Conditionals
Control Flow Statements

Thus far, all our programs execute in a linear fashion. 

*Control flow statements* enable our programs to have various types of branching.

We’ll see three types of control flow statements:
- conditionals
- loops
- methods
Conditional Statements

Basic premise: *if* some expression is true, then execute this code

“If the user’s purchase is more than $50, do not charge shipping”

More complex conditional statements enable different logic

“If there are 8 or more people eating, then charge 20% gratuity. Otherwise, charge 15% gratuity.”

“If income is less than $25,000, tax at a rate of 15%. If income is $25,000-$50,000, tax at a rate of 20%. If income is greater than $50,000, tax at a rate of 25.”
Types of Conditional Statements

- if statements
- if-else statements
- if-else if statements
- if-else if-else statements
Types of Conditional Statements

- if statements
- if-else statements
- if-else if statements
- if-else if-else statements
if Statements

Allows us to make a decision on whether or not to execute some code

*if* the boolean expression is true, execute the code in the brackets
otherwise (i.e., if the boolean expression is false), skip the code in the brackets

```cpp
if (<boolean expression>) {
    // code to execute if boolean expression is true
}
// code to execute after if statement
```
if Statements

Allows us to make a decision on whether or not to execute some code

if the boolean expression is true, execute the code in the brackets
otherwise (i.e., if the boolean expression is false), skip the code in the brackets

Code example:

```java
if (<boolean expression>) {
    // code to execute if boolean expression is true
}
// code to execute after if statement
```
if Statements

Allows us to make a decision on whether or not to execute some code

if the boolean expression is true, execute the code in the brackets
otherwise (i.e., if the boolean expression is false), skip the code in the brackets

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
}
//code to execute after if statement
```
if Statements

Allows us to make a decision on whether or not to execute some code

- If the boolean expression is true, execute the code in the brackets.
- Otherwise (i.e., if the boolean expression is false), skip the code in the brackets.

```
if (<boolean expression>) {
    // code to execute if boolean expression is true
}

// code to execute after if statement
```
if Statements

Allows us to make a decision on whether or not to execute some code

if the boolean expression is true, execute the code in the brackets
otherwise (i.e., if the boolean expression is false), skip the code in the brackets

```c
if (<boolean expression>) {
    // code to execute if boolean expression is true
}

// code to execute after if statement
```
if Statements

Allows us to make a decision on whether or not to execute some code

*if* the boolean expression is true, execute the code in the brackets

otherwise (i.e., if the boolean expression is false), skip the code in the brackets

```
if (<boolean expression>) {
    //code to execute if boolean expression is true
}

> //code to execute after if statement
```
Types of Conditional Statements

- if statements
- if-else statements
- if-else if statements
- if-else if-else statements
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}
//code to execute after if-else statement
```

N.B.: notice the lack of boolean expression with the else block!
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```c
> if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}

//code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

if the boolean expression is true, execute the code in the block underneath the if
else (i.e., if the boolean expr. is false), execute the code in the block underneath the else

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}

//code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```java
if (<boolean expression>) {
    // code to execute if boolean expression is true
} else {
    // code to execute if boolean expression is false
}

> // code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}
//code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```plaintext
if (<boolean expression>) {
    // code to execute if boolean expression is true
} else {
    // code to execute if boolean expression is false
}
// code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}

//code to execute after if-else statement
```
if-else Statements

Makes a decision to execute one block of code or another block of code

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else* (i.e., if the boolean expr. is false), execute the code in the block underneath the *else*

```java
if (<boolean expression>) {
    //code to execute if boolean expression is true
} else {
    //code to execute if boolean expression is false
}
//code to execute after if-else statement
```

N.B.: exactly one of these two blocks will execute!
Types of Conditional Statements

if statements

if-else statements

if-else if statements

if-else if-else statements
if-else if Statements

Makes at most one decision amongst many boolean expressions

* if the boolean expression is true, execute the code in the block underneath the *if

* else if* the previous boolean expr. is false and the boolean expr. in the *else if* is true,
execute the code in the block underneath the *else if*

otherwise (i.e., if the previous boolean exprs. are false), skip all the code in the brackets

Can have one or more *else if* statements
if-else if Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement

N.B.: we could have greater or fewer else if blocks; this example happens to have two
if-else if Statements

```c
> if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement
```
if-else if Statements

```java
if (<boolean expression 1>) {
   // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
   // code to execute if boolean expression 2 is true
   // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
   // code to execute if boolean expression 3 is true
   // and the previous boolean expressions are false
}
// code to execute after if-else if statement
```
if-else if Statements

```plaintext
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true and the previous boolean expressions are false
}
> // code to execute after if-else if statement
```
if-else if Statements

if (<boolean expression 1>) {
    //code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    //code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    //code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

//code to execute after if-else if statement
if-else if Statements

```
if (<boolean expression 1>) {
    //code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    //code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    //code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}
//code to execute after if-else if statement
```
if-else if Statements

```java
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}
// code to execute after if-else if statement
```
if-else if Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement
### if-else if Statements

```c
>if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}
// code to execute after if-else if statement
```
if-else if Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true and the previous boolean expressions are false
}

// code to execute after if-else if statement
if-else if Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement
if-else if Statements

if (<boolean expression 1>) {
   // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
   // code to execute if boolean expression 2 is true
   // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
   // code to execute if boolean expression 3 is true
   // and the previous boolean expressions are false
}

// code to execute after if-else if statement
if-else if Statements

N.B.: the order of the if and else if statement(s) matters!

```c
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

> // code to execute after if-else if statement
```
if-else if Statements

> if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement
if-else if Statements

```c
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}

// code to execute after if-else if statement
```
if (boolean expression 1) {
    // code to execute if boolean expression 1 is true
} else if (boolean expression 2) {
    // code to execute if boolean expression 2 is true and the previous boolean expressions are false
} else if (boolean expression 3) {
    // code to execute if boolean expression 3 is true and the previous boolean expressions are false
}

// code to execute after if-else if statement
if-else if Statements

N.B.: in this case, we execute none of the if or else if statements
Types of Conditional Statements

if statements

if-else statements

if-else if statements

if-else if-else statements
if-else if-else Statements

Makes exactly one decision amongst many boolean expressions

*if* the boolean expression is true, execute the code in the block underneath the *if*

*else if* the previous boolean expr. is false and the boolean expr. in the *else if* is true, execute the code in the block underneath the *else if*

*else* (i.e., if all previous boolean exprs. are false), execute the code in the block underneath the *else*

Can have one or more *else if* statements

Must have exactly one *else* statement
if-else if-else Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean expressions are false
}

// code to execute after if-else if-else statement

N.B.: we could have greater or fewer else if blocks; this example happens to have two
if-else if-else Statements

```c
> if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean
    // expressions are false
}
```

// code to execute after if-else if-else statement
if-else if-else Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean expressions are false
}

// code to execute after if-else if-else statement
if-else if-else Statements

```java
if (<boolean expression 1>) {
    //code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    //code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    //code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    //code to execute if the previous boolean
    // expressions are false
}

//code to execute after if-else if-else statement
```
if-else if-else Statements

```plaintext
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean expressions are false
}
// code to execute after if-else if-else statement
```
if-else if-else Statements

```c
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean
    // expressions are false
}
> // code to execute after if-else if-else statement
```
if-else if-else Statements

if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean
    // expressions are false
}

// code to execute after if-else if-else statement
if-else if-else Statements

```java
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean
    // expressions are false
}

// code to execute after if-else if-else statement
```
if-else if-else Statements

```
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean
    // expressions are false
}

// code to execute after if-else if-else statement
```
if-else if-else Statements

if (<boolean expression 1>) {
    //code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    //code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    //code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
}> else {
>    //code to execute if the previous boolean
>    //expressions are false
}

//code to execute after if-else if-else statement
if-else if-else Statements

N.B.: we will always execute exactly one of these blocks

```c
if (<boolean expression 1>) {
    // code to execute if boolean expression 1 is true
} else if (<boolean expression 2>) {
    // code to execute if boolean expression 2 is true
    // and the previous boolean expressions are false
} else if (<boolean expression 3>) {
    // code to execute if boolean expression 3 is true
    // and the previous boolean expressions are false
} else {
    // code to execute if the previous boolean expressions are false
}
```

//code to execute after if-else if-else statement
In A Nutshell

We have three types of conditional blocks: if, else if, and else.

Every conditional statement...

  - must start with exactly one if block
    - if blocks are always the first statement
  - can be followed by zero or more else if blocks
  - can have exactly zero or one else blocks
    - else blocks are always the last statement (if included)

If there is an else block, exactly 1 block will execute.

If there is no else block (i.e., just if, or if + else if), then either 0 or 1 blocks will execute.
A Note on Naming Variables...

Sensibly naming variables becomes more important with control flow statements.

What is the following code used for?

```java
if (x == 'A' && y == 'S') {
    System.out.println("Match found");
}
```
A Note on Naming Variables...

Sensibly naming variables becomes more important with control flow statements

What is the following code used for?

```java
if (firstInitial == 'A' && lastInitial == 'S') {
    System.out.println("Match found");
}
```
Activity Diagrams: Symbols

- Activity
- Control flow
- Merge/split
- Start
- End
Activity Diagrams: Symbols

- **Activity**: Represents a single activity, a set of instructions.
- **Control flow**: In software... one or more statements
- **Merge/split**: “read in user input”
- **Start**
- **End**
Activity Diagrams: Symbols

- **Activity**: control flow: the order of execution
- **Control flow**: Communicates the order of activities
  e.g., you print a prompt before reading input
- **Merge/split**
- **Start**
- **End**

Diagram:
- Print prompt
  - Read input
Activity Diagrams: Symbols

- **Activity**: Used to merge or split control flow. Handle different sequences of activities. E.g., asking user for age.

- **Control flow**

- **Merge/split**

- **Start**

- **End**

The diagram shows a process for handling user input based on age:
- Scan user input.
- If age <= 12, print "child discount"; otherwise, charge ticket price.
Activity Diagrams: Symbols

- **Activity**: Denotes where the algorithm begins and ends
- **Control flow**
- **Merge/split**
- **Start**
- **End**

Diagram example:
- **Start** connected to **print prompt**
- **print prompt** connected to **print output**
- **print output** connected back to **Start**
Combining Conditional Statements

Power of programming comes from combining control flow statements
placing one conditional statement after another
nesting one conditional statement inside another

Can do this in as many possible combinations as you can imagine!
but, you typically don’t need to get too complex
greater complexity often indicates a lack of clear problem solving strategy
To Consider

You know you need to use a conditional statement, but how to pick?

Ask yourself…

if none of my conditions work out, do I have a default outcome? **have an else**

are my outcomes mutually exclusive of one another? **have an if + else if(s)**

do I want to be able to execute two or more of my outcomes? **use a sequence of conditionals**