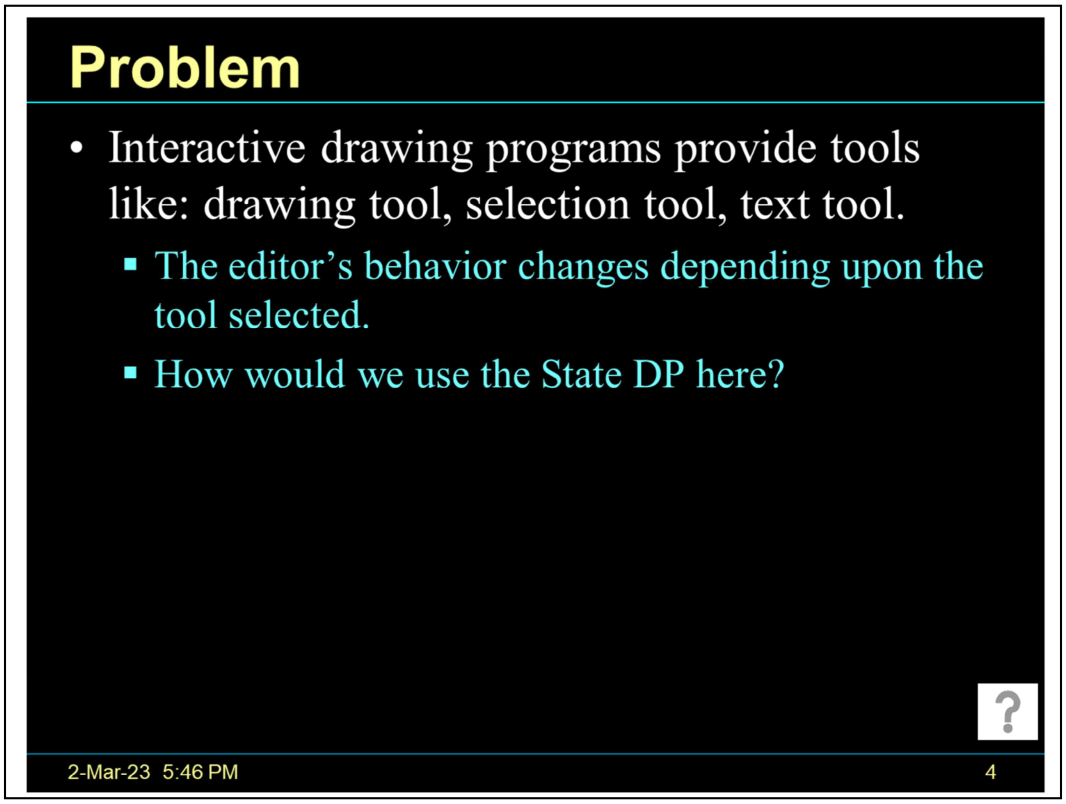


Non-Software Analog

- When you go to a movie you are issued a ticket.
- You have a specific state for next 3 hours.

2-Mar-23 5:46 PM 3

This ticket - or the absence thereof - encapsulates your privileges to see the movie for which you paid. It grants to entry into the theater and in so doing becomes torn. Once you have a torn ticket you are able to enter and leave the theater freely for the duration of the movie. Any time you are inside the theater, you may be subjected to a test: an usher may check your ticket to make sure you are properly authorized for the movie you are viewing.

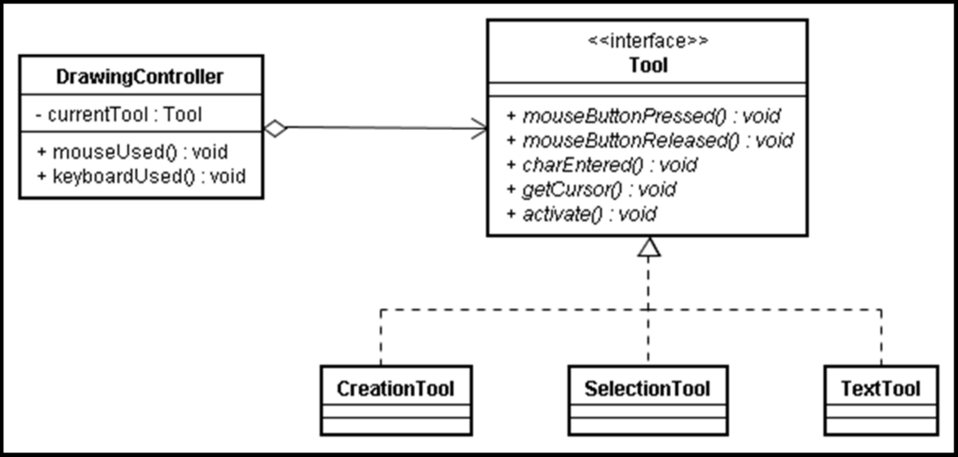


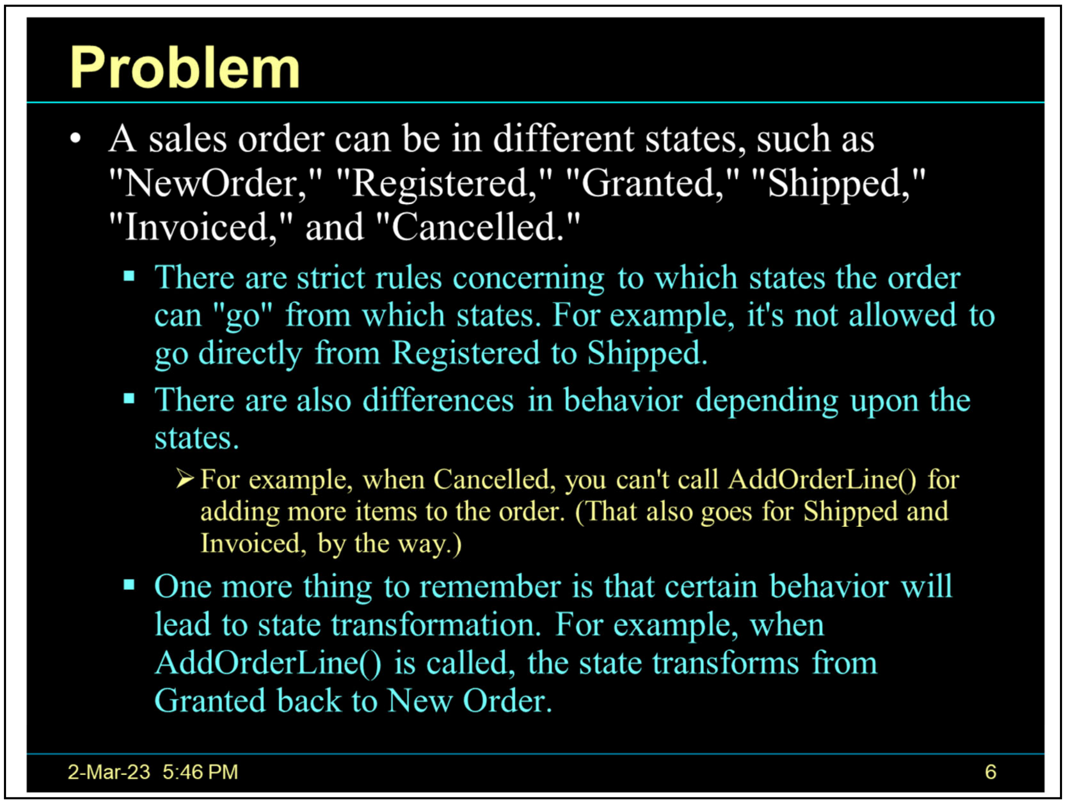
Problem

- Interactive drawing programs provide tools like: drawing tool, selection tool, text tool.
  - The editor's behavior changes depending upon the tool selected.
  - How would we use the State DP here?

2-Mar-23 5:46 PM 4

State DP


DrawingController
- currentTool : Tool
+ mouseUsed() : void
+ keyboardUsed() : void
<<interface>>
Tool
+ mouseButtonPressed() : void
+ mouseButtonReleased() : void
+ charEntered() : void
+ getCursor() : void
+ activate() : void
CreationTool
SelectionTool
TextTool

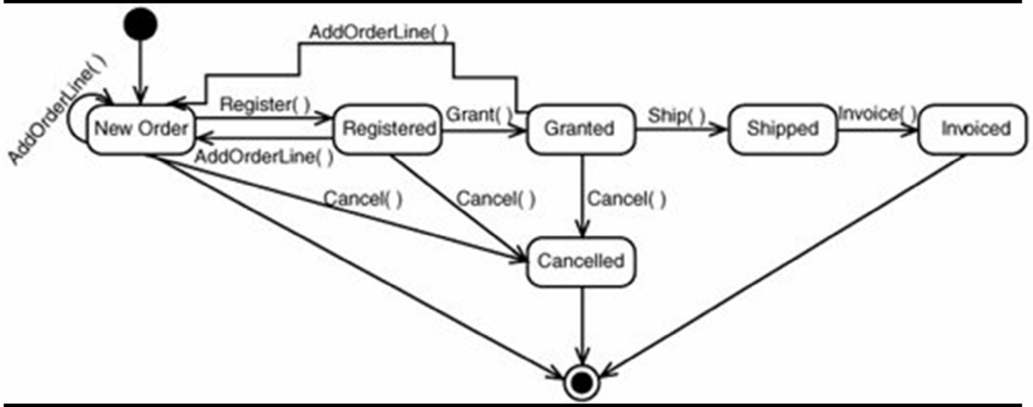


Problem

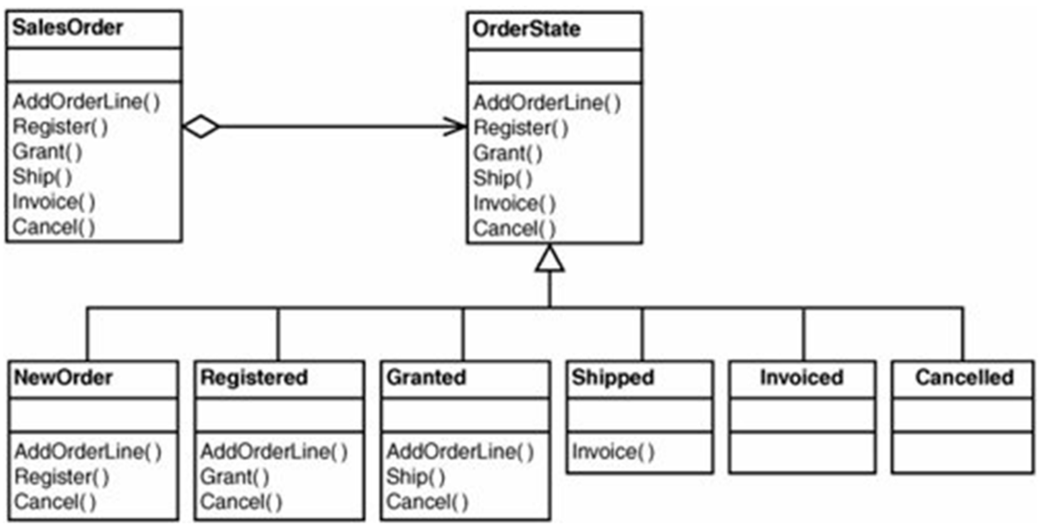
- A sales order can be in different states, such as "NewOrder," "Registered," "Granted," "Shipped," "Invoiced," and "Cancelled."
  - There are strict rules concerning to which states the order can "go" from which states. For example, it's not allowed to go directly from Registered to Shipped.
  - There are also differences in behavior depending upon the states.
    - For example, when Cancelled, you can't call AddOrderLine() for adding more items to the order. (That also goes for Shipped and Invoiced, by the way.)
  - One more thing to remember is that certain behavior will lead to state transformation. For example, when AddOrderLine() is called, the state transforms from Granted back to New Order.

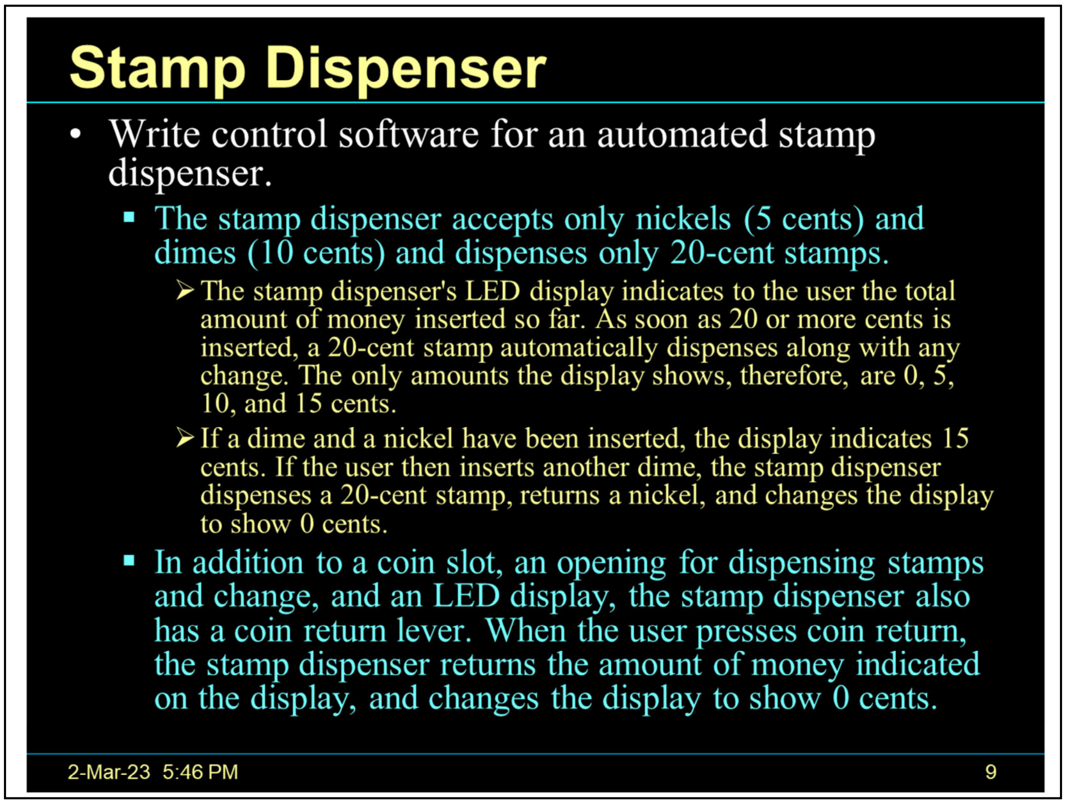
2-Mar-23 5:46 PM 6

Draw Class Diagram



Solution


SalesOrder
AddOrderLine()
Register()
Grant()
Ship()
Invoice()
Cancel()
OrderState
AddOrderLine()
Register()
Grant()
Ship()
Invoice()
Cancel()
NewOrder
AddOrderLine()
Register()
Cancel()
Registered
AddOrderLine()
Grant()
Cancel()
Granted
AddOrderLine()
Ship()
Cancel()
Shipped
Invoice()
Invoiced
Cancelled

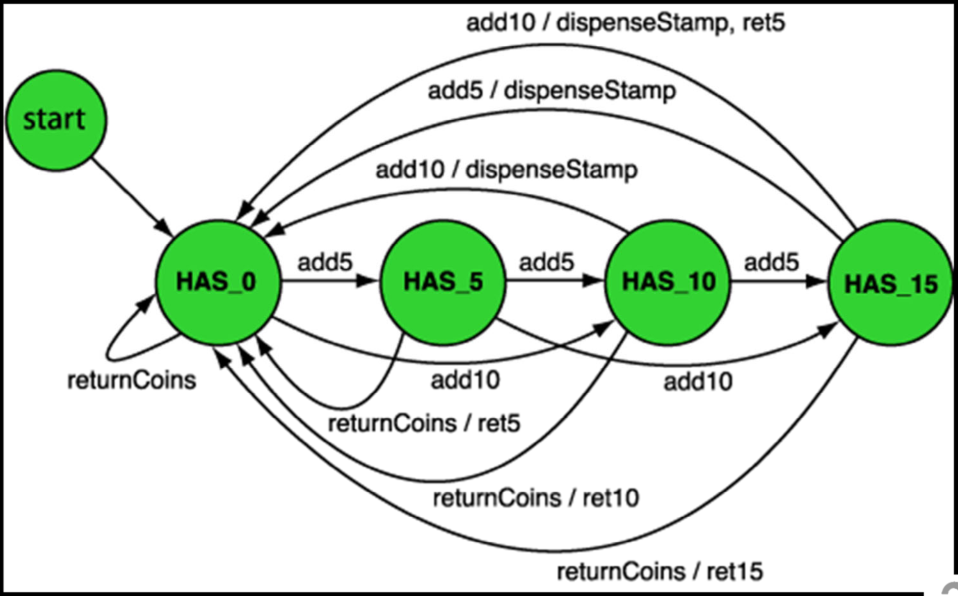


Stamp Dispenser

- Write control software for an automated stamp dispenser.
  - The stamp dispenser accepts only nickels (5 cents) and dimes (10 cents) and dispenses only 20-cent stamps.
    - The stamp dispenser's LED display indicates to the user the total amount of money inserted so far. As soon as 20 or more cents is inserted, a 20-cent stamp automatically dispenses along with any change. The only amounts the display shows, therefore, are 0, 5, 10, and 15 cents.
    - If a dime and a nickel have been inserted, the display indicates 15 cents. If the user then inserts another dime, the stamp dispenser dispenses a 20-cent stamp, returns a nickel, and changes the display to show 0 cents.
  - In addition to a coin slot, an opening for dispensing stamps and change, and an LED display, the stamp dispenser also has a coin return lever. When the user presses coin return, the stamp dispenser returns the amount of money indicated on the display, and changes the display to show 0 cents.

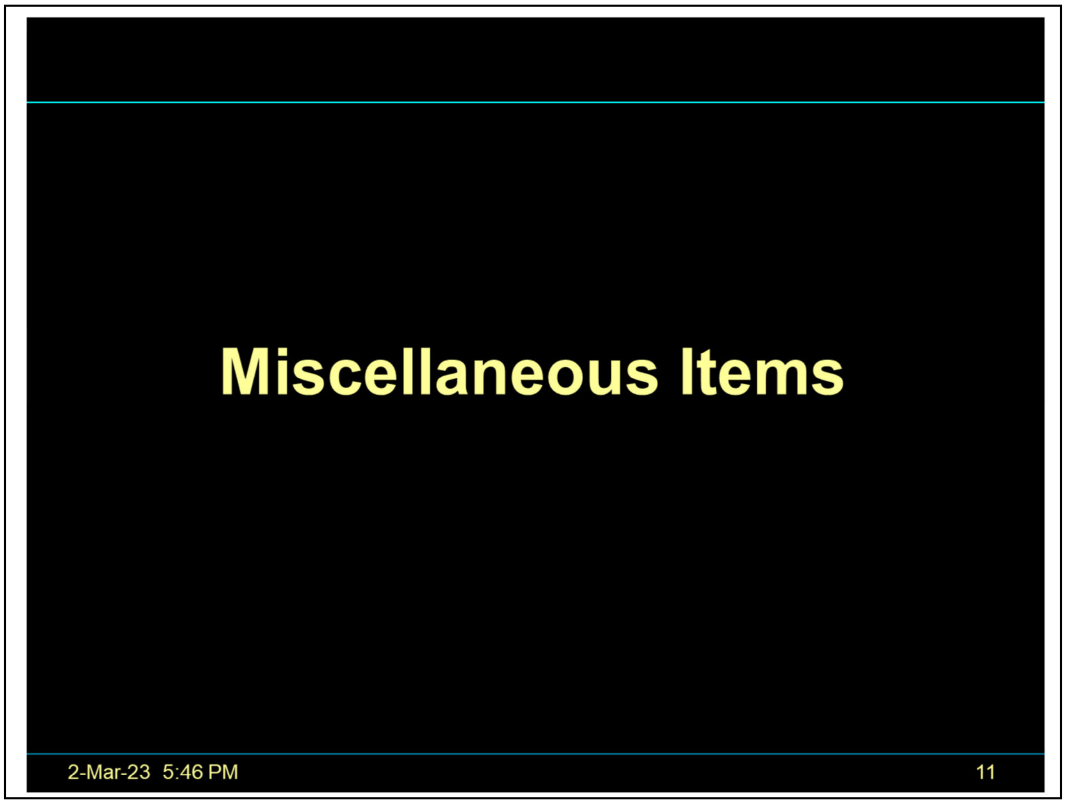
2-Mar-23 5:46 PM 9

State diagram



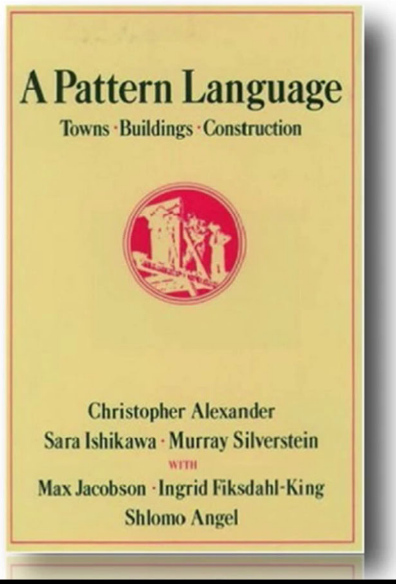
Draw the Class diagram using State DP

?



Miscellaneous Items

Design Patterns in Architecture


A Pattern Language
Towns · Buildings · Construction
Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

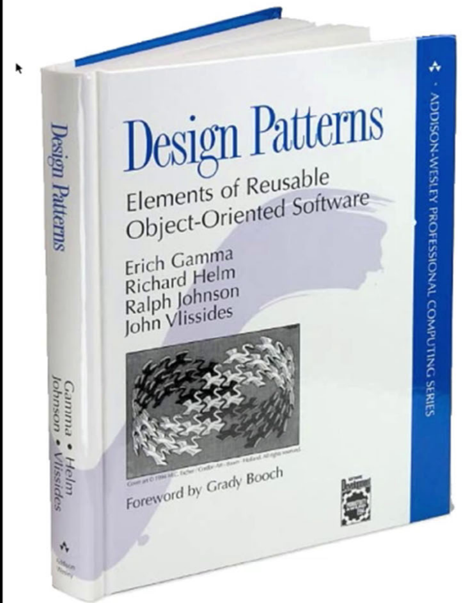
Christopher Alexander

Oxford University Press (1977)

*buildings, not software

2-Mar-23 5:46 PM 12

Gang of Four

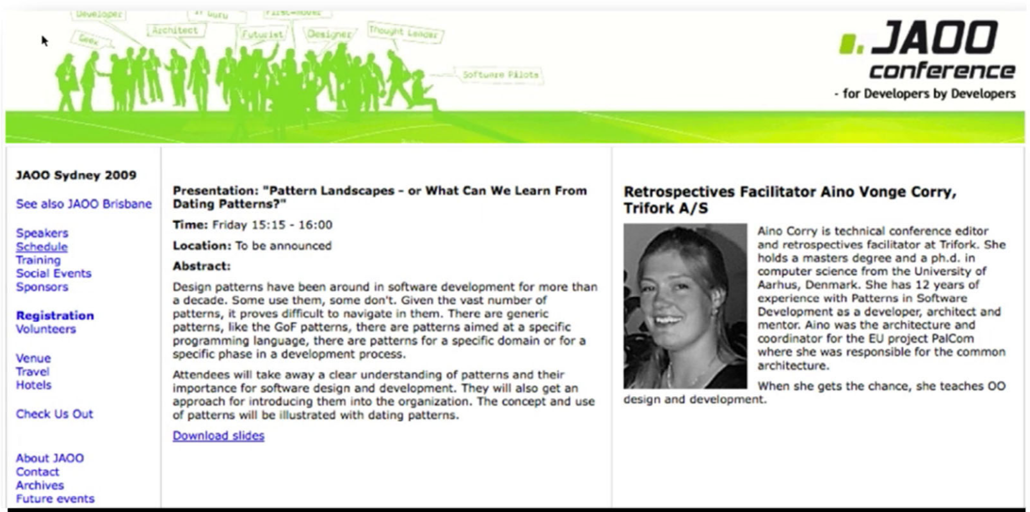

Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Published: 1984

2-Mar-23 5:46 PM

Patterns Everywhere



JAOO Sydney 2009

See also JAOO Brisbane

Speakers
Schedule
Training
Social Events
Sponsors

Registration
Volunteers

Venue
Travel
Hotels

Check Us Out

About JAOO
Contact
Archives
Future events

Presentation: "Pattern Landscapes - or What Can We Learn From Dating Patterns?"

Time: Friday 15:15 - 16:00

Location: To be announced

Abstract:

Design patterns have been around in software development for more than a decade. Some use them, some don't. Given the vast number of patterns, it proves difficult to navigate in them. There are generic patterns, like the GoF patterns, there are patterns aimed at a specific programming language, there are patterns for a specific domain or for a specific phase in a development process.

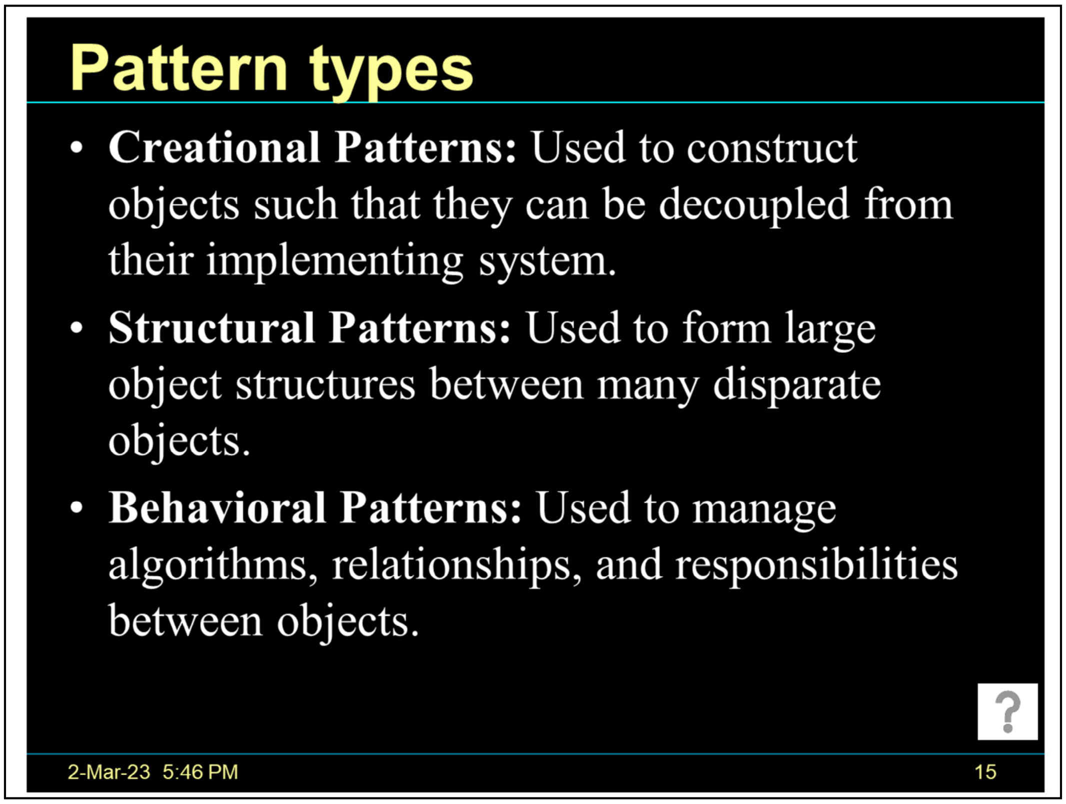
Attendees will take away a clear understanding of patterns and their importance for software design and development. They will also get an approach for introducing them into the organization. The concept and use of patterns will be illustrated with dating patterns.

Download slides

Retrospectives Facilitator Aino Vonge Corry, Trifork A/S

Aino Corry is technical conference editor and retrospectives facilitator at Trifork. She holds a masters degree and a ph.d. in computer science from the University of Aarhus, Denmark. She has 12 years of experience with Patterns in Software Development as a developer, architect and mentor. Aino was the architecture and coordinator for the EU project PalCom where she was responsible for the common architecture.

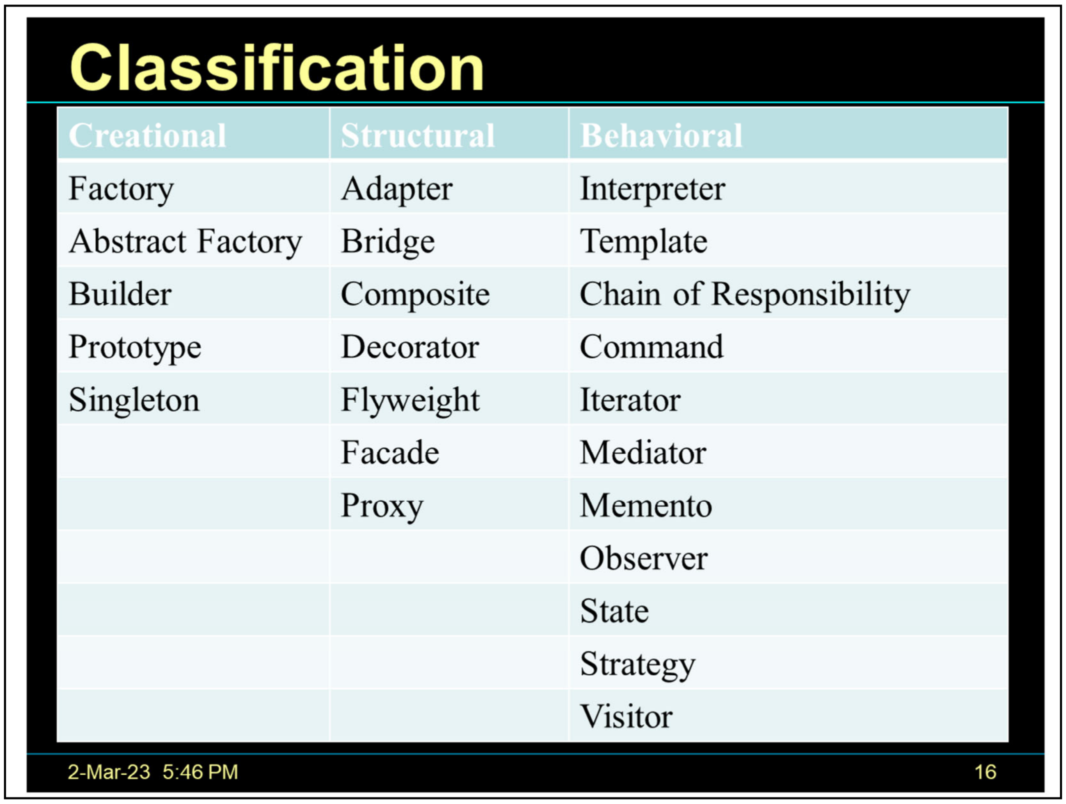
When she gets the chance, she teaches OO design and development.



Pattern types

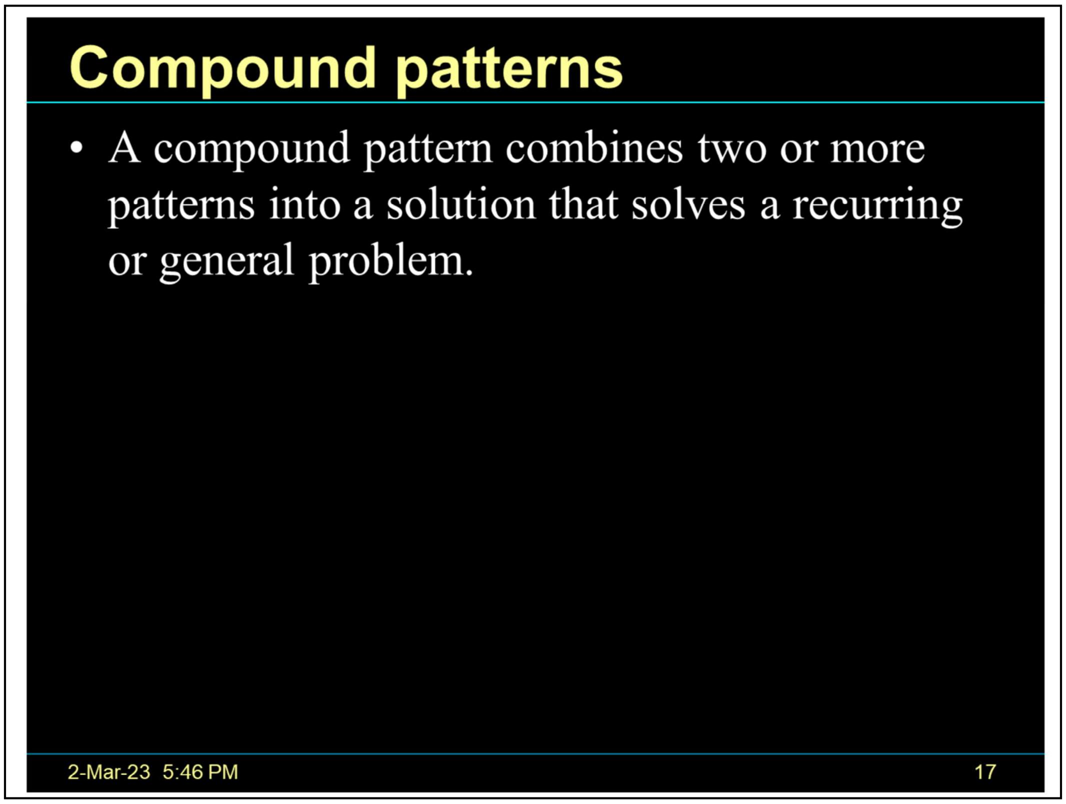
- **Creational Patterns:** Used to construct objects such that they can be decoupled from their implementing system.
- **Structural Patterns:** Used to form large object structures between many disparate objects.
- **Behavioral Patterns:** Used to manage algorithms, relationships, and responsibilities between objects.

2-Mar-23 5:46 PM 15



Classification

Creational	Structural	Behavioral
Factory	Adapter	Interpreter
Abstract Factory	Bridge	Template
Builder	Composite	Chain of Responsibility
Prototype	Decorator	Command
Singleton	Flyweight	Iterator
	Facade	Mediator
	Proxy	Memento
		Observer
		State
		Strategy
		Visitor



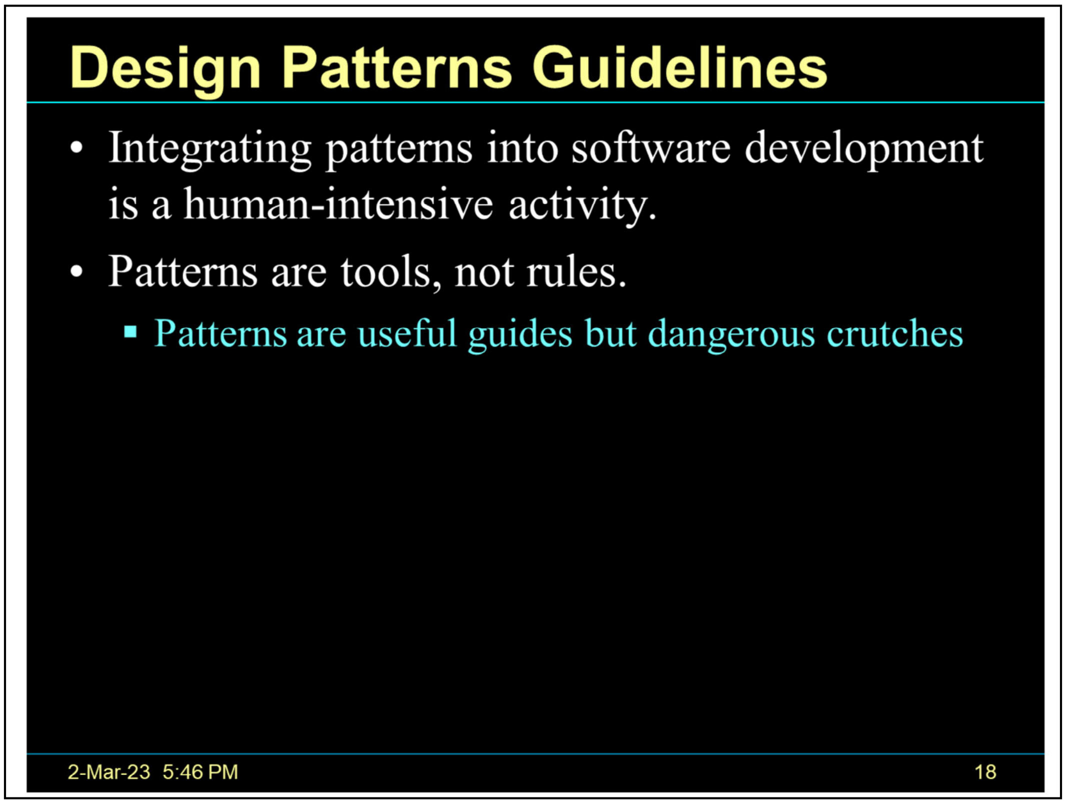
Compound patterns

- A compound pattern combines two or more patterns into a solution that solves a recurring or general problem.

2-Mar-23 5:46 PM 17

Patterns are often used together and combined within the same design solution.

This is not a new design pattern but existing pattern working together.



Design Patterns Guidelines

- Integrating patterns into software development is a human-intensive activity.
- Patterns are tools, not rules.
  - Patterns are useful guides but dangerous crutches

2-Mar-23 5:46 PM 18

Patterns do not lead to direct code reuse.

Patterns are tools:

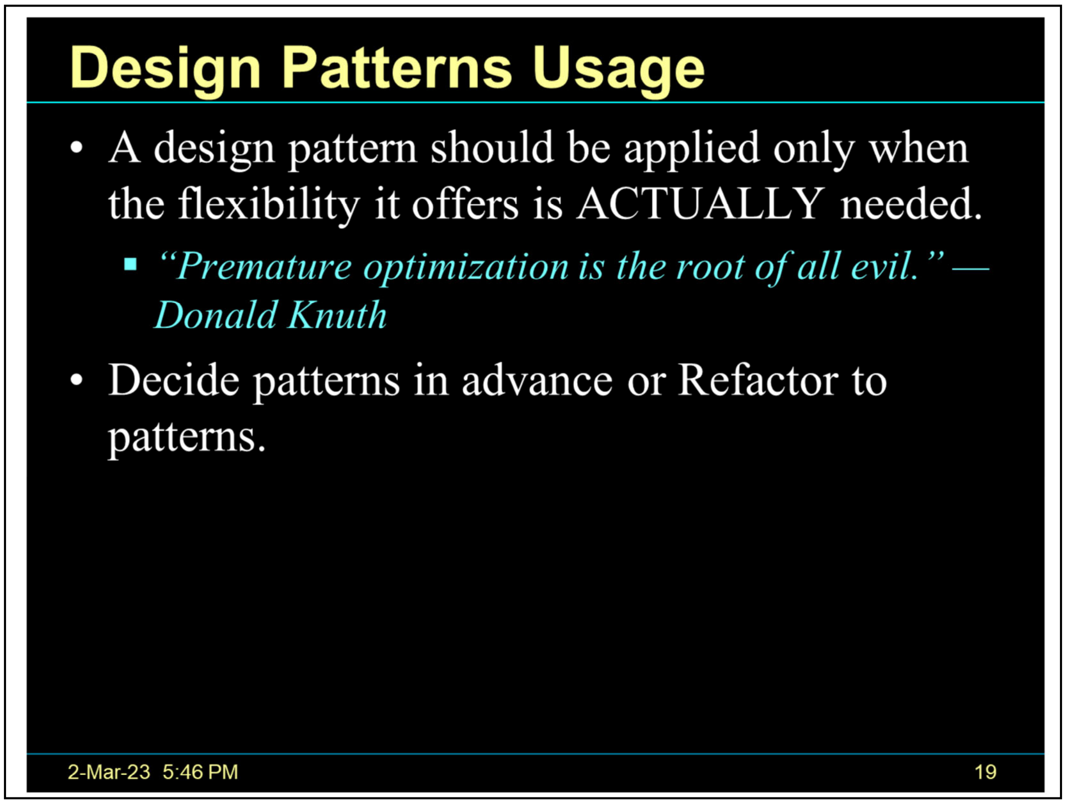
 They need to be tweaked and adapted to the problem at hand.

A Design pattern can be applied only if we are clear about what we want to do.

 A design pattern tells how to solve a particular problem.

 How often do programmers/designers have a clear idea about the problem?

  So many people find the use of Design patterns frustrating.



Design Patterns Usage

- A design pattern should be applied only when the flexibility it offers is ACTUALLY needed.
  - *"Premature optimization is the root of all evil."* — *Donald Knuth*
- Decide patterns in advance or Refactor to patterns.

2-Mar-23 5:46 PM 19

Often design patterns achieve the flexibility by introducing additional levels of indirection. They add complexity to our code.

Like optimization, we must delay using a DP.

Apply design patterns extensively while designing a framework / library interface for external world.

For custom applications

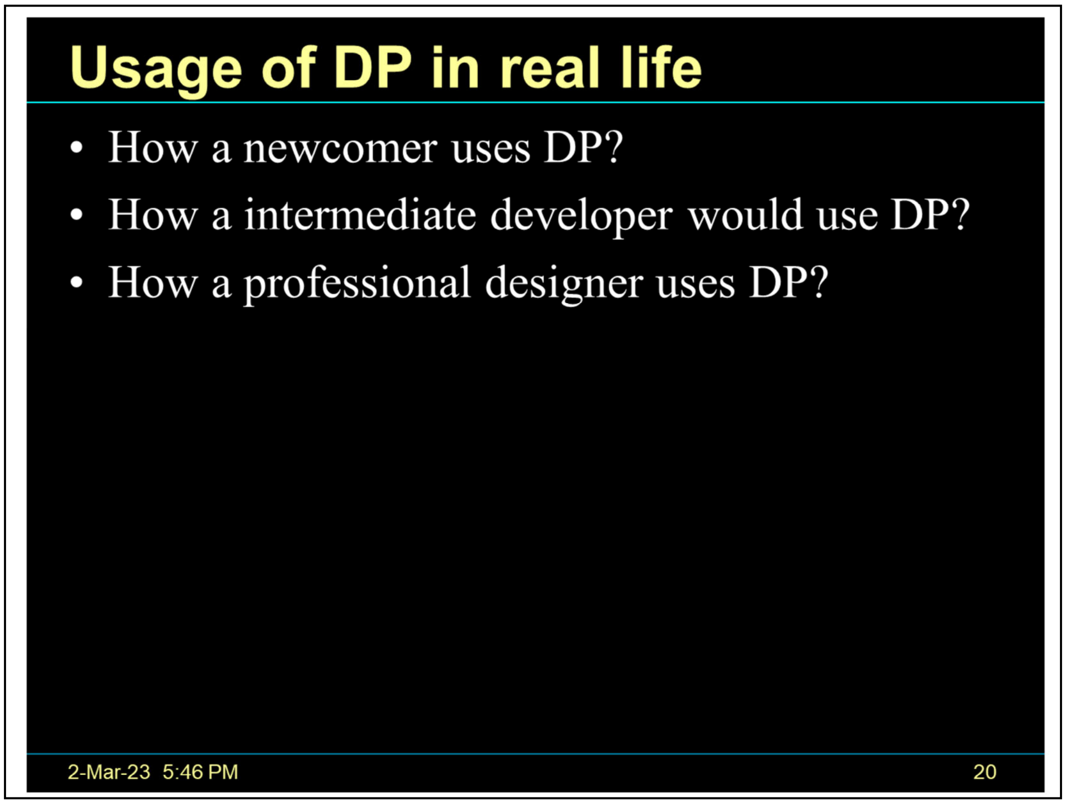
Use DPs *only if* the requirements are clear and the flexibility of DPs is needed.

If requirements are not clear,

First get the code working with the simplest solution possible.

Ensure that the code does what needs to be done.

Then refactor to use a design pattern, if required.



Usage of DP in real life

- How a newcomer uses DP?
- How a intermediate developer would use DP?
- How a professional designer uses DP?

2-Mar-23 5:46 PM 20

How a newcomer uses DP?

He tries to use it everywhere.

How a intermediate developer would use DP?

He tries to estimate where patterns are needed and where they are not?

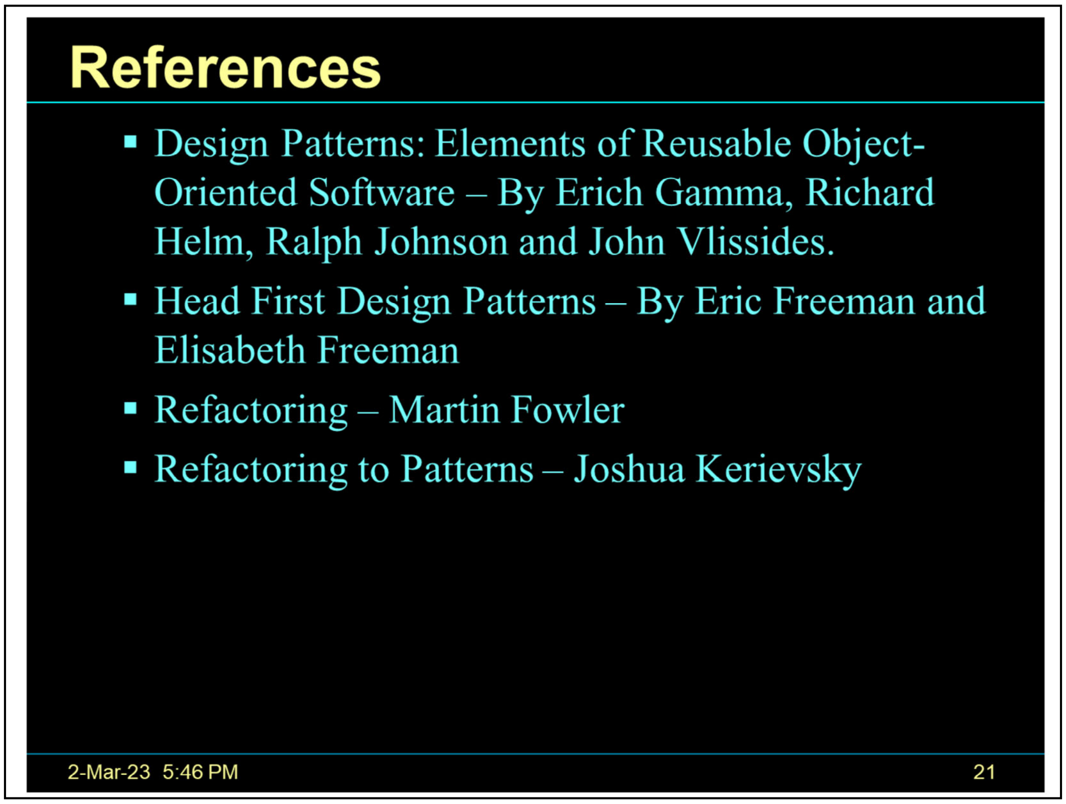
He tries to adapt the patterns to the problem at hand.

How a professional designer uses DP?

He designs the simplest solution without worrying about the list of DPs.

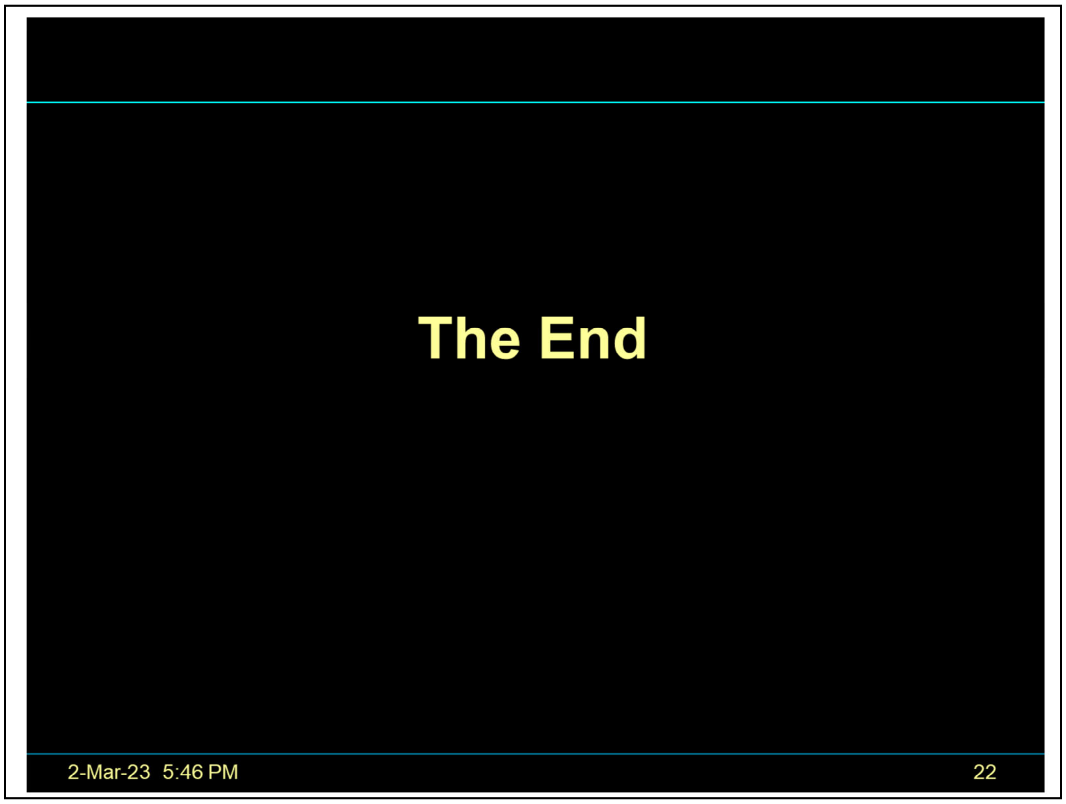
The design uses the necessary patterns with necessary customization.

Patterns have become ingrained and their use in the unconscious.


References

- Design Patterns: Elements of Reusable Object-Oriented Software – By Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides.
- Head First Design Patterns – By Eric Freeman and Elisabeth Freeman
- Refactoring – Martin Fowler
- Refactoring to Patterns – Joshua Kerievsky

2-Mar-23 5:46 PM 21



The End

2-Mar-23 5:46 PM 22