Class Relationships

November 2, 2007

No Class on Monday Nov 5
• Due to night exam last week (midterm 2)

Final Exam
• When:    Thursday, December 13, 2007
• Where:   Curtiss Hall, room 127 (classroom)
• Time:    4:30-6:30pm

• Also see this page (for large classes like ComS 207):
  • http://www.iastate.edu/~registrar/exams/groups.shtml

• NOTE: The time listed on this page
  does **NOT** apply to ComS207!!!
  • http://www.iastate.edu/~registrar/exams/

Example: Sudoku_Solver.java

Solving Sudoku Puzzles With Recursion
(http://www.websudoku.com/)

Rule #1: 1..9 must be in each row
Sample that satisfies rule #1

Rule #2: 1..9 must be in each column

Sample that satisfies rules #1 and #2

Rule #3: 1..9 must be in each 3x3 window

Sample that satisfies rules #1, #2, and #3

Quick Review of Last Lecture
Visibility Modifiers

<table>
<thead>
<tr>
<th>Variables</th>
<th>public</th>
<th>private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Violate encapsulation</td>
<td>Enforce encapsulation</td>
<td></td>
</tr>
</tbody>
</table>

| Methods | Provide services to clients | Support other methods in the class |

The static Modifier

- We declare static methods and variables using the static modifier
- It associates the method or variable with the class rather than with an object of that class
- Static methods are sometimes called class methods and static variables are sometimes called class variables

Static Variables

- Normally, each object has its own data space, but if a variable is declared as static, only one copy of the variable exists

```java
private static float price;
```

- Memory space for a static variable is created when the class is first referenced
- All objects instantiated from the class share its static variables
- Changing the value of a static variable in one object changes it for all others

Classes

- A class can contain data declarations and method declarations

Objects – instances of classes

- Object obj1:
  - int size = 5;
  - int weight = 170;

- Object obj2:
  - int size = 10;
  - int weight = 130;

Note that the variables can have different values in the two objects
Classes

• Things change if we declare a static variable

```
static int size;
int weight;
```

Objects – instances of a class with a static variable ‘size’

```
obj1
int weight = 170;
static int size;

obj2
int weight = 130;
```

Objects – instances of classes

• Note that the variables can have different values in the two objects

```
int size = 5;
int weight = 170;

int size = 10;
int weight = 130;
```

Static Class Members

• The order of the modifiers can be interchanged, but by convention visibility modifiers come first
  • Recall that the main method is static – it is invoked by the Java interpreter without creating an object
  • Static methods cannot reference instance variables because instance variables don’t exist until an object exists
  • However, a static method can reference static variables or local variables

Method Control Flow

• If the called method is in the same class, only the method name is needed

```
compute
myMethod;
```

Accessing Variables

• If the called method is in the same class, only the method name is needed

```
compute
myMethod[];
```

```
myVariable = 5;
OK
```
Accessing Variables

- Static methods cannot use non-static class variables.

```
myMethod();  // Error

int myVariable;
myVariable = 5;  // Error
```

Accessing Variables

- Static methods can use static class variables.

```
static int myVariable;
myVariable = 5;  // OK
```

Static Class Members

- Recall that a static method is one that can be invoked through its class name.
- For example, the methods of the Math class are static:
  ```
  result = Math.sqrt(25);
  ```
- Variables can be static as well.
- Determining if a method or variable should be static is an important design decision.

```
class Helper {
    public static int cube(int num) {
        return num * num * num;
    }
}
```

Because it is declared as static, the method can be invoked as:
```
value = Helper.cube(5);
```

Method Control Flow

- Static methods can only call other static methods within the same class.

```
main() {
    Helper.cube();
    helpMe();
}
```

Chapter 6
Section 6.4
Class Relationships

- Classes in a software system can have various types of relationships to each other
- Three of the most common relationships:
  - Dependency: A uses B
  - Aggregation: A has-a B
  - Inheritance: A is-a B

Dependency Example: Client-Server

- A dependency exists when one class relies on another in some way, usually by invoking the methods of the other
- We’ve seen dependencies in many previous examples
- We don’t want numerous or complex dependencies among classes
- Nor do we want complex classes that don’t depend on others
- A good design strikes the right balance

Dependency

- Some dependencies occur between objects of the same class
- A method of the class may accept an object of the same class as a parameter
- For example, the concat method of the String class takes as a parameter another String object
  \[ \text{str3 = str1.concat(str2);} \]
- This drives home the idea that the service is being requested from a particular object

Concatenation Example

- The following example defines a class called Rational to represent a rational number
- A rational number is a value that can be represented as the ratio of two integers
- Some methods of the Rational class accept another Rational object as a parameter
- See RationalTester.java (page 297)
- See Rational.java (page 299)
Aggregation

- An aggregate is an object that is made up of other objects
- Therefore aggregation is a *has-a* relationship
  - A car *has a* chassis
  - A student *has an* address

Aggregation Example: Components of a Car

Chasis

Tyres

Steering Wheel

What type of Steering Wheel?
Aggregation

- In software, an aggregate object contains references to other objects as instance data
- The aggregate object is defined in part by the objects that make it up
- This is a special kind of dependency – the aggregate usually relies on the objects that compose it

Aggregation Example: Components of a Student

Student

First Name
Last Name
Home Address
School Address

john

John
Smith
21 Jump Street
800 Lancaster Ave.

marsha

Marsha
Jones
123 Main Street
800 Lancaster Ave.

© 2004 Pearson Addison-Wesley. All rights reserved

© 2004 Pearson Addison-Wesley. All rights reserved
Aggregation in UML

- StudentBody
  - main (args : String[]): void
- Student
  - firstName : String
  - lastName : String
  - homeAddress : Address
  - schoolAddress : Address
  - toString() : String
- Address
  - streetAddress : String
  - city : String
  - state : String
  - zipCode : long
  - toString() : String

Aggregation

- In the following example, a Student object is composed, in part, of Address objects.
- A student has an address (in fact each student has two addresses).
- See StudentBody.java (page 304)
- See Student.java (page 306)
- See Address.java (page 307)
- An aggregation association is shown in a UML class diagram using an open diamond at the aggregate end.

THE END