Agenda: Discussion Week 8
May 18, 2009

• Discussion section evaluations
• Follow-up from last week’s discussion
  – Static vs. instance methods
  – Ragged arrays
• PSA 4
  – Palindrome
  – Dictionary
  – Fractals
• Recursion examples
• Suggestions?
Static vs. Instance: Variables

• Instance variables (default)
  – Associated with an object
  – Each object has distinct copies of instance variables

• Static variables (aka “class variables”)
  – Associated with the class, rather than with any object
  – Every instance of the class shares a class variable, which is in one fixed location in memory

• Reference
Static vs. Instance: Methods

- **Instance methods**
  - Default: `public void instanceMethod() {}`
  - Associated with an object
  - Can use the instance variables of that object

- **Static methods**
  - Use `static` keyword: `public static void staticMethod() {}`
  - Associated with the class, rather than with any object
  - Can’t use the instance variables of any object of the class
2-D arrays: Rectangular

- Rectangular arrays
  - have same number of elements in each row of a 2-D array

```java
int rows = 3;
int columns = 2;
String[][] rectangular2dArray = new String[rows][columns];
for (int r = 0; r < rows; r++) {
    for (int c = 0; c < columns; c++) {
        rectangular2dArray[r][c] = "(" + r + "," + c + ")";
    }
}
```

<table>
<thead>
<tr>
<th>Rows</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>(0,1)</td>
</tr>
<tr>
<td>(1,0)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(2,0)</td>
<td>(2,1)</td>
</tr>
</tbody>
</table>
2-D arrays: Ragged

- Ragged arrays
  - can have a different number of elements in each row of a 2-D array

```
int rows = 3;
String[][] ragged2dArray = new String[rows][];
for (int r = 0; r < rows; r++) {
    int columns = r + 1;
    ragged2dArray[r] = new String[columns];
    for (int c = 0; c < columns; c++) {
        ragged2dArray[r][c] = "(" + r + "," + c + ")";
    }
}
```

<table>
<thead>
<tr>
<th>Rows</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td></td>
</tr>
<tr>
<td>(1,0)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(2,0)</td>
<td>(2,1)</td>
</tr>
</tbody>
</table>
Recursion

• Method reuse
  – Base case: smallest part with known solution
  – Recursive case: break problem into smaller parts

• Loops and recursion
  – All recursive code can be implemented with loops
  – Looping: aka “iteration”
PSA 4: Palindrome

• Check if a string is a palindrome (E.g., “racecar”)
• Recursion
  – Base case
    • If string is 1 character long or empty, return true
  – Recursive case
    • If first and last letters are equal, make recursive call on string with those letters removed
    • Otherwise, return false
• Looping
  – Solve same problem with looping
• Extra credit: ignore white space, punctuation
PSA 4: Dictionary (1)

• Use binary search to look up word in dictionary

• Recursion:
  – Base case
    • If \( left > right \), word is not in dictionary, so return -1
    • If size of section is 1, check if word to lookup matches word in section (if size of section is 1, \( left == right == mid \))
      – If words match, return index \( (row \) in dictionary 2-D array)
      – Otherwise word is not in dictionary, so return -1
PSA 4: Dictionary (2)

– Recursive case
  • When size of section is > 1, calculate mid-point of section
    – If word to look up matches word at mid-point, return mid-point
    – If word to look up comes before word at mid-point, search before mid-point (i.e., look left)
    – If word to look up comes after word at mid-point, search after mid-point (i.e., look right)
PSA 4: Dragon Fractal

• **Recursion:** `draw(int depth, Point p1, Point p2, String direction)`
  
  – **Base case**
    • If depth is 0, draw a line between points p1 and p2, and then sleep
  
  – **Recursive case**
    • Make a left or right turn
      – Calculate x and y coordinates of new point newP
      – Make a right recursive call to draw from p1 to newP
      – Make a left recursive call to draw from newP to p2
PSA 4: Colored Dragon Fractal

• Extra credit: Add color to Dragon Fractal
• Options:
  – Draw example
  -OR-
  – Design your own
Recursion examples

• Counting
  – Up: 1, 2, 3, ..., n
  – Down: n, n-1, n-2, ..., 1
• Sum of arithmetic sequence (discussion 7): 1 + 2 + 3 + ... + n
• Factorial: 1 * 2 * 3 * ... * n
• Fibonacci: 1 + 1 + 2 + 3 + 5 + ... + Fib(n - 1) + Fib(n - 2)
• Linear search
  – Example: Search for 2 in array containing 1, 2, 3
• Binary search
  – Example: Look up word in dictionary, number in telephone book
• Print triangle of stars
  *
  **
  ***
Recursion: Counting

• count(n):
  – If n > 1, count(n – 1) and print n
• Up: 1, 2, 3, …, n
• Down: n, n-1, n-2, …, 1
• Does print statement go before or after recursive call?
Recursion: Sum of arithmetic sequence

• $1 + 2 + 3 + \ldots + n$
• $\text{sumSequence}(n) =$
  – 1 if $(n == 1)$
  – Otherwise $n + \text{sumSequence}(n - 1)$
Recursion: Factorial

• $1 \times 2 \times 3 \times \ldots \times n$
• \( \text{Factorial}(n) = \)
  – 1 if (n == 1)
  – Otherwise n \times \text{Factorial}(n - 1)
Recursion: Fibonacci

• \(1 + 1 + 2 + 3 + 5 + \ldots + \text{Fib}(n - 1) + \text{Fib}(n - 2)\)
• \(\text{Fib}(n) = \)
  – 1 if \((n \leq 2)\)
  – Otherwise \(\text{Fib}(n - 1) + \text{Fib}(n - 2)\)
Recursion: Linear search

- Example: Search for 2 in array containing 1, 2, 3
- linearSearch(double[] a, double toFind, int lastPos)
  - If element at position lastPos in double array a equals toFind, return lastPos
  - If there are no more elements in array to search, return -1
  - Otherwise search in array up to element before lastPos:
    linearSearch(double[] a, double toFind, lastPos – 1)
Recursion: Binary search

- Example: Look up word in dictionary, number in telephone book
- binarySearch(double[] a, double toFind, int left, int right)
- if (n == 1)
  - If we find a match, return index
  - If we don’t find a match, return -1
- if (left > right)
  - Return -1
- otherwise calculate mid-point (mid)
  - if toFind matches mid, return index
  - if toFind comes before mid, search left
  - if toFind comes after mid, search right
Recursion: Print triangle of stars
(version 1)

• if (n > 1) call printTriangle(n – 1), print n stars on a line
  *
  **
  ***

public static void printTriangle(int n) { // Assume n is >= 1 in this example
  if (n == 1) { // Base case: n == 1
    System.out.println("*");
  } else { // Recursive case: n > 1
    String s = "";
    for (int i = 0; i < n; i++) {
      s += "*";
    }
    printTriangle(n – 1);
    System.out.println(s);
  }
}
public static void printTriangle(int n) {
    // Assume n is >= 1 in this example
    if (n > 1) {
        printTriangle(n - 1);
    }
    for (int i = 0; i < n; i++) {
        System.out.print("*");
    }
    System.out.println();
}