CSCI 136
Data Structures &
Advanced Programming

Jeannie Albrecht

Lecture 2

Feb 10, 2014

Administrative Details

- Lab 1 handout/PDF
- Prelab (should be completed before lab):
  - Lab 1 design doc
  - Use Boggle design doc as model - no real code!
- TA hours start on Wed

2

Last Time

- Hello.java
  - Write a program that prints “Hello” to the terminal

3

Hello.java

```
/*
 * This program prints out a message to the terminal.
 */
public class Hello {

    // Just print a message.  Nothing complicated here...
    public static void main(String args[]) {
        System.out.println("Hello.");
    }
}
```

4

Today’s Outline

- Continue Java refresher
  - Sum.java
    - Write a program that adds two integers together and returns the sum
    - Use command-line args and Scanner
  - Object-Oriented Program (OOP) Design
    - Basic concepts
    - Java-specific features

5

Sum1.java

```
/*
 * A program to add together two numbers from command line args.
 */
public class Sum1 {

    public static void main(String args[]) {
        int n = Integer.valueOf(args[0]);
        int n2 = Integer.valueOf(args[1]);
        System.out.println("Answer is " + (n+n2));
    }
}
```

6

Sum2.java

```
import java.util.Scanner;

/*
 * A program to add together two numbers from the terminal.
 */
public class Sum2 {

    // Create a new Scanner, read in two integers, and print their sum.
    public static void main(String args[]) {

        // create a new scanner for the terminal input
        Scanner in = new Scanner(System.in);

        System.out.print("Give me a number: ");
        int n = in.nextInt();
        System.out.print("Give me another number: ");
        int n2 = in.nextInt();

        System.out.println("Answer is " + (n + n2));
    }
}
```

7

Object-Oriented Programming

- Objects are building blocks of software
- Programs are collections of objects
  - Cooperate to complete tasks
  - Represent “state” of the program
  - Communicate by sending messages to each other

8

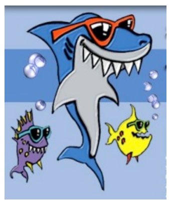
Object-Oriented Programming

- Objects can model:
  - Physical items - Dice, board, dictionary
  - Concepts - Date, time, words, relationships
  - Processing - Sort, search, simulate
- Objects contain:
  - Properties (instance variables)
    - Attributes, relationships to other objects, components
      - Letter value, grid of letters, number of words
  - Capabilities (methods)
    - Accessor and mutator methods
      - addWord, lookupWord, removeWord

9

Sharks and Minnows

- Let’s look at an example: WaTor
- What objects are being modeled?
  - Physical items
  - Concepts
  - Processing
- The shark object contains:
  - Properties
  - Capabilities



10

Next Up: Implementing a Card Object

- Think before we code!
- Start general.
  - Build an *interface* that advertises all public features of a card
  - Not an implementation (define methods, but don’t include code)
- Then get specific.
  - Build specific implementation of a card using our general card interface

11

Start General: CardInterface

- What data do we have to represent?
  - Properties of cards
  - How can we represent these properties?
- What methods do we need?
  - Capabilities of cards
  - Do we need *accessor* and *mutator* methods?

*

12