Design Patterns

Prof. Dr. Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg

ADAP C08

Three Examples

1. File / Directory
2. Position / Portfolio
3. TestCase / TestSuite

File / Directory Example 1 / 3

File / Directory Example 2 / 3

File / Directory Example 3 / 3

File / Directory Exercise

Position / Portfolio Example

TestCase / TestSuite Example

Composite Structure Diagram (Original)

Quiz: Configuring a Computer

- You are configuring a computer. The computer consists of parts. Some parts are atomic (a keyboard, a memory bank, a hard disk), some are composite (memory subsystem, storage subsystem, video subsystem), meaning you can configure its parts.

  Using the Composite design pattern, how would you design a class hierarchy to represent a computer configuration?

  Select all correct statements.

  - Each type of atomic part is represented as its own class.
  - Each type of composite part is represented as its own class.
  - All part classes are direct subclasses of an abstract Part class.

Answer 1 / 2: Configuring a Computer

Answer 2 / 2: Configuring a Computer

- **How would you design a class hierarchy to represent a computer configuration?**
  - Each type of atomic part is represented as its own class.
    - **Yes. Different types of objects should be represented as different classes.**
  - Each type of composite part is represented as its own class.
    - **Yes. Different types of objects should be represented as different classes.**
  - All part classes are direct subclasses of an abstract Part class.
    - **No. Having a Part class makes sense, but there will be many part classes that will not be direct subclasses. An example are the classes for the specific types of subsystems.**

Definition of Design Pattern

The **abstraction** of a common **solution** to a recurring **problem** for a given **context**. [DR]

From a Written Exam

Benefits of Using Design Patterns

Faster, better, cheaper …

1. **designing of software**
2. **documenting software**
3. **communicating designs**

The Design Patterns (“Gang-of-Four”) Book

Highly Abridged History of Design Patterns

1. A Pattern Language
2. “No Object is an Island”
3. ET++ and Interviews
4. Design Pattern Catalog
5. A System of Pattern

Pragmatics of Design Patterns

1. Descriptions
2. Collections
3. Applications

Describing Design Patterns 1 / 2

Problem: How to design a uniform yet flexible object hierarchy?

Context: You need an object hierarchy that you want to handle in a uniform way yet extend it dynamically. Frequently, algorithms need to run over the hierarchy.

Solution: Separate container functionality from domain behavior. Create a container class that can manage, at runtime, components of a generic type. Create all domain-specific classes separately. Make all classes implement the generic component protocol.

Design Pattern Description Formats 2 / 2

Collections of Design Patterns

1. Pattern Collections
2. Pattern Handbooks
3. Pattern Languages

Design Pattern Map

Applying Design Patterns

1. By-hand Instantiation
2. As a Design Template
3. As a Language Feature

Design Pattern vs. Instance (Model)

- Pattern
  - Illustration, not a model
  - Generic terms, for example
    - Component, Composite, Leaf
    - getComponent, addComponent

- Instance
  - A specific model (UML, code)
  - Specific terms, for example
    - Test, TestCase, TestSuite
    - run, addTest, getTests

Design Pattern vs. Template

Design vs. Implementation

Quiz: Abstraction Levels

- You are looking at a class diagram with class names like KeyboardPart, MemorySubsystem, and GraphicsCard.

  The class diagram represents most likely what type of model?

  Select all that apply.

  - A design pattern
  - A design template
  - An implementation

Answer: Abstraction Levels

- **The class diagram represents most likely what type of model?**
  - A design pattern
    - **No. A design pattern (illustration of possible class models) should not contain application-specific class names.**
  - A design template
    - **No. A design template (class model for copying) should not contain application-specific class names.**
  - An implementation
    - **Yes. Application-specific class names indicate an implementation of a design pattern.**

Singleton Example 1 / 2

```
public class PhotoFactory {
  private static PhotoFactory instance = new PhotoFactory();

  public static PhotoFactory getInstance() {
    return instance;
  }

  protected PhotoFactory() {
    // do nothing
  }

  ...
}
```

Singleton Example 2 / 2

```
public class PhotoFactory {
  private static PhotoFactory instance = null;

  public static synchronized PhotoFactory getInstance() {
    if (instance == null) {
      setInstance(new PhotoFactory());
    }
    return instance;
  }

  protected static synchronized void setInstance(PhotoFactory pf) {
    assert instance == null;
    assert pf != null;
    instance = pf;
  }

  protected PhotoFactory() {
    // do nothing
  }
  ...
```

As a Programming Language Feature

- Double dispatch, for example: **`draw(device, figure);`**

Java Annotation Type for Design Patterns

```
@interface DesignPattern {
  String name();
  String[] participants();
}
```

Annotated File / Directory Example

```
@DesignPattern {
  name = "Composite",
  participants = { "Component" }
}
public class Node { ... }
```

```
@DesignPattern {
  name = "Composite",
  participants = { "Composite" }
}
public class Directory extends Node { ... }
```

```
@DesignPattern {
  name = "Composite",
  participants = { "Leaf" }
}
public class File extends Node { ... }
```

Levels of (Design) Patterns

1. **Architectural Patterns [1]**
2. **Design Patterns**
3. **Programming Patterns**

[1] A.k.a architectual style

Example of an Architectural Pattern

- Publish / Subscribe Architecture
  - Purpose
    - Create a system that can be
      - easily extended and
      - evolved at runtime
  - Components
    - Events: Data structures that capture a particular event
    - Publishers: Provide (and possibly create) events to the system
    - Subscribers: Receive events from publishers
    - Event Channels: Link subscribers to publishers
  - Examples
    - Linda (historic)
    - MQSeries (current)
    - ESB (whole category)

Example of a Programming Pattern (“Idiom”)

```
public class Counter {
  protected int count = 0;

  public synchronized int getNext() {
    return count++;
  }

  ...

}
```

Review / Summary of Session

- Design patterns
  - Definition, purpose, history
  - When compared with other patterns
  - Ways of implementing patterns

- Collections of patterns
  - Collections, handbooks, languages
  - Relationships between patterns

Thank you! Questions?

dirk.riehle@fau.de – https://oss.cs.fau.de

dirk@riehle.org – https://dirkriehle.com – @dirkriehle

Credits and License

- Original version
  -
  -
- Contributions
  - None yet