Software Analysis and Design: Modeling

Hogeschool Rotterdam
2021

Informatica::Analysis::INFSADx1-A

Agenda

- Software Design
- Design Patterns

Part 1: Software Design

Questions

Part 1:

- What do we mean by software design?
- Why do we need a proper design?
- What are the techniques to discover and organise candidate objects?
- How does “decomposition” help in the design?
- What is cohesion? Which one is desirable: low or high?
- What is coupling? Which one is desirable: low or high?

What is Software Designing?

It is *a process* of:

- Defining methods, functions, classes / objects, modules, etc.
- Defining overall structure
- Defining relationships and interactions among the elements (classes / objects, methods / functions, ...)

so that the resulting *functionality* will satisfy the extracted *requirements*.

We have learned UML, so ...

Can we design a software?

We have learned UML, so ...

Can we design a software?

We have learned a *visual language* to express our *thoughts* during the *design process*.

Designing a software is more than just UML.

Why “Design”?

One may say: *“I have developed many programs without a proper design, and they all work fine.”*

Software Engineering is **more** than just *making a software*
You will need to **maintain** and **change** it *for a long time*.
Now answer these:

- Is it hard to **change** the code?
- Does **modifying** a small code *producing a bug* somewhere else?
- Is it hard to **reuse** the code?
- Is it hard to **maintain** the software *after its release*?

Why “Design”?

Now answer these:

- Is it hard to **change** the code?
- Does **modifying** a small code *producing a bug* somewhere else?
- Is it hard to **reuse** the code?
- Is it hard to **maintain** the software *after its release*?

Without applying proper design principles, more likely the answer for the above questions will be positive.

Remember ...

You will be working in a team.

Designing is:

- not just coding
- not only UML
- about applying design principles
- about communicating ideas to other developers
- about understanding the consequences of your choices:
  - “Why do I need inheritance here?”
  - “What if I divide this one class ... into three classes?”
  - “Which object should be the receiver of the message … ?”
  - “What is the state of the object after receiving the message … ?”
  - etc

Some principles ...

In addition to experience there are a few important fundamental principles to know.

Concepts like:

- Cohesion
- Coupling
- Separation of Concerns

Activity [IN CLASS]

Read the following articles:

1. OOAD - Conceptual Design (click here) [~ 10 min]
   https://medium.com/omarelgabrys-blog/object-oriented-analysis-and-design-conceptual-model-part-2-ce730ac4eb31
2. An Overview of OO Design (click here) [~ 15 min]
   https://medium.com/free-code-camp/a-short-overview-of-object-oriented-software-design-c7aa0a622c83

Questions

Part 1:

- What do we mean by software design?
- Why do we need a proper design?
- What are the techniques to discover and organise candidate objects?
- How does “decomposition” help in the design?
- What is cohesion? Which one is desirable: low or high?
- What is coupling? Which one is desirable: low or high?

Part 2: Design Patterns

Questions

Part 2:

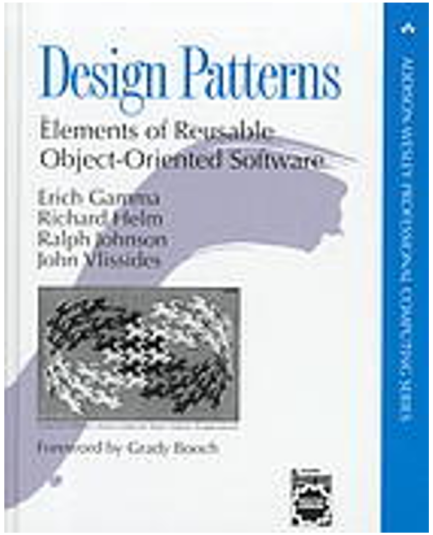
- What is a design pattern and why do we study them?
- How abstract classes and interfaces are useful in software maintenance?
- What are the categories of design patterns? Name one from each category.
- When do we use Singleton pattern?
- When is it helpful to use Observer pattern?

Introduction

In software design, you will encounter similar problems over and over again.

You will find out a common patterns in those problems.

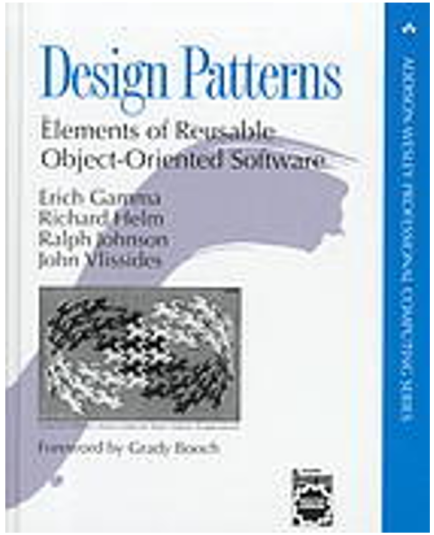
Experts classified these common patterns and proposed *successful solutions*.



https://en.wikipedia.org/wiki/Design_Patterns

Motivations

- Reusable successful designs
- Efficient in documentation, communication and maintenance.
- Efficient in design, development and test process.



https://en.wikipedia.org/wiki/Design_Patterns

Background

In order to understand design patterns you need to be experienced in using:

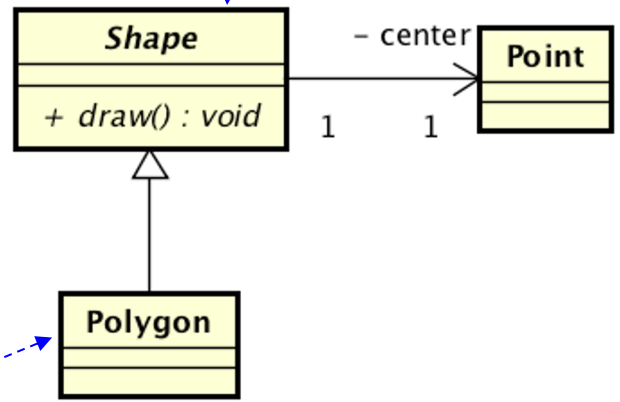
- Abstract and Concrete classes
- Interfaces
- Inheritance and Realisation

Abstract class

Classes without implementation.

- Implementation of the operations (behaviour) is decided by subclasses.
- Abstract class contains only signature of the methods.

Abstract class: defines a common abstract behaviour for its subclasses.



Concrete classes are responsible to implement the behaviour:

- Implement the code needed for the method draw().
- Any other concrete class inheriting from Shape can implement draw().
- All the changes in the code and extending the program with various shapes, is hidden from the user of Shape (any class in association with Shape).
- Shape can be inherited by various shapes and all have Point: no need for repetition.

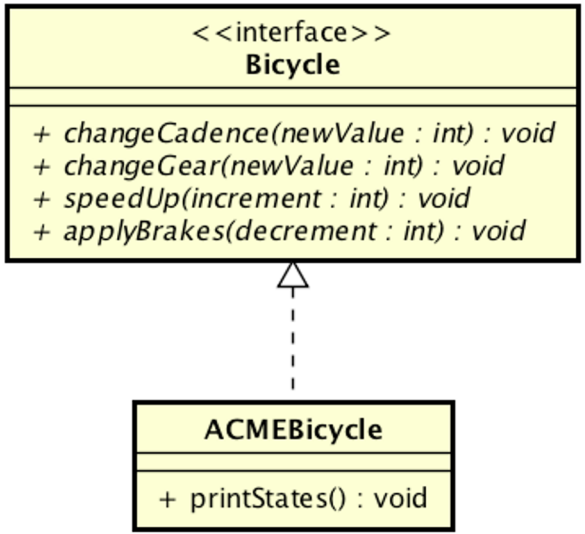
Interfaces and Realization

How this model is implemented in Java?

- Find out here.
  - https://docs.oracle.com/javase/tutorial/java/concepts/interface.html

Users (classes) that are in association with Bicycle remain unaware of all realising classes:

- Extension of the code is easier as long as they adhere to the methods of Bicycle.
- Modifying the implementations of the methods remain safe.
- Interface and realising classes are easier to be reused.



Activity [IN CLASS]

Read the following article:

1. Design Patterns Simplified (click here) [~ 10 min]
   https://medium.com/@Mahmoud_Zalt/software-design-patterns-simplified-8a72232d52b1

Activity [IN CLASS]

Singleton: A “Creational Pattern”

- Read Singleton design pattern below [~ 20 min]:
- General: https://sourcemaking.com/design_patterns/singleton
- In C#: https://www.dofactory.com/net/singleton-design-pattern

Activity [IN CLASS]

Observer: A “Behavioural Pattern”

- Read Observer design pattern below [~ 25 min]:
- General: https://sourcemaking.com/design_patterns/observer
- In C#: https://www.dofactory.com/net/observer-design-pattern

Questions

Part 2:

- What is a design pattern and why do we study them?
- How abstract classes and interfaces are useful in software maintenance?
- What are the categories of design patterns? Name one from each category.
- When do we use Singleton pattern?
- When is it helpful to use Observer pattern?

Exercises

TO DO IN CLASS!

A case: “Project Distribution”

In the assignment we have studied a case study: data set anonymization by experts.
Story: A data owner needs to announce activations of (anonymization) projects to a group of experts. Propose your solution and implement a simple prototype.
Hint:

- Determine the core elements of the problem.
- Choose the best solution (a design pattern).
- Identify the elements of your solution in the problem statement. Use class diagrams.
- Develop a sample code (C#) where it can be seen how the solution would work.

exceed expectations