CMPSCI 187: Programming With Data Structures

Review for First Midterm
9 October 2011

Format

- Two hours, closed-book, no calculators, computers, etc.
- Question types as on practice exam:

Java Concepts (10 * 2 = 20)
Software Engineering (2 * 5 = 10)
Tracing Code (3 * 5 = 15)
Finding Errors (3 * 5 = 15)
Timing Analysis (2 * 5 = 10)
Short Code Writing (1 * 10 = 10)
Long Code Writing (1 * 20 = 20)

Last Year’s Practice: Java Concepts

- Explain difference between the two items in each pair

int and integer
& and &&
static field and instance field
char and char[ ]
abstract data type and data structure
return value and side effect
throw and throws
Stack and StackADT
object and class
.equals (for Strings) and == (for Strings)

Last Year’s Software Engineering Questions

- Reverse display in Maze so that row 0 is at bottom rather than top
- Change path method so it only shows paths of five or fewer steps.
- Give each cell of a Maze a label, to be a `String`, in addition to its other attributes.
- In the maze-search project, allow paths to take diagonal steps as well as up, down, left, and right.

Tracing Code

- What is the output? Uses given Dog class and initial code to run establishing named Dogs:

```
pack[0] = ace;
pack[1] = cardie;
pack[2] = ace;
pack[2].setAge(5);
System.out.println(pack[0].getAge( ));

s.push (cardie);
biscuit = s.peek( );
s.push (ace);
cardie = s.pop( );
System.out.println(s.pop( ).getName( ));

s.push(cardie);
s.push(biscuit);
s.push(ace);
while (s.peek( ).getAge( ) != 3) s.pop;
System.out.println (s.pop( ).getName( ));
```

Finding Errors (Last Year’s Practice)

- May fail to compile, cause exception, or give clearly unintended output:

```
// new method in Dog class
public int dogYears( ) {
   int age;
   return age * 7;}

for (int i=0; i < 3; i++)
    pack[i] = new Dog(“dog #” + i, i);
for (int j = 0; j <= pack.length; j++)
    System.out.println (pack[i].getName( ));

pack[0] = “Ace”;
pack[1] = “Cardie”;
pack[2] = “Biscuit”;
```

Finding Errors (Last Year’s Real Exam)

- Create array but use it before populating it

```
public String toString (String name, int age) {
            String w = name + ", age: " + age;
            return w;}
```

```
Dog [ ] [ ] superpack = new Dog [5] [3];
        for (int j = 0; j < 5; j++)
           for (int i = 0; i < 3; i++)
               superpack [i] [j] = ace;
```

Timing Analysis (Practice)

- Find big-O running time of each method in terms of n, the input size:

```
public void replace (Dog [ ] kennel) {
   int n = kennel.length;
   for (int i = 0; i < n; i++) {
       kennel[i] = new Dog (“Cardie”, 3);
       for (int j = 0, (j < 7) && (j < i); j++)
           kennel[i].setAge(4);}}

// a solution to Discussion #3 in pseudocode
// move all n containers from ship to left stack
// while left stack is not empty
//      shift all containers from left to right, finding lowest
//      shift all containers from right to left, but sending out
             the first one with lowest label instead of shifting
```

Timing Analysis (Real)

- Double loop taking $O(n^2)$ even though inner loop is to i, not n:

```
 public void replace (Dog [ ] kennel) {
              int n = kennel.length;
              for (int i = 0; i < n; i++) {
                  kennel[i] = new Dog ("Cardie", 3);
                  for (int j = 0; j < i; j++)
                      kennel[j].setAge(4);}}


// while (the ship has more containers) {
//    unload the next container from the ship into temp storage
//    shift containers from left to right or right to left
//       as needed until the label of the new container is
//       between the labels of the top containers of the
//       left and right stacks
//    push the new container onto the left stack
// shift all containers to the right stack
// pop and send out each container in turn from the right stack
```

Short Coding Problem (Practice)

- Take Stack as parameter, alter it by tossing all but bottom two elements, reversing those two.
- Special cases if initially zero or one element -- empty the stack.
- I should have “Stack<T>” instead of “Stack” in the solution

Last Year's Short Coding Problems

- Write a method that takes a Stack as a parameter and alters it by throwing away all but the two bottom elements, and reversing the order of those two.
- Write a new ShowDog class extending given Dog class by adding two new attributes, the breed of a dog as a string and the entry number as an int. Give a four parameter constructor, but you need not give getters and setters

Long Coding Problem (Practice)

- JeopardyBoard, Category, Question, with given attributes.
- Method for Category to return first available Question if any.
- Method for JeopardyBoard to input Category number, return first available Question in that Category if any.

Long Coding Problem (Real)

- Write `Chessboard` as 8 by 8 array of `Square` objects. Each `Square` has a `ChessPiece` (assumed to be already defined) and an `int` called `grains`.
- Write getters and setters, but not constructors, for these classes.
- Write a method that sets the grains field of each square by setting the first one at 1 and setting each other one to twice the previous one's value.
- Trick Question: What is the value of the int in the last square? When you double any int 32 or more times, it becomes 0. Java doesn't throw an exception for integer overflow -- if you cause an int to become greater than 2^{32}, you take your chances. (This was worth two points, and no one got it fully right.)