Introduction to Software Design

Object Oriented Software Engineering
Lecture 6

Dr. Graham McDonald
graham.mcdonald@glasgow.ac.uk

.earning outcomes

- What is Design?
- Discuss the different software design principles
- Broadly distinguish between different design patterns
- Identify anti-patterns
- History of design patterns

Recommended Readings I

- **Design Patterns: Elements of Reusable Object-Oriented Software** By Gamma, Erich; Richard Helm, Ralph Johnson, and John Vlissides (1995). Addison-Wesley. ISBN 0-201-63361-2.
- **Head First Design Patterns** By Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates. First Edition October 2004 ISBN 10: 0-596-00712-4

What is Design?

- The process of envisioning and planning the creation of objects, interactive systems, services, etc.
  - **User-centered**, i.e. users are at the heart of the design thinking approach.
  - About **creating solutions** for people, physical items or abstract systems **to address a need or a problem**.

- Applied arts
- Architecture
- Automotive design
- Biological design
- Communication design
- Configuration design
- Design management
- Engineering design
- Experience design
- Fashion design
- Game design
- Graphic design
- Information architecture
- Information design
- Industrial design
- Instructional design
- Interaction design
- Interior design
- Landscape architecture
- Lighting design
- Modular design
- Motion graphic design
- Organization design
- Product design
- Process design
- Service design
- Social design
- Software design
- Sound design
- Spatial design
- Strategic design
- Systems architecture
- Systems design
- Systems modeling
- Urban design
- User experience design
- Visual design
- Web design

Software Design

Software design is the process of **creating a specification** of a **software artifact**, intended to **accomplish goals**, using a set of **primitive components** and **subject to constraints**.

Software Design Viewpoints

Software design may refer to either:

"all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems"

or

"the activity following requirements specification and before programming"

Design Concepts

Fundamental software design principles
PHAME **P**rinciples:

- **H**ierarchy
- **A**bstraction
- **M**odularisation and
- **E**ncapsulation

Grady Booch (2004). Object-Oriented Analysis and Design with Applications (3rd ed.). MA, USA: Addison Wesley.

Principle of Hierarchy

- Provides you the ability to break a system design into a taxonomical representation through the abstraction of packages and classes.
  - Enables you to understand different aspects of the hierarchie

- A subclass inherits state and behaviour in the form of variables and methods from its superclass and the rest of its ancestors.
  - As you drop down in the hierarchy, the classes become more and more specialised.

Principle of Hierarchy

Hierarchy enables you to look at a problem space and design a solution from the space by creating generalisation and Inheritance relationships between the different objects that define the problem space.

Principle of Abstraction

- Abstraction is the property for which **only the essential details** are **displayed to the user**.
  - The trivial or the non-essentials units are not displayed to the user.
  - A car is viewed as a car rather than its individual components.

Principle of Abstraction

Data Abstraction is the process of identifying only the required characteristics of an object ignoring the irrelevant details.

```
public class Person {
    private Car car;

    public Car getCar() {
        return car;
    }

    public void setCar(Car car) {
        this.car = car;
    }

    public void drive(int speed) {
        car.accelerate(speed);
        //The person does not need to understan
        //acceleration details
    }
}

class Car{
    public void accelerate(int speed) {
        //the low level acceleration details
        //is abstracted into this function
    }
}
```


Principle of Modularisation

‘Modularization consists of dividing a program into modules which can be **compiled separately**, but which have connections with other modules’

Liskov

Principle of Modularisation

'Modularity is the property of a system that has been de-composed into a set of cohesive and loosely coupled modules.'

Grady Booch

- Strive to build modules that are cohesive (by grouping logically related abstractions).
- Aim for loosely coupled software (by minimizing the dependencies among modules).

Principle of Encapsulation

Encapsulation involves bundling data and methods that work on that data within **one unit**, e.g., a class in Java.

Encapsulation provides explicit barriers among different abstractions and thus leads to a clear **separation of concerns**.

Principle of Encapsulation

- Encapsulation **hides the internal representation**, or state, of an object from the outside.
- This is called **information hiding**.

```
class Bank{
    private double balance;

    public double getBalance() {
        auditAccount("Action:getBalance"
                    +balance+"Date:"+new Date(
        return balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
        auditAccount("Action:setBalance"
                    +balance+"Date:"+new Date()
    }

    private void auditAccount(String action) {
        //store action in log
    }
}
```


Principle of Encapsulation

Advantages:

- Encapsulation protects an object from unwanted access by clients.
- Encapsulation allows access to a level without revealing the complex details below that level.
- It reduces human errors.
    - Simplifies the maintenance of the application Makes the application easier to understand.

The SOLID Principles of Software Design

SOLID principles that help software developers design maintainable and extendable classes.

- Single responsibility
- Open-closed,
- Liskov substitution
- Interface segregation and
- Dependency inversion.