Lecture 7: Design and Classes

Last time:
1. More assignment operators
2. Precedence and short-circuiting
3. Type casting

Today:
1. Basics of program design
2. Pseudo code
3. Objects and classes

The Software Lifecycle

Requirements → Design → Coding → Testing → Deployment → Maintenance → Evolution

In the Real World, Requirements and Design Rule

- Getting requirements right is essential for successful projects
  - FBI electronic case file (junked after $180m)
  - IRS system upgrade in late 90s (junked after >$2bn)
  - FAA air-traffic control (false starts, >$10bn spent)
- Good design makes other parts of lifecycle easier
- In “the real world” coding typically < 30% of total project costs
Program Design

- There are many aspects to good design
  - Architecture
  - Modeling
  - Requirements decomposition
  - Pseudo-code
- In this class we will focus on latter

What Is “Pseudo-code”?%

- When developing a complex part of a program (an algorithm), one of the tools often useful is pseudo-code.
- It’s not English, not programming language -- somewhere between.
- Captures the flow of the program without worrying about language-specific details.

Example:

- Requirement: email program that allows you to send a message either to one person, or to your whole address book.
- Pseudo-code:
  - prompt "Enter message: ":
  - input message
  - prompt "Send to whole address book?":
  - input answer
  - if answer == "no"
    - prompt "Enter recipient: ":
    - input recipient
    - send message to recipient
  - otherwise
    - for each recipient, r, in address book
      - send message to r
What Is Pseudo-Code? (cont.)

- NOT English
- NOT a program
- Something in-between
  - Captures the “logic” and “flow” of the algorithm
  - Note that pseudo-code could be translated into ANY programming language (not just Java)
- Good programming practice
  - Write pseudo-code first and keep it as your design
  - Include it as comments in your code to help you connect code to design

Testing

- Some testing is done by customer (acceptance testing)
  - E.g. testing we do on your projects!
  - You want to avoid errors surfacing during acceptance testing
- How to avoid errors during acceptance testing?
  - Test thoroughly before release
  - Cover all cases in code (if/else branches, etc.)
  - Identify “corner cases” (extreme values of inputs) and test with these
- We will study testing more later in semester

Questions

- What is System in System.out.println()?
- Why use str.equals("cat") to compare equality of String str and “cat“?
- Is the similarity of the notations System.out.println() str.equals() sc.nextInt() important, or coincidental?
Answers

- ... will come, but are based on fact that Java is an object-oriented (OO) language
- Other OO languages: C++, Smalltalk, Eiffel
- What is an “object”?
- What does it mean for a language to be object-oriented?

Objects

- Bundles of (related)
  - data ("state")
  - operations ("behavior")
- Data often referred to as instance variables
- Operations usually called methods
- Invoking operations can change state (values stored in instance variables)

Example of Objects

- Bank account
  - State: account number, type, owner, balance, etc.
  - Operations: deposit, withdrawal, transfer, balance inquiry, etc.
- Student
  - State: name, ID, date of birth, major,...
  - Operations: lookup grades, lookup class, compute tuition, compute age, ...
- Scanner
  - State: stream to read from (e.g. System.in)
  - Operations: get an int, double, byte, ...
- String
  - State: sequence of characters
  - Operations: comparison test, equality test, (much more), ...
Sample Student Object

<table>
<thead>
<tr>
<th>State</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>getAge</td>
</tr>
<tr>
<td>ID</td>
<td>getGrades</td>
</tr>
<tr>
<td>DOB</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td></td>
</tr>
</tbody>
</table>

Accessing State / Methods

- If o is an object
  - v is an instance variable of the object
  - m is a method of the object
- Then o.v is how to access the data in v
- o.m is how to invoke m
- So
  - System is an object, with out an instance variable
  - out is also an object, with println a method
  - System.out.println is how to access this method!
- Suppose str is a String
  - str is an object!
  - Methods of this object: equals, compareTo, etc.
- str.equals, str.compareTo, etc. invokes these methods on that object

Object-Oriented Programming

- Programs are collections of interacting objects
- Writing programs involves identifying what the objects should be and programming them
- Object-oriented languages provide features to ease object-oriented programming
- Defining objects involves indentifying
  - state
  - methods
More Questions

- How do you create objects?
- Do objects have “types”?
- Answer: classes

Classes

- “Blueprints” (“templates”) for objects
- Classes include specifications of
  - Instance variables (including types, etc.) to include in objects
  - Implementations of methods to include in objects
- Classes can include other information also, as will be seen later
  - static methods / instance variables
  - public / private methods, instance variables
  - And so on

Student Class Example

Conceptually:
- Instance variables:
  - String name
  - int ID
  - int dateOfBirth
  - String major
- Methods
  - getAge
  - getGrades
  - etc.
- The actual class implementation will include code for the methods
- This describes a blueprint for student objects