

Today

- Continue discussing *design*
- Note: book thinks we are still in elaboration iteration #1
- In fact we are in #2 since we did some implementation
- Today's reference: text chapter 16
- Today's key term: GRASP

The GRASP Concept

- **GRASP** stands for
 - General
 - Responsibility
 - Assignment
 - Software
 - Patterns
- The mnemonic idea is, successful OOA/D requires "grasping" the GRASP principles
 - What do the parts mean?

GRASP Concept (cont.)

General Responsibility Assignment Software Patterns

- General = _____
- Responsibility = _____
- Assignment = _____
- Software = _____
- Patterns = _____

GRASP Concept (cont.)

General Responsibility Assignment Software Patterns

- General = Abstract; **widely applicable**
- Responsibility = **Obligations**, duties
- Assignment = **Giving a responsibility to a module**
- Software = **Computer code**
- Patterns = Regularities, templates, **abstractions**
- How would you parenthesize the GRASP phrase?
- Let's look at these terms in more detail

GRASP Responsibilities

- Responsibilities – obligations (of an object)
 - Responsibilities (obligations) have 2 types
 - What an object is responsible for **doing**
 - What are some "doing"-type responsibilities?
 - What an object is responsible for **knowing**
 - What are some "knowing"-type responsibilities?

GRASP: "Doing"-Type Responsibilities

- 1. Perform a directly useful action
 - Change some data
 - (Assign, calculate, create an object,...)
- 2. Initiate actions in other objects
 - What do those objects do?
- 3. Control/coordinate other objects
 - They might change some data
 - If they don't, what *do* they do?
- Analogy: top brass, middle brass, and everyone else

Example of a Responsibility

- See Figure 16.1
- Objects of class Sale...
 - have the responsibility of making payments
 - this is indicated by the incoming message
 - this responsibility is handled by what method?
 - to meet the responsibility does it do 1, 2, or 3?

GRASP: "Knowing"-Type Responsibilities

- Have member data
 - I have a working computer, paper, (no pen/pencil)
 - Note how useful this information is in class...
- Know what you (an object) can do yourself
 - Calculate something
 - Create other objects
 - What parts of an object hold this information?
- Understand related objects
 - What use is that?

Another Example...

- Consider the class containing main(...)
- Does this class have responsibility of type 1, 2, or 3?

GRASP Responsibilities: Knowing and Doing

- | | |
|---|---|
| <ul style="list-style-type: none">■ 1. Understand member data<ul style="list-style-type: none">■ I have a working computer, paper, (no pen/pencil)■ Note how useful this information is in class...■ 2. Understand what you (an object) can do yourself<ul style="list-style-type: none">■ Calculate something■ Create other objects■ 3. Understand interactions among related objects<ul style="list-style-type: none">■ What use is that? | <ul style="list-style-type: none">■ 1. Perform a directly useful action<ul style="list-style-type: none">■ Change some data<ul style="list-style-type: none">■ (Assign, calculate, create an object,...)■ 2. Initiate actions in other objects<ul style="list-style-type: none">■ What do those objects do?■ 3. Control/coordinate other objects<ul style="list-style-type: none">■ They might change some data■ If they don't, what do they do?■ Analogy: top brass, middle brass, and everyone else |
|---|---|

Match blue dots on right to blue dots on left!

Object Responsibilities

- Notice where this is going:
 - Design the objects, including
 - Their **data**
 - Their **methods**
 - Their **interactions** with (messages to) other objects
 - Which of these corresponds to 1, 2, & 3?
- 1. Understand member data
- 2. Understand what you (an object) can do yourself
- 3. Understand related objects

Responsibilities and Interaction Diagrams

- Review:
 - Two kinds of interaction diagrams
 - *Collaboration Diagrams*
 - *Sequence Diagrams*
 - Also recall *Class Diagrams*
 - See Figures
- These help greatly in guiding responsibility assignment

Responsibilities and Interaction Diagrams

- Two kinds of interaction diagrams
 - Collaboration Diagrams; Sequence Diagrams
 - Are these better for designing
 - "knowing"-type responsibilities, or
 - "doing"-type responsibilities?
 - What about Class Diagrams?

Responsibilities and Interaction Diagrams

- Three kinds of diagrams
 - Collaboration Diagrams; Sequence Diagrams; Class Diagrams
 - Which is better for which "doing"-type responsibilities?
- "Doing"-type responsibilities:
 - (try this 3rd) Perform a directly useful action
 - (try this 1st) Initiate actions in other objects
 - (try this 2nd) Control/coordinate other objects

Responsibilities and Interaction Diagrams II

- Three kinds of diagrams
 - Collaboration Diagrams; Sequence Diagrams; Class Diagrams
 - Which is better for which "knowing"-type responsibilities?
- "Knowing"-type responsibilities:
 - Have member data
 - Know what you (an object) can do yourself
 - Understand related objects
- Which part of the diagram type is best for which?

GRASP: Software Patterns

- **Software Pattern:** *a general solution strategy for a general kind of problem*
 - If you can categorize the problem...
 - Then, you can apply the right software pattern
- Patterns should have names
 - ...to make them easier to think/talk about
- A Pattern should come with
 - Suggestions for how to apply it
 - Explanation of its trade-offs

GRASP Software Patterns: the Shoulds

- Consider analogies to medicines and cars
- Patterns should have names
 - What is the medicine analogy? Car analogy?
- Patterns should come with suggestions for how to apply them
 - What is the medicine analogy? Car analogy?
- Patterns should come with explanations of trade-offs
 - What is the car analogy? Medicine analogy?

A Few GRASP Patterns

- Perhaps the most important are:
 - High Cohesion pattern
 - Low Coupling pattern
 - Information Expert pattern
 - Creator pattern
 - Controller pattern



GRASP Patterns

- High Cohesion pattern
- Applying this pattern **assigns** responsibilities so that parts that interact a lot are grouped together
- This tends to minimize system complexity
- Is global data a way to high cohesion or low cohesion?
- Why does Java not have global data??



GRASP Patterns

- Low Coupling pattern
- Applying this pattern **assigns** responsibilities to minimize interaction
- Is global data a way to high coupling or low coupling?
- Why does Java not have global data??



GRASP Patterns

- Information Expert pattern
 - Applying this solution strategy results in part of the design
 - Assign information and computations on that information to the right classes
 - Example: consider Sale, SalesLineItem, and ItemNumber classes
 - Which should be *assigned responsibility* for getting the grand total?
 - Asking/answering this question = applying the information expert pattern



GRASP Patterns

- Creator pattern
 - **Assign** responsibilities to objects, per...
 - which objects should create which other objects?
 - This is important in OO systems
 - Answering the question is applying the creator pattern
 - Example: Sale, SalesLineItem, and ItemNum classes
 - Which should be assigned the responsibility of...
 - creating new SalesLineItems?



GRASP Patterns

- Controller pattern
- **Assigns** responsibility for handling input events to appropriate objects
- Example: cashier login event
- Which class or object should handle that?