CSE 142

Midterm Review Problems

Lecture outline

- expressions
- parameter mystery
- while loop simulation
- assertions
- programming problems

Expressions

- integer division and mod: quotient and remainder

```
14 / 3 is 4,      14 % 3 is 2
7 / 10 is 0,      7 % 10 is 7
```

- precedence: ( ) before * / % before + -

```
5 + 2 * 6 - (1 + 7) % 3
5 + 2 * 6 -     8   % 3
5 +  12   -     8   % 3
5 +  12   -         2
   17     -         2
          15
```

Expressions 2

- String concatenation: same precedence as integer `+ -`, evaluated left-to-right with other `+ -` operations

```
1 + 2 + "3" + 4 + 5
  3   + "3" + 4 + 5
      "33"  + 4 + 5
          "334"   + 5
              "3345"
```

- type promotion: done as needed when `int` and `double` are mixed

```
50 / 6 / 5.0
   8   / 5.0
     1.6
```

Expression questions

- Evaluate the following expressions:

```
16 / 3 + 3.2 * 2
8 / 7 + "5 / 4" + 8 / 3
88 % 10 % 3 * 16 / 10
29 / 3 / 2 / 4.0 + 3.6 * 2
1.4 + (3 + 2 * 6) / (8 - 14 / 3) * 2.2
```

Expression answers

- Correct answers:

```
16 / 3 + 3.2 * 2                                  11.4
8 / 7 + "5 / 4" + 8 / 3                           "15 / 42"
88 % 10 % 3 * 16 / 10                             3
29 / 3 / 2 / 4.0 + 3.6 * 2                        8.2
1.4 + (3 + 2 * 6) / (8 - 14 / 3) * 2.2            8.0
```

Parameter mystery question

- What is the output of the following program?

```
public class Mystery {
    public static void main(String[] args) {
        String john = "skip";
        String mary = "george";
        String george = "mary";
        String fun = "drive";
        String work = "john";

        speak(mary, john, fun);
        speak(george, work, john);
        speak(fun, "george", "work");
        speak(george, mary, john);
        speak(george, "john", "dance");
    }

    public static void speak(String mary, String john, String fun) {
        System.out.println(john + " likes to " + fun + " with " + mary);
    }
}
```

Parameter mystery tips

- Try making a table of the parameters passed to each call:

```
public class Mystery {
    public static void main(String[] args) {
        String john = "skip";
        String mary = "george";
        String george = "mary";
        String fun = "drive";
        String work = "john";

        speak(mary, john, fun);
        speak(george, work, john);
        speak(fun, "george", "work");
        speak(george, mary, john);
        speak(george, "john", "dance");
    }

    public static void speak(String mary, String john, String fun) {
        System.out.println(john + " likes to " + fun + " with " + mary);
    }
}
```

george	skip	drive
mary	john	skip
drive	george	work
mary	george	skip
mary	john	dance

Parameter mystery answer

- What is the output of the following program?

```
public class Mystery {
    public static void main(String[] args) {
        String john = "skip";
        String mary = "george";
        String george = "mary";
        String fun = "drive";
        String work = "john";

        speak(mary, john, fun);
        speak(george, work, john);
        speak(fun, "george", "work");
        speak(george, mary, john);
        speak(george, "john", "dance");
    }

    public static void speak(String mary, String john, String fun) {
        System.out.println(john + " likes to " + fun + " with " + mary);
    }
}
```

```
skip likes to drive with george
john likes to skip with mary
george likes to work with drive
george likes to skip with mary
john likes to dance with mary
```

While loop mystery question

- Given the following program,

```
public static void mystery(int y) {
    int x = 0;
    int z = 0;
    while (y > 0) {
        System.out.print(x + " " + z + " ");
        x++;
        z = z + y % 10;
        y = y / 10;
    }
    System.out.println(x + " " + z);
}
```

- What is the output of the following sequence of calls?

```
mystery(0);
mystery(8);
mystery(32);
mystery(72);
mystery(184);
mystery(8239);
```

While loop mystery tips

- Keep track of each variable's value as it changes:

```
public static void mystery(int y) {
    int x = 0;
    int z = 0;
    while (y > 0) {
        System.out.print(x + " " + z + " ");
        x++;
        z = z + y % 10;
        y = y / 10;
    }
    System.out.println(x + " " + z);
}
```

x	y	z
0	184	0
1	18	4
2	1	12
3	0	13

- What is the output of the following call?

```
mystery(184);
```

- Sometimes, these problems are performing real computations in disguise.
  - What is this problem really doing?

While loop mystery answers

- Given the following program,

```
public static void mystery(int y) {
    int x = 0;
    int z = 0;
    while (y > 0) {
        System.out.print(x + " " + z + " ");
        x++;
        z = z + y % 10;
        y = y / 10;
    }
    System.out.println(x + " " + z);
}
```

- What is the output of the following sequence of calls?

```
mystery(0);
mystery(8);
mystery(32);
mystery(72);
mystery(184);
mystery(8239);
```

```
0 0
0 0 1 8
0 0 1 2 2 5
0 0 1 2 2 9
0 0 1 4 2 12 3 13
0 0 1 9 2 12 3 14 4 22
```

Assertions question

- State whether each assertion is ALWAYS, NEVER, or SOMETIMES true at each point in the code:

```
public static void mystery(Scanner console) {
    int x = 0;
    int y = 1;
    int next = console.nextInt();

    // Point A
    while (next != 0) {
        // Point B
        y = y * next;
        if (next < 0) {
            x++;
            // Point C
        }

        next = console.nextInt();
        // Point D
    }
    // Point E
    System.out.println(y + " " + x);
}
```

	next < 0	y > 0	x > 0
Point A			
Point B			
Point C			
Point D			
Point E			

Assertions answer

- At each point, ask yourself, "How could I have arrived here?"
  - For example, Part B could be the start of the first or 100th loop pass.
  - For example, Part E might or might not have ever entered the loop.

```
public static void mystery(Scanner console) {
    int x = 0;
    int y = 1;
    int next = console.nextInt();

    // Point A
    while (next != 0) {
        // Point B
        y = y * next;
        if (next < 0) {
            x++;
            // Point C
        }

        next = console.nextInt();
        // Point D
    }
    // Point E
    System.out.println(y + " " + x);
}
```

	next < 0	y > 0	x > 0
Point A	sometimes	always	never
Point B	sometimes	sometimes	sometimes
Point C	always	sometimes	always
Point D	sometimes	sometimes	sometimes
Point E	never	sometimes	sometimes

Programming question tips

- Recognize which programming tools to use to solve each problem.
  - Repeat actions a specific number of times: `for` loop.
  - Decide between several logical choices: `if/else` statements.
  - Repeat an unknown number of times: `while` loop.
- Read the problems carefully!
  - Does it want you to *print* a result, or *return* it?
  - What values does the method use for computation? Are these values parameters, are they read from a `Scanner`, etc.?
  - What type of value (if any) does the method return?
  - Have you handled all special cases? What if the integer is 0, or negative? What if the string has no letters? What if there is only one word in the string? Many words?
- Get your thoughts onto the page.
  - A partial answer is better than none at all.
  - Writing the correct method header will earn at least 1 point.
  - If you can solve all of the problem except one part, leave that part blank or write what you wanted to do as a comment.
  - Watch the clock! 50 minutes disappear quickly.

Programming question

- Write a method named `printMultiples` that accepts as parameters an integer *n* and number of multiples *k*, and prints as output the first *k* multiples of *n* beginning with *n* itself. Assume that the value passed for *k* is greater than 0.

  For example, the following three calls,

```
printMultiples(2, 5);
printMultiples(10, 7);
printMultiples(1, 8);
```

  would produce the following output:

```
The first 5 multiples of 2 are 2, 4, 6, 8, 10
The first 7 multiples of 10 are 10, 20, 30, 40, 50, 60, 70
The first 1 multiples of 8 are 8
```

Programming answer

```
public static void printMultiples(int n, int k) {
    System.out.print("The first " + k +
            " multiples of " + n + " are " + n);
    for (int i = 2; i <= k; i++) {
        System.out.print(", " + i * n);
    }
    System.out.println();
}
```

Programming question

- Write a method named `before` that accepts as parameters four integers representing two month/day combinations and that returns whether the first date comes earlier in the year than the second date. The first integer in each pair represents the month between 1 and 12. The second integer in each pair represents the day of the month between 1 and 31. You may assume that your method is passed valid values.

  For example, these calls should return `true`:

```
before(6, 3, 9, 19)     // June 3 is before September 19
before(1, 10, 1, 11)    // January 10 is before January 11
```

  These calls should return `false`:

```
before(10, 1, 2, 25)    // October 1 not before February 25
before(8, 5, 8, 5)      // August 5 not before August 5
```

Programming answers

```
public static boolean before(int m1, int d1, int m2, int d2) {
    if (m1 < m2) {
        return true;
    } else if (m1 > m2) {
        return false;
    } else if (d1 < d2) {
        return true;
    } else {
        return false;
    }
}

public static boolean before(int m1, int d1, int m2, int d2) {
    if (m1 == m2) {
        return (d1 < d2);
    } else {
        return (m1 < m2);
    }
}

public static boolean before(int m1, int d1, int m2, int d2) {
    return (m1 < m2) || (m1 == m2 && d1 < d2);
}
```

Programming question

- Write a method named `rollDoubles` that simulates the rolling of two six-sided dice until "doubles" occurs (i.e., until the two dice values are the same), reporting each roll's result and the total number of rolls it took to reach doubles. The dice should randomly take values between 1 through 6 with equal likelihood.

  The format of your output should match the following log of execution:

```
Next roll = 2, 6
Next roll = 3, 1
Next roll = 5, 2
Next roll = 1, 2
Next roll = 2, 2
Doubles after 5 rolls
```

  Here is another example in which doubles occurs on the first roll:

```
Next roll = 4, 4
Doubles after 1 rolls
```

Programming answers

```
public static void rollDoubles() {
    Random r = new Random();
    int roll1 = r.nextInt(6) + 1;
    int roll2 = r.nextInt(6) + 1;
    System.out.println("Next roll = " + roll1 + ", " + roll2);
    int count = 1;

    while (roll1 != roll2) {
        roll1 = r.nextInt(6) + 1;
        roll2 = r.nextInt(6) + 1;
        System.out.println("Next roll = " + roll1 + ", " + roll2);
        count++;
    }
    System.out.println("Doubles after " + count + " rolls");
}

public static void rollDoubles() {
    Random r = new Random();
    int count = 0;
    int roll1, roll2;
    do {
        roll1 = r.nextInt(6) + 1;
        roll2 = r.nextInt(6) + 1;
        System.out.println("Next roll = " + roll1 + ", " + roll2);
        count++;
    } while (roll1 != roll2);
    System.out.println("Doubles after " + count + " rolls");
}
```

Programming question

- Write a method named `weave` that accepts as parameters two integers *a* and *b* and returns the result of weaving their digits together to form a single number. The last pair of digits in the result should be the last digit of *a* followed by the last digit of *b*. The second-to-the-last pair of digits in the result should be the second-to-the-last digit of *a* followed by the second-to-the-last digit of *b*. And so on. For example:

  The call `weave(128, 394)` should return `132984`
  The call `weave(394, 128)` should return `319248`

  If one of the numbers has more digits than the other, you should imagine that leading zeros are used to make the numbers of equal length. For example:

  The call `weave(2384, 12)` should return `20308142`
  The call `weave(9, 318)` should return `30198`

  You may assume that the numbers passed to weave are non-negative. You may not use strings to solve this problem; you must solve it using integer arithmetic.

Programming answer

```
public static int weave(int a, int b) {
    int answer = 0;
    int multiplier = 1;
    while (a != 0 || b != 0) {
        answer = answer + multiplier * (a % 10 * 10 + b % 10);
        multiplier *= 100;
        a /= 10;
        b /= 10;
    }
    return answer;
}
```