Memory.java

1: /* Author: Richard Eisenberg
2:   File: Memory.java
3:   Demonstrates different allocation schemes in Java.
4: */
5:
6: public class Memory
7: {
8:   // raises base to non-negative power exp
9:   // arguments are allocated on the stack
10:  public static int pow(int base, int exp)
11:  {
12:    if(exp == 0)
13:    {
14:       return 1;
15:    }
16:    else
17:    {
18:       return base * pow(base, exp-1);
19:    }
20:  }
21:  }
22:
23:  // increment the first element in an array
24:  public static void increment(int[] x)
25:  {
26:    // x is allocated on the stack, but it refers
27:    // to an array allocated on the heap
28:    x[0]++;
29:  }
30:
31:  // tries to increment an argument, but doesn’t really work
32:  public static void increment(int x)
33:  {
34:    x++;
35:  }
36:  }
37:  public static void main(String[] args)
38:  {
39:    System.out.println("2^3: "+ pow(2,3));
40:    // Create a stack variable (num1) that refers to a 1-element
41:    // array allocated on the heap
42:    int[] num1 = new int[1];
43:    num1[0] = 5;
44:    System.out.println("num1 before increment: "+ num1[0]);
45:    increment(num1);
46:    System.out.println("num1 after increment: "+ num1[0]);
47:    // Create a stack variable (num2)
48:    int num2 = 5;
49:    System.out.println("num2 before increment: "+ num2);
50:    increment(num2);
51:    System.out.println("num2 after increment: "+ num2);
52:    int[] num3 = num1;
53:    System.out.println("num3 before increment: "+ num3[0]);
54:    increment(num3);
55:    System.out.println("num3 after increment: "+ num3[0]);
56:    System.out.println("num1 after incrementing num3: "+ num1[0]);
57:  }