More Class Design

Lecture 10, Wed Jan 27 2010

borrowing from slides by Paul Carter and Wolfgang Heidrich

http://www.cs.ubc.ca/~tmm/courses/111-10
Reading Assignments

- Chapter 3
Review: Random Numbers

- **Random** class in `java.util` package
  - `public Random()`
    - Constructor
  - `public float nextFloat()`
    - Returns random number between 0.0 (inclusive) and 1.0 (exclusive)
  - `public int nextInt()`
    - Returns random integer ranging over all possible int values
  - `public int nextInt(int num)`
    - Returns random integer in range 0 to (num-1)
Review: return Statement

■ Use the return statement to specify the return value when implementing a method:

```java
int addTwoInts (int a, int b) {
    return a+b;
}
```

■ Syntax: return expression;

■ The method stops executing at that point and “returns” to caller.
Review: Tester Classes

- **Die** class has no main method.

- Best is to write another class that instantiates some objects of your new class and tries them out.
  - Sometimes called a “tester” or “testbench”
Implementing Die and RollDice

- first pass
- testing
- refining
Information Hiding

- Hide fields from client programmer
  - maintain their integrity
  - allow us flexibility to change them without affecting code written by client programmer

- Parnas' Law:
  - "Only what is hidden can be changed without risk."
Public vs Private

- **public** keyword indicates that something can be referenced from outside object
  - can be seen/used by client programmer
- **private** keyword indicates that something cannot be referenced from outside object
  - cannot be seen/used by client programmer
- Let’s fill in public/private for *Die* class
Public vs. Private Example

public class Die {
    ...
    public int roll() {
        ...
        private void cheat(int nextRoll) {
            ...
        }
    }
}
Public vs. Private Example

```java
Die myDie = new Die();

int result = myDie.roll();  // OK
myDie.cheat(6);             // not allowed!
```