CS100M
Introduction to Computer Programming

Spring 2004
Lecture 18
Methods

Announcements

- Prelim 2 leftovers in Carpenter (wait for announcement)
- Prelim 1 regrades done
- A4 reminder
- DIS appts policy enhancement (see website)
- Java:
  - reminder: OK to use any IDE
  - can't get DrJava to work? Use command line for now (reminder: see E7!)

Overview

- programming reminder:
  - find nouns
  - find verbs
- nouns in OOP deal with variables and objects
- verbs in OOP pretty much the same as in MATLAB
  - operators
  - methods
- methods reminder
  - define your own actions
  - must be inside a class
- for now, putting all methods in same class for convenience

Method Syntax

```
modifiers returntype name(params) stuff
  block
```

modifers
- privacy
- static or non-static

returntype
- must be valid type: class or primitive
- may be void (no return)

name
- must be valid Java identifier
- usually start with lowercase letter

params
- arguments to the method
- must be declared

stuff
- more things involving OOP

Why Methods?

```
public class Methods0 {

  public static int myRand(int low, int high) {
    if (low > high) {
      System.out.println("myRand Failure!");
      System.exit(0);
    }
    return (int) (Math.random()*(high-low+1)) + (int) low;
  }

}
```


Placement and Scope

- methods must be inside a class!
- all methods in a class can call each other...generally,
  - write your methods in any order
  - static methods call static methods (no objects)
  - non-static methods call non-static methods (objects)
- reminders about scope:
  - each method treated as an enclosing block (including params)
  - consequence: all params and declared local variables inside the method are visible ONLY inside the method

Example

```
public class Methods1 {

  public static void test3() {
      System.out.println("test3");
  }

  public static void main (String[] args) {
    test1();
    test2();
    test3();
    // test4();
    // Methods1.test4();
  }

  public static void test2() {
    System.out.println("test2");
  }

  public static void test1() {
    System.out.println("test1");
  }

  public void test4() {
    System.out.println("Can you make this print?");
  }

}
```


More Notes

Params and Return Type

- parameters:
  - Java is pass by value!
  - caller sends values of passed variables, not the variables themselves!
- don't want to return anything?
  - use void method
  - you may use a return statement anywhere (or not)
- want to return something?
  - method must have a type
  - return statement expression type must match the method's type

9

Example

```
public class Methods2 {
  public static void main (String[] args) {

    System.out.println( and(true,false) );
    boolean t1 = nand(true,false);
    boolean t2 = xor(true,false);
    System.out.println( and(t1,t2) );
  }

  // Return short-circuiting AND of t1,t2:
  public static boolean and(boolean t1, boolean t2) {
    return t1 && t2;
  }

  // Return exclusive OR of t1,t2:
  public static boolean xor(boolean t1, boolean t2) {
    return t1 != t2;
  }

  // Return NOT-AND of t1,t2:
  public static boolean nand(boolean t1, boolean t2) {
    return !(t1 & t2);
  }

}
```

10

More Notes

11

Another Example

```
public class Methods3 {
  public static void main (String[] args) {
    int[] x = {1, 4, 5, -1};
    search1(x,-1);
    System.out.println(search2(x,-1));
  }

  // linear search for target in data x, return early if found:
  public static void search1(int[] x, int target) {
    for (int i = 0; i <= x.length; i++)
      if (x[i] == target) {
        System.out.println("Success!");
        return;
      }
    System.out.println("Fail!");
  }


  // linear search for target in data x,
  // return true if found, else false:
  public static boolean search2(int[] x, int target) {
    for (int i = 0; i <= x.length; i++)
      if (x[i] == target)
        return true;
    return false;
  }

}
```

12

Overloading

- can write more than one method with the same name in same class
- why?
  - consider **`System.out.println`**
  - you can call **`println`** with doubles, ints, booleans....
  - see Java API for descriptions: **http://java.sun.com/j2se/1.4.2/docs/api/java/io/PrintStream.html#println()**
- rules:
  - you may change order of arguments, types, number of params and combinations of these changes
  - these do not constitute overloading:
    - changing just the return type
    - changing just the param names

Example

```
public class Methods4 {
  public static void main(String[] args) {
    System.out.println("int:  "+myRand(1,10));
    System.out.println("bit:  "+myRand());
    System.out.println("char: "+myRand('a','z'));
  }

  // Return random int, low <= high:
  public static int myRand(int low, int high) {
    if (low > high) {
      System.out.println("myRand Failure!");
      System.exit(0);
    }
    return (int) (Math.random()*(high-low+1)) + (int) low;
  }

  // Return random bit:
  public static boolean myRand() {
    return 1 == (int) (Math.random()*2);
  }

  // Return random letter, c1 <= c2:
  public static char myRand(char c1, char c2) {
    return (char) myRand((int)c1,(int)c2);
  }

}
```


More Notes

Suggested Exercises

- Write a version of **`myRand`** for doubles. Account for roundoff error (an "eps") when considering if **`low`** > **`high`**.
- Write a program that computes arithmetic expressions as functions. So, you will need methods for add, sub, div, mul, and neg. Demonstrate how your program works.
- Write a method that mimics the behavior of the modulus operator.
- Write a class that has a collection of useful String handling methods: convert to uppercase, convert to lowercase, remove all non-alphanumeric characters, encrypts by shifting each character by 1 letter.