Object-Oriented Principles

Lecturer: Adel Vahdati

Moving into Design

- The purpose of the analysis phase is to figure out what the business needs.
- The purpose of the design phase is to decide how to build it.

Object-Oriented Principles

- **Open Closed Principle (OCP)**
  - *Classes should be open for extension but closed for modification.*
- **Liskov Substitution Principle (LSP)**
  - *Subclasses should be substitutable for their base classes.*
- **Dependency Inversion Principle (DIP)**
  - *Depend upon abstractions. Do not depend upon concretions.*
- **Interface Segregation Principle (ISP)**
  - *Many specific interfaces are better than a single, general interface.*
- **Composite Reuse Principle (CRP)**
  - Delegation can be a better alternative to Inheritance.
- **Principle of Least Knowledge (PLK)**
  - For an operation O on a class C, only operations on the following objects should be called: itself, its parameters, objects it creates, or its contained instance objects.

GRASP

- Acronym stands for General Responsibility Assignment Software Patterns.
- Describe fundamental principles for object-oriented design and responsibility assignment, expressed as patterns.

GRASP: Patterns

- Information Expert
- Creator
- Low Coupling
- High Cohesion
- Controller
- Polymorphism
- Indirection
- Pure Fabrication
- Protected Variations

GRASP: Information Expert

- As a general principle of assigning responsibilities to objects, assign a responsibility to the information expert:
  - i.e. the class that has the information necessary to fulfill the responsibility.

GRASP: Creator

- Assign class B the responsibility to create an instance of class A if one or more of the following is true:
  - B aggregates A objects.
  - B contains A objects.
  - B records instances of A objects.
  - B closely uses A objects.
  - B has the initializing data that will be passed to A when it is created (thus B is an Expert with respect to creating A).
- B is a creator of A objects.
- If more than one option applies, prefer a class B which aggregates or contains class A.

GRASP: Low Coupling

- Assign a responsibility so that coupling remains low.
- A class with high (or strong) coupling relies on many other classes.
- Such classes may be undesirable; some suffer from the following problems:
  - Changes in related classes force local changes.
  - Harder to understand in isolation.
  - Harder to reuse because its use requires the additional presence of the classes on which it is dependent.

GRASP: High Cohesion

- Assign a responsibility so that cohesion remains high.
- A class with low cohesion does many unrelated things, or does too much work.
- Such classes are undesirable; they suffer from the following problems:
  - hard to comprehend
  - hard to reuse
  - hard to maintain
  - Delicate: constantly affected by change

GRASP: Controller

- Assign the responsibility for receiving or handling a system event message to a class representing one of the following choices:
  - Represents the overall system, device, or subsystem *(facade controller)*.
  - Represents a use case scenario within which the system event occurs (a use-case- or session-controller).

GRASP: Polymorphism

- When related alternatives or behaviors vary by type (class), assign responsibility for the behavior — using polymorphic operations — to the types for which the behavior varies.
- Define the behavior in a common base class or, preferably, in an interface.

GRASP: Indirection

- Assign the responsibility to an intermediate object to mediate between other components or services so that they are not directly coupled.
- The intermediary creates an *indirection* between the other components.
- Beware of transitive visibility.

GRASP: Pure Fabrication

- Assign a highly cohesive set of responsibilities to an artificial or convenience class that does not represent a problem domain concept — something made up, to support high cohesion, low coupling, and reuse.
- Example: a class that is solely responsible for saving objects in some kind of persistent storage medium, such as a relational database

GRASP: Protected Variations

- Identify points of predicted variation or instability; assign responsibilities to create a stable interface around them.

References

- Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development,* 3rd Ed. Prentice-Hall, 2004.
- Ramsin, Raman. "Home." Department of Computer Science and Engineering, Sharif University of Technology. Accessed February 15, 2025. https://sharif.edu/~ramsin/index.htm.