UNIT II DESIGN PATTERNS

GRASP: Designing objects with responsibilities – Creator – Information expert – Low Coupling – High Cohesion – Controller - Design Patterns – creational - factory method - structural – Bridge – Adapter - behavioral – Strategy – observer

General Responsibility Assignment Software Patterns (GRASP)

Guidelines for assigning responsibility

to classes and objects

in object-oriented design.

Introduction

- GRASP is created by Craig Larman
  - To encompass nine OO design principles related to creating responsibilities for classes
- GRASP = General Responsibility Assignment S/w Patterns (or Principles)
- Name chosen to suggest the importance of
  - **Grasp**ing the fundamental principles of OO design & responsibility assignment, expressed as patterns
- These principles can be viewed as design patterns
  - It offer benefits similar to the classic "Gang of Four" patterns

GRASP

- This approach to understand and using design principles is
  - Based on patterns of assigning responsibilities
- All patterns ideally have suggestive names
- GRASP patterns communicate
  - **Fundamental principles of responsibility assignment in object design**

Responsibility-Driven Design (RDD)

- GRASP patterns are used to assign responsibility to objects
- Their use results in a RDD for Object Orientation (OO)
  - Contrast to (the more traditional) Data-Driven Design
- With this point of view,
  - Assigning responsibilities to objects is a large part of basic object design

Responsibilities

- GRASP is a learning aid for OO design with responsibilities
  - **Responsibilities are assigned to classes of objects during object design.**

- UML defines a responsibility as "a contract or obligation of a classifier"

- These responsibilities are of the following two types:

  1. Doing responsibilities of an object
  2. Knowing responsibilities of an object

Types of Responsibility

- Doing responsibilities of an object include:
  - Doing something itself, such as creating an object or doing a calculation
  - Initiating action in other objects
  - Controlling and coordinating activities in other objects

- Knowing responsibilities of an object include:
  - Knowing about private encapsulated data
  - Knowing about related objects
  - Knowing about things it can derive or calculate

Responsibility in NextGen POS

- Doing Responsibility
  - **“A Sale is responsible for creating SalesLineItems"**
- Knowing Responsibility
  - **“A Sale is responsible for knowing its total"**

Methods in Assigning Responsibilities

- Responsibilities are implemented using methods
  - That either act alone or collaborate with other methods and objects.
- Example:
  - Sale class define a method getTotal() to know its total;
  - To fulfill that responsibility, object of Sale may collaborate with other objects, such as sending agetSubtotal() to each SalesLineItem object asking for its subtotal.
- UML Interaction diagram helps to identify
  - The assignment of responsibilities to software classes

GRASP patterns

1. Creator
2. Information Expert
3. Low Coupling
4. Controller
5. High Cohesion
6. Polymorphism
7. Indirection
8. Pure Fabrication
9. Protected Variation

- All patterns answer some software problem,
  - These problems are common to almost every s/w development project.

What to look for in GRASP patterns

- Low Coupling
  - Reducing the impact of change
- High Cohesion
  - Similar functionality in the same place
- Low representational gap (LRP)
  - Minimal conceptual gaps