Java Review

CS 240

Alex Vondrak

ajvondrak@csupomona.edu

Winter 2012

Programming Contests

`http://www.csupomona.edu/~carich/programming_contests/`

- Hosted by Computer Science Society (CSS) & Dr. Rich each quarter
- Monetary prizes (usually AmEx gift cards)
- Four problem statements with sample input/output
  - Write program to solve problem in general
  - Email it to Dr. Rich
  - Run against larger "acid" input, compared against "acid" output
  - If incorrect, get to resubmit... but you get penalty time
- Goal: solve the most problems in the shortest amount of time

We'll review Java by looking at the Spring 2011 problem **Dancing Squares**

Multiple Choice Question

```
public static void main(String[] arguments)
```

What does the `main` method always do?

(A) It constructs an instance of the class in which it's defined.

(B) It's automatically called when we run the program.

(C) It executes the control-flow logic of the program.

(D) It processes command-line arguments.

Multiple Choice Question

```
Scanner in = new Scanner(System.in);
```

What are `Scanners` used for?

(A) Reading input from a file or from the keyboard

(B) Writing output to a file or the screen

(C) Both input and output

(D) I've never seen `Scanner` before

Multiple Choice Question

```
int n = in.nextInt();
```

What does this line do?

(A) Asks the user to enter a number

(B) Tests whether or not the user has typed in a number

(C) Generates a default number in case the user doesn't enter anything

(D) Reads input from the user, assuming it's a number

Multiple Choice Question

```
Square first = new Square(n, in);
```

What do we call the value that's stored in `first`?

(A) A class

(B) A variable

(C) An instance

(D) A constructor

Multiple Choice Question

```
import java.util.Scanner;

class dancing_squares {

    static class Square {

      int n;
      char[][] s;
```

Why is the Square class declared **static**?

(A) It belongs to every instance of dancing_squares at once

(B) The main method is static

(C) Nested classes must be static

(D) Because **static** is similar to **private**, but not as restrictive

Multiple Choice Question

```
import java.util.Scanner;

class dancing_squares {

    static class Square {

        int n;
        char[][] s;
```

Other than "variables" or "instance variables", what do we call `n` and `s`?

(A) Fields
(B) Primitive data types
(C) Arrays
(D) Methods

Multiple Choice Question

```
Square(int n) {
    this.n = n;
    this.s = new char[n][n];
}
```

What is being defined right here?

(A) A method

(B) A Square

(C) A constructor

(D) An instance

Multiple Choice Question

```
Square(int n) {
    this.n = n;
    this.s = new char[n][n];
}
```

What does **this** refer to?

(A) The current method

(B) The current class

(C) The current instance

(D) Just the object being constructed

Multiple Choice Question

```
Square(int n) {
    this.n = n;
    this.s = new char[n][n];
}
```

The field `s` is a two-dimensional array of **`char`**s. Both of its indices (i.e., `s[index1][index2]`) have the same range. What is that range?

(A) From 0 to `n-1` (inclusive)

(B) From 1 to `n-1` (inclusive)

(C) From 0 to `n` (inclusive)

(D) From 1 to `n` (inclusive)

Multiple Choice Question

```
Square(int n, Scanner in) {
    this(n);

    for (int r=n-1; r>=0; r--) {
        String line = in.next();

        for (int c=0; c<n; c++)
            s[r][c] = line.charAt(c);
    }
}
```

Why can we have another constructor here?

(A) Its return type is **void**
(B) We call the first with **this**(n); anyway
(C) We define as many constructors as we want
(D) It takes different arguments

Multiple Choice Question

```
Square(int n, Scanner in) {
    this(n);

    for (int r=n-1; r>=0; r--) {
        String line = in.next();

        for (int c=0; c<n; c++)
            s[r][c] = line.charAt(c);
    }
}
```

What values does `r` range over in the first **`for`**-loop?

(A) `n-1` to 1 (inclusive)
(B) `n-1` to 0 (inclusive)
(C) 0 to `n-1` (inclusive)
(D) 1 to `n-1` (inclusive)

Multiple Choice Question

```
Square(int n, Scanner in) {
    this(n);

    for (int r=n-1; r>=0; r--) {
        String line = in.next();

        for (int c=0; c<n; c++)
            s[r][c] = line.charAt(c);
    }
}
```

What values does c range over in the second **for**-loop?

(A) n-1 to 1 (inclusive)
(B) n-1 to 0 (inclusive)
(C) 0 to n-1 (inclusive)
(D) 1 to n-1 (inclusive)

Multiple Choice Question

```
Square(int n, Scanner in) {
    this(n);

    for (int r=n-1; r>=0; r--) {
        String line = in.next();

        for (int c=0; c<n; c++)
            s[r][c] = line.charAt(c);
    }
}
```

How many lines are read in?

(A) 1
(B) n
(C) n-1
(D) n^2

Multiple Choice Question

```
public static void main(String[] arguments) {

  Scanner in = new Scanner(System.in);
  int n = in.nextInt();
  Square first = new Square(n, in);

  for (int count=2; in.hasNext(); count++) {
     Square next = new Square(n, in);
     System.out.printf("Square %d is %s square 1.%n", count,
        (  first.equals(next.rotate(0))
        || first.equals(next.rotate(0).hFlip())
```

What is count used for?

(A) To track the number of lines we've read

(B) To track the number of times we've rotated a Square

(C) To track the number of Squares we've read

(D) To track whether or not we've reached the end of the input

Multiple Choice Question

```
public boolean equals(Object other) {

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            if (s[r][c] != ((Square)other).s[r][c]) return false;

    return true;
}
```

What does `((Square)other)` mean?

(A) `other` must be a `Square`

(B) `other` must be a `Square`, or a subclass of it

(C) Convert `other` into a `Square`

(D) If `other` is a `Square`, perform the operation, otherwise continue silently

Multiple Choice Question

```
public boolean equals(Object other) {

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            if (s[r][c] != ((Square)other).s[r][c]) return false;

    return true;
}
```

What are expressions of the form `(type) object` called?

(A) Typecasts

(B) Autoboxing

(C) Narrowing conversions

(D) Widening conversions

Multiple Choice Question

```
public boolean equals(Object other) {

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            if (s[r][c] != ((Square)other).s[r][c]) return false;

    return true;
}
```

How many times do we compare `s[r][c]` to `other.s[r][c]` if the Squares are not equal?

(A) `n`

(B) `n + n`

(C) n^2

(D) Somewhere between 1 and n^2

Multiple Choice Question

```
public boolean equals(Object other) {

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            if (s[r][c] != ((Square)other).s[r][c]) return false;

    return true;
}
```

How many times do we compare `s[r][c]` to `other.s[r][c]` if the Squares are equal?

(A) `n`

(B) `n + n`

(C) n^2

(D) Somewhere between 1 and n^2

Multiple Choice Question

```
public boolean equals(Object other) {

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            if (s[r][c] != ((Square)other).s[r][c]) return false;

    return true;
}
```

Why couldn't we have just used `first == /* whatever */` in the `main` method?

(A) `first.equals(/* whatever */)` is easier to read

(B) == only tests if the two Squares are the same instance

(C) The `equals` method lets us compare against any type of `Object`

(D) == only works between primitive data types

Multiple Choice Question

```
Square hFlip() {

    Square result = new Square(n);

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            result.s[r][c] = s[n-1-r][c];

    return result;
}
```

The method definition could be stated more completely. Which of the following is equivalent?

(A) `public Square hFlip()`
(B) `private Square hFlip()`
(C) `protected Square hFlip()`
(D) `static Square hFlip()`

Multiple Choice Question

```
Square hFlip() {

    Square result = new Square(n);

    for (int r=0; r<n; r++)
        for (int c=0; c<n; c++)
            result.s[r][c] = s[n-1-r][c];

    return result;
}
```

Is it possible for `s[n-1-r][c]` to trigger an `ArrayIndexOutOfBoundsException`?

(A) Yes

(B) No

Multiple Choice Question

```
Square rotate(int count) {

    if (count == 0)
        return this;
    else {
        Square result = new Square(n);

        for (int r=0; r<n; r++)
            for (int c=0; c<n; c++)
                result.s[r][c] = s[n-1-c][r];

        return result.rotate(count-1);
    }
}
```

What does the `result.rotate(count-1)` accomplish?

(A) Modifies `result.s` to rotate it `count-1` more times

(B) Constructs a new `Square` that has been rotated `count-1` more times

(C) Constructs a new `Square` that has been rotated one more time

(D) None of the above

Multiple Choice Question

```
System.out.printf("Square %d is %s square 1.%n", count,
  (  first.equals(next.rotate(0))
  || first.equals(next.rotate(0).hFlip())
  || first.equals(next.rotate(1))
  || first.equals(next.rotate(1).hFlip())
  || first.equals(next.rotate(2))
  || first.equals(next.rotate(2).hFlip())
  || first.equals(next.rotate(3))
  || first.equals(next.rotate(3).hFlip())
  )  ? "identical to" : "distinct from");
```

What does a %d mean in the format string of a `printf`?

(A) Format a single digit

(B) Format any data

(C) Format a primitive data type

(D) Format a decimal number

Multiple Choice Question

```
System.out.printf("Square %d is %s square 1.%n", count,
  (  first.equals(next.rotate(0))
  || first.equals(next.rotate(0).hFlip())
  || first.equals(next.rotate(1))
  || first.equals(next.rotate(1).hFlip())
  || first.equals(next.rotate(2))
  || first.equals(next.rotate(2).hFlip())
  || first.equals(next.rotate(3))
  || first.equals(next.rotate(3).hFlip())
  )  ? "identical to" : "distinct from");
```

What does a %s mean in the format string of a printf?

(A) Format a String

(B) Format any object as a String

(C) Format "special" data

(D) Format the argument in a simplified way

Multiple Choice Question

```
System.out.printf("Square %d is %s square 1.%n", count,
  (  first.equals(next.rotate(0))
  || first.equals(next.rotate(0).hFlip())
  || first.equals(next.rotate(1))
  || first.equals(next.rotate(1).hFlip())
  || first.equals(next.rotate(2))
  || first.equals(next.rotate(2).hFlip())
  || first.equals(next.rotate(3))
  || first.equals(next.rotate(3).hFlip())
  )  ? "identical to" : "distinct from");
```

When does the expression `a || b` result in **`false`**?

(A) a is **`true`** and b is **`true`**

(B) a is **`true`** and b is **`false`**

(C) a is **`false`** and b is **`true`**

(D) a is **`false`** and b is **`false`**

Multiple Choice Question

```
System.out.printf("Square %d is %s square 1.%n", count,
  (  first.equals(next.rotate(0))
  || first.equals(next.rotate(0).hFlip())
  || first.equals(next.rotate(1))
  || first.equals(next.rotate(1).hFlip())
  || first.equals(next.rotate(2))
  || first.equals(next.rotate(2).hFlip())
  || first.equals(next.rotate(3))
  || first.equals(next.rotate(3).hFlip())
  )  ? "identical to" : "distinct from");
```

Which of the following best describes the expression "`a ? b : c`"?

(A) If a, then b, else c

(B) If a, then c, else b

(C) If b, then a, else c

(D) If c, then a, else b