CS180 Lab 04
Selection

Week 4
Jan. 28 - Feb. 3

Selection Statements

- **Used to create control within a program**
  - Allows the program to make decisions based on user input, variables, or anything really

- **Three selection statements in Java (and most languages)**
  - `if / else if`
  - `else`
  - `switch`

`if` statement

```
...
if (condition == true) {
    statement(s) to execute
    ...
}
...
```

← Your condition goes here. The block below will not run unless this condition asserts (returns) true.

`if else` statement

```
...
if (condition == true) {
    statement(s) to execute
    ...
}
else if (condition == true) {
    other statement(s)
    ...
}
...
```

`if` statements can be joined together by using `else if`. Only one statement in this chain will run, so try to keep the conditions mutually exclusive (no overlap).

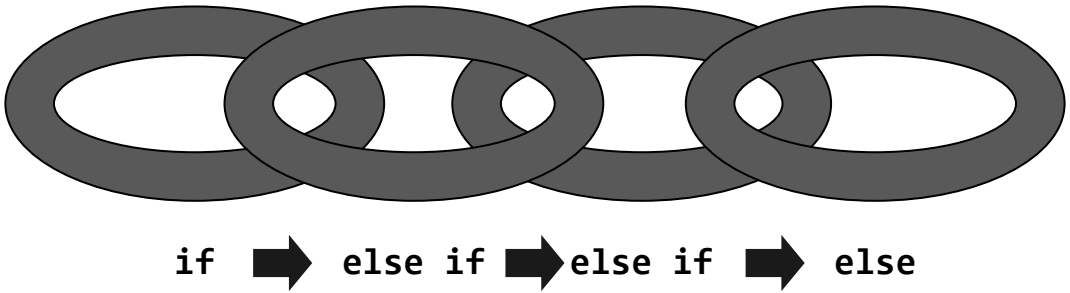
← Another condition goes here. If this block runs, any other blocks below will not run.

`else` statement

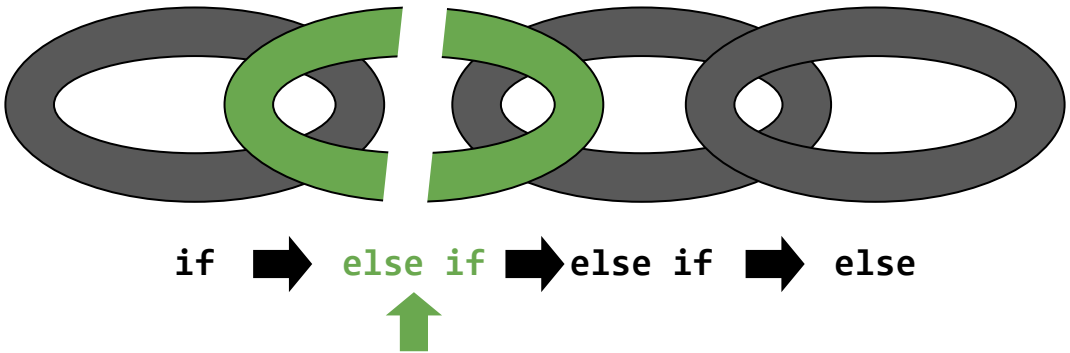
```
...
if (condition == true) {
    statement(s) to execute
    ...
}
...
else {
    Other statement(s)
    ...
}
...
```

← This acts as a catch-all. If none of the blocks chained above run, this will run instead. Oftentimes, this is not needed, but can make your code more readable.

Think of `if`, `else if`, and `else` statements as a chain

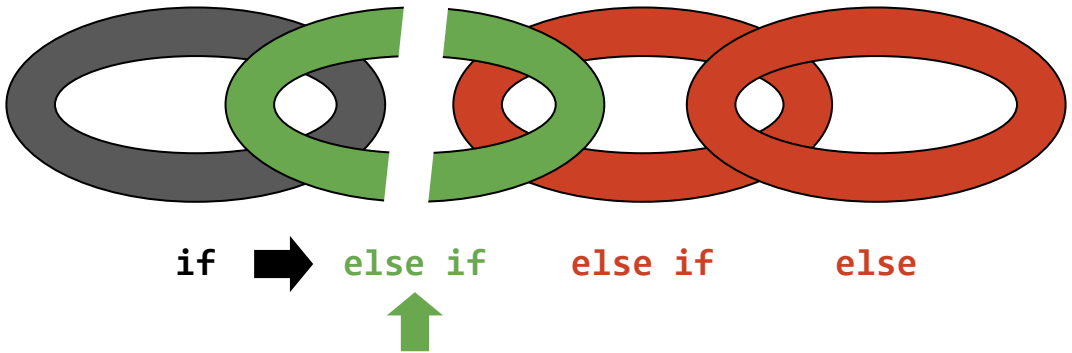


Think of `if`, `else if`, and `else` statements as a chain



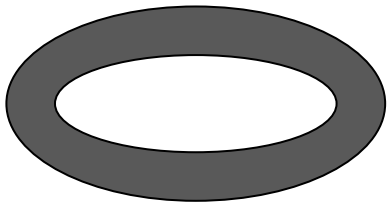
If one of the statements is ran, the link breaks.

Think of `if`, `else if`, and `else` statements as a chain

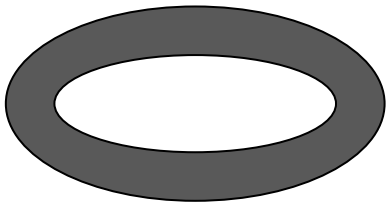


This means the links after cannot be ran.

Of course we can have separate `if` statements which don't affect each other

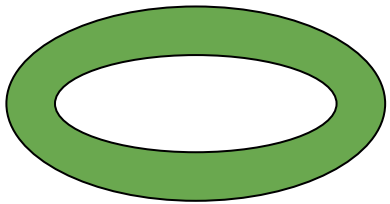


`if`

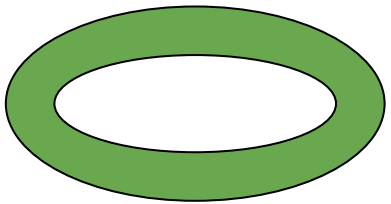


`if`

Of course we can have separate `if` statements which don't affect each other



`if`



`if`

Both statements can be true and both can be executed.

`switch` statement

```
...
switch (variable x) {
    case a:
        statement(s) to execute
        ...

    case b:
        other statement(s)
        ...
}
...
```

← x is what you are testing on. This will be the basis for selection.

`switch` statement

```
...
switch (variable x) {
    case a:
        statement(s) to execute
        ...

    case b:
        other statement(s)
        ...
}
...
```

← If x == a, then this block of statements will execute.

`switch` statement

```
...
switch (variable x) {
    case a:
        statement(s) to execute
        ...

    case b:
        other statement(s)
        ...
}
...
```

← If x == b, then this block of statements will execute.

`switch` statement

```
...
switch (variable x) {
    case a:
        ...

    default:
        other statement(s)
        ...
}
...
```

← You can also have a catch-all that will run regardless of the other blocks running.

`switch` statement

- **Cases in switch statements do not break the execution of other cases after them**
  - In fact, all cases after the first case to match the condition **will** run

- **Oftentimes, this is not ideal for our purposes**
  - To fix this, we add **`break`** statements in our cases

`switch` statement

```
...
switch (variable x) {
    case a:
        statement(s) to execute
        ...
        break;

   ...
}
...
```

← This will stop all other cases from running. Do this for any cases where you do not want execution in the `switch` to continue.