CISC 3120

C20: Some Design Patterns

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Concept of design patterns
- A few creational patterns
  - Static factory
  - Abstract factory
  - Builder
- A behavioral pattern
  - Observer
- References

Design Patterns

- Design solutions to recurring problems
  - Problem: what recurring problems to solve?
  - Solution: what is the solution?
  - Context: under what conditions that the solution is intended to solve the recurring problem?

Static Factory

- A static method returns instances of the class
  - Static method belongs to class
    - There is no "this" instance of the class
  - Examples in Java API
    - javafx.scene.paint.Color
      - static Color rgb(int red, int green, int blue)

Problem, Solution, and Context

- There is a need to construct the instance of the class in many different ways
  - Wait! Would constructors solve the problem?
    - How well does overloading solve the problem?
- Static factory method
  - Example: java.text.Collator
    - static Collator getInstance(Locale desiredLocale)
- When do we create static factory method for our own classes?

Abstract Factory

- There are many subclasses that share the same set of methods
  - Wait! Would constructors solve the problem?
    - You have to rely on concrete subclasses, how well do their constructors help you solve the problem?
- Factory interface (Java 8) or abstract Factory class
  - Example: (In the JavaFXDrawingBoard app) javafx.scene.control.SpinnerValueFactory<T>
- When do we create factory interface or abstract factory method for our own classes?

Builder

- Objects of a classes can be constructed in extremely many different ways
  - Wait! Can we use constructors, static factory method, abstract factory class?
- Solution
  - Builder builds an object step by step, with initialization parameter
  - Example:
    - (In CmdLineArgsDemo app) org.apache.commons.cli.Options
      - Options addOption(…)
    - (In LinkNetInterfaceExplorer) java.lang.StringBuilder
      - StringBuilder append(…)
- When do we create factory interface or abstract factory method for our own classes?

Observer

- One depends on many other classes
  - In MVC, when model changes, view also needs to change.
- Solution
  - An observable object can have one ore observers, and observers can be notified the changes the observable object
  - Commonly used in the User Interface design
- When do we create factory interface or abstract factory method for our own classes?

Example: Using Java Observable Class

- Java provides java.util.Observable
- See ObservableDemo in the "designpattern" directory of the "samplerepository" repository

Example: Using JavaFX Beans

- Package javafx.beans and javafx.beans.property
- A few example programs
  - SimpleEditor
  - TcpMessenger

Additional Reading

- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Reading assignments at the class website

Questions?

- Discussed a few design patterns
  - Static factory
  - Abstract factory
  - Builder
  - Observer
- A few classes and interfaces
  - Timer, TimerTask, Runnable, Observable, JavaFX Property