More Loops

In this lecture we will repeat ourselves.

In this lecture we will repeat ourselves and cover:

- Incrementor / decrementor review
- More loops
- Break/Continue
- Do-While loops

Your future in CS

I used to include this on my slides, but since these slides have changed - going to just leave it up here for every notebook. I get a lot of questions about more programming courses, the concentrations, and minors in computer science. Here is a brief reminder.

CS 165 – Next Course In Sequence, also consider CS 220 (math and stats especially)

- CO Jobs Report 2021 – 77% of *all* new jobs in Colorado require programming
- 60% of all STEM jobs requires *advanced* (200-300 level)
- 31% of all Bachelor of Arts degree titled jobs also required coding skills
- 2016 Report found on average jobs that require coding skills paid $22,000 more

- Concentrations in CS:
    - Computer science has a number of concentrations.
        - General concentration is the most flexible, and even allows students to double major or minor pretty easily.
        - Software Engineering
        - Computing Systems
        - Human Centered Computing
        - Networks and Security
        - Artificial Intelligence
        - Computer Science Education.
    - Minors:
        - Minor in Computer Science - choose your own adventure minor
        - Minor in Machine Learning - popular with stats/math, and engineering
        - Minor in Bioinformatics - Biology + Computer Science

Reading Review

-- There will be no iclicker required, just discussion due to guest instructor.

Answer the questions based on the following code:

Question 1:

What is printed at the end?

Question 2:

What is the value of counter?

In [4]:
```
int counter = 10;
String tracker = "Track:";

do {
    tracker += " " + counter;

}while(++counter < 11);

System.out.println(counter);
System.out.println(tracker);
```

```
11
Track: 10
```

And the final discussion item, what happens if we change

`++counter` to `counter++`

In [5]:
```
int counter = 10;
String tracker = "Track:";

do {
    tracker += " " + counter;

}while(counter++ < 11);

System.out.println(counter);
System.out.println(tracker);
```

```
12
Track: 10 11
```

Incrementor and Decrementors

While not "required" for coding, there are uses for incrementor and decrementor operators.

- ++ or -- BEFORE the variable means
  - add or subtract by 1, and then use the modified variable
- ++ or -- AFTER the variable means
  - use the value in the variable, and then add or subtract by one modifying the variable

Admittedly, good coding can avoid using it, but there are cases where it just makes life easier.

In [7]:
```
public class TrackerExample {
    ArrayList<String> values;
    int current = 0;

    public TrackerExample(ArrayList<String> toTrack) { // just adding values here to m
        values = toTrack;
    }

    public String getCurrent() {
        return values.get(current++);
    }

}

ArrayList<String> students = new ArrayList<>();

students.add("Scoobert \"Scooby\" Doo");
students.add("Norville \"Shaggy\" Rogers");
students.add("Daphne Blake");
students.add("Velma Dinkley");
students.add("Fred Jones");
TrackerExample tracked = new TrackerExample(students);


System.out.println(tracked.getCurrent());

System.out.println(tracked.getCurrent());
```

```
Scoobert "Scooby" Doo
Norville "Shaggy" Rogers
```

Needed? - Your call.

```
public String getCurrent() {
    return values.get(current++);
}
```

could have been written as:

```
public String getCurrent() {
    String tmp = values.get(current);
    current += 1;
    return tmp;
}
```

The style / choice is yours.

Do-While

When it comes to looping we have covered:

- while loop
- for loop
- for:each loop

There is a 4th type in java (and most languages)

The **do-while** loop.

What makes it different?

The do-while loop is the loop in which 1 run is guaranteed!

Format:

```
do {
    // code to repeat
}while(/*condition*/); // notice semi at the end.
```

In [10]:
```
Scanner in = new Scanner(System.in);

String answer = "y";
while(answer.startsWith("y")) {
    System.out.println("Continue (yes or no)? ");
    answer = in.nextLine().trim();
}

System.out.println("Answer is currently: " + answer);
do {
    System.out.println("Continue (yes or no)? ");
    answer = in.nextLine().trim();
}while(answer.startsWith("y"));
```

```
Continue (yes or no)?
Answer is currently: no
Continue (yes or no)?
Continue (yes or no)?
Continue (yes or no)?
```

For the most part, you actually don't use do-while, but for the cases it is useful, it is extremely useful!

In class activity

Write a program that prompts the user to enter a number between 1-12. Keep looping until they guess the correct number.

See Inclass.java for a skeleton setup.

In [15]:
```
class GuessingGame {
    Random rnd = new Random();
    Scanner in = new Scanner(System.in);

    public void runGame() {
        int correct = rnd.nextInt(12)+1; // nextInt returns 0-11, +1 makes it 1-12
        int guess = 0;
        System.out.println("TESTING/SPOILER - the number is " + correct);
        do {
            System.out.print("Guess a number between 1 and 12: ");
            if(in.hasNextInt()) {
                guess = in.nextInt();
            }else {
                System.out.println("Not a number! Try again");
```

```
                in.next();
            }
            System.out.println();
        }while(guess != correct);
        System.out.println("Congrats you escaped!");
    }
}

GuessingGame game = new GuessingGame();
game.runGame();
```

```
TESTING/SPOILER - the number is 12
Guess a number between 1 and 12:
Congrats you escaped!
```

Break and Continue

- break
  - Sometimes you want to exit a loop early
- continue
  - Sometimes you want to restart at the top of the loop / skip the rest of the code below it.
  - Once we introduce continue, for and while loops start to become different!

Useful commands to help modify how loops work!

Let's look at code.

In [16]:
```
for(int i = 1; i < 10; i++) {
    if (i % 2 == 0) break;
    System.out.println("Odd? " + i);
}
```

```
Odd? 1
```

Notice, the loop stopped the moment break was executed!

In [17]:
```
for(int i = 0; i < 10; i++) {
    if (i%2 == 0) continue;
    System.out.println("Odd? " + i); // this line will never be executed when i is eve
}
```

```
Odd? 1
Odd? 3
Odd? 5
Odd? 7
Odd? 9
```

More realistic example?

What if we wanted to skip empty lines when reading a file?

- Option A: We nest everything in a giant if/else statement
- Option B: We have small if with a continue that skips the rest of the lines

Both are valid, and the preference is yours.

Nesting Loops

Loops can contain any block of code, which means it can contain more loops!

Let's look at the following code, and try to figure out what is printed!

In [22]:
```
for (char c = 'A'; c < 'D'; c++) {
    System.out.print("Seat:");
    for (int i = 0; i < 2; i++) {
        System.out.print(" " + c + i);
    }
    System.out.println();
}
```

```
Seat: A0 A1
Seat: B0 B1
Seat: C0 C1
```

Question 1:

What code would I want to change if I wanted to make 3 seats per row?

Question 2:

What code would I change if I wanted to add another row?

Question 3:

What are other examples you can think of with row by column (you are looking at one...)

In Class Activity

Your task is to do the following:

- In the 'countCharacters' method, it will count
  - count the number of times a character shows up
  - print the character:count
    - after you are done counting, but before the next letter
- You *do not* have to worry about removing duplicates.
  - Though if you have time, that would be good challenge using an ArrayList to track the characters you have seen.

Implementation tips:

1. discuss as a group how would you do this
   - Can you talk about it in english and then work towards pseudocode.
2. First code what you know.

- Write a loop that just prints the characters followed by a colon.
- Add an inner loop that also loops through the string. Print out what is going on!

Example `countCharacters("blueberry")` will print:

```
b:2
l:1
u:1
e:2
b:2
e:2
r:2
r:2
y:1
```

In [26]:
```java
public static void countCharacters(String line) {
    for(int i = 0; i < line.length(); i++) {
        char current = line.charAt(i);
        int count = 0;
        for(int j = 0; j< line.length(); j++) {
            if(current == line.charAt(j)) {
                count += 1;
            }
        }
        System.out.println(current + ":" + count);
    }
}


countCharacters("blueberry");
System.out.println();
countCharacters("kinnikinnick");
```

```
b:2
l:1
u:1
e:2
b:2
e:2
r:2
r:2
y:1

k:3
i:4
n:4
n:4
i:4
k:3
i:4
n:4
n:4
i:4
c:1
k:3
```

Overall

There is a lot of power with loops BUT humans find them hard. We aren't used to thinking in this manner.

- Keep working and practicing.
- Develop in small steps.
- Constantly print or use a debugger to see/visualize what is going on!
- Never write all your code without compiling / testing it, always do it in steps.