CS 125 - Lecture 10

Objectives: **Incrementing; Code analysis; Fun with String objects**

To-do: **MP1 re-grade? MP2 is out; readings; Piazza challenge #1**

Discuss - what is short-circuiting?

1. **Strings Activity**

- Fix the program below to determine the average word length in a text file 'speech.txt'. For simplicity, you can assume one word per line and no punctuation.

```
public class Speech {
    public static void main(String[] args)
    {

        TextIO.readFile("speech.txt");


        while( ! TextIO.eof() ) {


        }
}
```

4. **Analyze this: How many dots are printed?**

```
public static void main(String[] args) {
   int a = 0;
   int b;
   while (a < 20) {
      a += 2;
      b = 1;
      while (b < 16) {
         TextIO.put('.');
         b = b * 2;
      }
      TextIO.putln(a);
   }
}
```

2. Write the following programs (don't waste time writing the opening Class and Program statements, or writing out entire prompt text below):

```
Enter a string with exactly 5 characters. You typed:1234
Try again!
Enter a string with exactly 5 characters. You typed:12345
Yes!
```

3. Write the following programs (don't waste time writing the opening Class and Program statements, or writing out entire prompt text below):

```
Please enter a string where the first and last letters are the
same:
You typed "abbA"
You win!
```

CS 125 - Lecture 10

4. Which code snippets increment the value of count?

```
count + 1;
count = count +1;
count += 1;
count ++;
++count;
count = 1 + count;
```

5. Fix and/or simplify the following statements (don't change the context).

```
boolean output = line.indexOf("spoon") == true;

boolean output = line.indexOf("spoon") != false;

if( score > 80 == true) TextIO.putln("First");

if( score > 70 == false) TextIO.putln("Second");

if( score > 60 == false) TextIO.put("");
```

6. Pre & Post Increment Challenge (aka unreadable code)

Why does the following code print "x=2,yPost=1, yPre=6"?

```
int x = 0;

int yPost = 2 * x++ + x;

int yPre = 2 * ++x + x;

System.out.println("x="+x+",yPost="+yPost+", yPre="+yPre);
```

Objects - a sneak preview:

Strings are objects - instances of the type String (a class).

String variables (objects) are created when the program is running. Because their lengths (size) can vary, memory to house them is not allocated until the program is running - NOT at compile time (ala ints, doubles). The memory for these types of objects are stored in a special part of memory called "The Heap".

7. Fill in the missing the code and fix any errors you notice.
Update the code so it keeps asking for a password until a good password is entered.

```
____________ done = false;
________________________
TextIO. ______________________ ( "Prompt the user: New password? 10
or more characters, mixed case, no spaces" );

__________________ = TextIO. ____________

___________ short = _________________; // true if too short
___________ noUpperCase= ____________
___________ hasSpaces = ____________
___________ badPass = short || noUpperCase && hasSpaces;

if( ____________) }
    TextIO.putln("Bad password - try again.");
}
TextIO.putln("Password accepted, thanks.");
```

8. **Fix this PALINDROME CHECKER:**

```
public static void main(String[] args) {
   String original = "Bob";
   String s = original.toUpperCase();
   boolean isPalindrome = true;
   // We'll change isPalindrome to false
   // if we find a counter-example
   int lengthToCheck = s.length() / 2;
   int i = 0;
   while (i < lengthToCheck && isPalindrome) {
      if (s.charAt(i) != s.charAt(s.length() - i)) {
         isPalindrome = false;
      }
      i++;
   }
   if (isPalindrome)
      TextIO.putln(original + " is a palindrome");
}
```

CS 125 - Lecture 10

7. Analyze this: How many dots are printed?

```
public static void main(String[] args) {
   int a = 0;
   int b;
   while (a < 20) {
      a += 2;
      b = 1;
      while (b < 16) {
         TextIO.put('.');
         b = b * 2;
      }
      TextIO.putln(a);
   }
}
```

2. Analyze this: A simple math puzzle

```
public static void main(String[] args) {
    int n = 0;
    // Get a valid integer
    while (n < 1) {
       TextIO.putln("Enter a positive integer");
       n = TextIO.getlnInt();
    }

    // Will this loop forever for a particular n?
    int iterations = 0;
    while (n != 1) {
       iterations ++;
        TextIO.put(n + “,");
       if (n % 2 == 1)
           n = n * 3 + 1;
        else
           n = n / 2;
    }
    TextIO.putln("1\nFinished in "+iterations+" iterations");
}
```

9. Spot the Mastikes

Some code starts with the following :

```
String s = TextIO.getln();
boolean ok = _____
```
see erroneous expressions below

We need you to fix the following to be correct and accurate Java expressions. Note: “iff” means “if and only if”.

Evaluates to true iff *s* contains "Jim" or "Fred".
(Ignore upper/lower case e.g. “jiM” should evaluate to true)

```
s.toLowerCase.indexOf('jim') > 0 | s.toLowerCase.indexOf('Fred') == true
```

Should be true iff *s* has at least four characters and starts with "ABCD":

```
s.length = 4 & s.substring(1,4) = “ABCD”
```

Write an expression that is true iff *s* starts with "ABC" or *s* is an empty string and false otherwise:

10. Code analysis

```
// What happens if it reads "Help"?
// What happens if it reads "Think Secret!"?
  TextIO.readFile(“data.txt”);

   String word = TextIO.getln();
   int posn = word.toLowercase().indexOf("secret");
   if(posn != -1) TextIO.putln( word.substring(0,posn) );
```

11. More code analysis

```
String s1 = "where am i?";
String s2 = "now";
System.out.println("s1 is " + s1);     ________________
System.out.println("s2 is " + s2);     ________________

s2 = s1;
System.out.println("s1 is " + s1);     ________________
System.out.println("s2 is " + s2);     ________________

s1 = "hello";
System.out.println("s1 is " + s1);     ________________
System.out.println("s2 is " + s2);     ________________
```

9. How to think about variables...

Fixed value: The role of a variable or an attribute is a fixed value, if its value is not changed after initialization.

Stepper: Stepper goes through a succession of values in some systematic way, predictable succession of values.

Most-recent holder: The value of a most recent holder is the latest gone through value of a certain group or simply the latest input value.

Most-wanted holder: The value of a most-wanted holder is the "best" or otherwise the most-wanted value out of the values gone through so far. THE most-wanted can mean, for example, the smallest or the biggest number or a number closest to a certain value.

Gatherer: The value of a gatherer accumulates all the values gone through so far.

Follower: A follower always gets the old value of another known variable or attribute as its new value.

One-way flag: A one-way flag has two possible values but cannot get its original value anymore after it has been once changed.

Temporary: The value of a temporary is always needed only for a very short period.

10. Truth Tables

Write out a Truth Table for the following expression and then simplify the expression. Variables a and *b* are boolean.

```
boolean c = (a || !b) != (a && b)
```

a	b	a || !b	a && b	c

simplify to: ______________________

8. Short circuiting - a cool trick (and it might be on the exam...)

```
double distToObstacle = …, speed = …;
boolean canAccelerate = distToObstacle / speed > 3.5;
```

Me: Avoid division-by-zero using short circuiting:

You: Modify the expression so that canAccelerate is also true if *speed* is exactly zero. Assume distance and speed are double types.

Under what conditions will the last term of each of these expressions be evaluated?

```
boolean openVault = businessHours && key1 && key2;
```