More Branching

- In this lecture, we will cover:
  - conditional expression / ternary statement
  - switch statement
  - enum

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

If/Else Statement a Review

```
if(/*condition is true*/) {
    // do something
}else {
    // do something if condition is false
}
```

A common case is to set a variable or value based on a condition:

In [1]:
```
boolean isOn = true;
int lightStrength  = -1;
if(isOn) {lightStrength = 100;}
else {lightStrength = 0;}

System.out.println(lightStrength);
```

```
100
```

Student Practice

- Create a new Java file
- Instead the main or additional static methods
  - Create if/else statements that set a value to a variable
  - It can be whatever you want (see above)

Conditional Expressions / Ternary Statements

- terms used inter-changeably
- A way to write a *simple* if/else on one line.

In [3]:
```
String time = 10 > 5 ? "hello" : "goodbye";

System.out.println(time);// what is printed?
```

```
hello
```

Conditional Expression format

- condition ? value if true : value if false

This is most commonly used with variable assignment, but doesn't have to be.

In [5]:
```
System.out.println(-10 > 5 ? "Greater!" : "Smaller!");
```

```
Smaller!
```

Student Practice

Rewrite the statements you wrote earlier as Ternary statements! Discuss them at the table.

Overall

Do you have to use conditional statements? nope, not at all. Can just make things easier.

Switch Statements

First, let's practice if/else if/else statements.

- Write an if/else statement that
- Checks to see if a student_class (String variable) is:
  - "Fencing"
    - Then set meeting_info (String variable) to "Wednesday, 4:30PM"
  - "Boxing"
    - Then set meeting_info (String variable) to "Thursday, "5:00 PM"
  - "Aikido"
    - Then set meeting_info (String variable) to "Monday, 6:00 AM"
  - "Nothing" or null
    - Then set meeting_info to the empty String ("")

In [6]:
```
public String getClassTime(String class_taught) {
    String meeting_time = "";
    if(class_taught.equalsIgnoreCase("fencing")) {
        meeting_time = "Wednesday, 4:30 PM";
    }else if(class_taught.equalsIgnoreCase("boxing")) {
        meeting_time = "Thursday, 5:00 PM";
    }else if(class_taught.equalsIgnoreCase("aikido")) {
        meeting_time = "Monday, 6:00 AM";
    }else {
        meeting_time = "";
    }
    return meeting_time;
}

System.out.println(getClassTime("Fencing"));
```

```
Wednesday, 4:30 PM
```

Switch Statements

- A statement that is a bunch of equal equivalencies is common
- switch statements were developed for that reason

- Format
  - Note: this format is for Java < 1.15 (and the more common way to do it)
    - java 1.15 they updated the format, but it is backwards compatible to the old format
    - Moral of the story, don't like IntellJ write them for you!

In [8]:
```
public static String switchTest(String name){
    String faeType;
    switch(name) {  // inside the switch is the 'variable to check'
        case "dyson":  // checks for 'equivalency' using == if primitive, or .equals i
            faeType = "Werewolf";
            break;   // says stop processing the code / leave the switch
        case "trick":
            faeType = "Sage";
            break;
        case "bo" :  // if break does not exist, continues processing code!
        case "aife":
            faeType = "Succubus";
```

```
            break;
            case "vex":
            faeType = "Mesmer";
            break;
        default:
            faeType = "human";
    }
    return faeType;
}
```

In [9]:
```
System.out.println(switchTest("trick"));
System.out.println(switchTest("vex"));
System.out.println(switchTest("kenzie"));
```

```
Sage
Mesmer
human
```

In [10]:
```
System.out.println(switchTest("Vex!")); // given the code above, what is printed?
```

```
human
```

Student Practice

Write the if/else if/else you wrote above as a switch statement!

In [23]:
```
public String getClassTime2(String class_taught) {
    String meeting_time = "";
    switch(class_taught.toLowerCase()) {
        case "fencing":
            meeting_time = "Wednesday, 4:30 PM";
            break;
        case "boxing":
            meeting_time = "Thursday, 5:00 PM";
            break;
        case "aikido":
            meeting_time = "Monday, 6:00 AM";
            break;
        default:
            meeting_time = "";
    }
    return meeting_time;
}

System.out.println(getClassTime2("Fencing"));
```

```
Wednesday, 4:30 PM
```

Enum

- Strings have a lot of variability
- Wouldn't it be great if we could define a small 'subset' of items
- Introducing Enum

In [16]:
```
public enum Names {
   DYSON,
```

```
    TRICK,
    BO,
    AIFE,
    VEX,
    KENZIE
}
```

- The enum you create is type (like objects are types)
- Actually a special object
  - Don't let the javadoc confuse you!
- Writing them is meant to be simple, the above would be in a file called Names.java

In [19]:
```
public static String switchTest2(Names name){ // notice the parameter
    String faeType;
    switch(name) {
        case DYSON:
            faeType = "Werewolf";
            break;
        case TRICK:
            faeType = "Sage";
            break;
        case BO:
        case AIFE:
            faeType = "Succubus";
            break;
                case VEX:
            faeType = "Mesmer";
            break;
        default:
            faeType = "human";
    }
    return faeType;
}

System.out.println(switchTest2(Names.BO));
System.out.println(switchTest2(Names.VEX));
System.out.println(switchTest2(Names.KENZIE));
```

```
Succubus
Mesmer
human
```

In [20]:
```
System.out.println(switchTest2(Names.ALICE)); // won't even compile
```

```
|   System.out.println(switchTest2(Names.ALICE)); // won't even compile
cannot find symbol
  symbol:   variable ALICE
```

Student Practice

- Create an Enum called MAClasses
  - Populate it with FENCING, BOXING, AIKIDO
- Add another switch statement that uses the enum instead of strings!

In [21]:
```
public enum MAClasses {
```

```
    FENCING, BOXING, AIKIDO
}
```

In [24]:
```
public String getClassTime3(MAClasses class_taught) {
    String meeting_time = "";
    switch(class_taught) {
        case FENCING:
            meeting_time = "Wednesday, 4:30 PM";
            break;
        case BOXING:
            meeting_time = "Thursday, 5:00 PM";
            break;
        case AIKIDO:
            meeting_time = "Monday, 6:00 AM";
            break;
        default:
            meeting_time = "";
    }
    return meeting_time;
}

System.out.println(getClassTime3(MAClasses.FENCING));
```

```
Wednesday, 4:30 PM
```

In [26]:
```
// Also you can convert between Strings and Enums (helpful for the final project)

System.out.println(MAClasses.BOXING.toString());

MAClasses classType = MAClasses.valueOf("FENCING");

System.out.println(getClassTime3(classType));
```

```
BOXING
Wednesday, 4:30 PM
```

Overall

Both ternary and switch statements are optional, but extremely useful!

Enums are used commonly to help create safer code