Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email – Please see syllabus

NEW! Supplemental Instruction: Sign up at my.newpaltz.edu

CONDITIONAL STATEMENTS

COMPUTER SCIENCE I

OBJECTIVES

- Review
- More on Variables
- Casting
- if Statements
- if/else Statements
- if/else if/else Statements
- switch Statements
- More Conditionals

TRY THE FOLLOWING

- What is **21 / 2** in Java?
- What is **21 % 3** in Java?
- What is **21 % 10** in Java?
- What is **21.0 / 2** in Java?
- Declare and initialize a variable to the value **true**.

TRY THE FOLLOWING

- What is **21 / 2** in Java? **10**
- What is **21 % 3** in Java? **0**
- What is **21 % 10** in Java? **1**
- What is **21.0 / 2** in Java? **10.5**
- Declare and initialize a variable to the value **true**

  boolean b = true;

SOME MORE SHORTCUTS

```
//declaring more than one variable on the same line
int x, y, z;

//same as declaring on separate lines
int x;
int y;
int z;
```

```
//declaring and initialing more than one variable on the same line
int x = 1, y = 2, z = 3;

//same as using separate lines
int x = 1;
int y = 2;
int z = 3;
```

TYPE CASTING

- **Casting** is an operation that converts a value of one data type into a value of another data type.
- Casting a type with a small range to a type with a larger range is known as **widening** a type. Casting a type with a large range to a type with a smaller range is known as **narrowing** a type.
- Java will automatically widen a type, but you must narrow a type explicitly.
  - **Widening a type example:**
    - int x = 5;
    - double y = x;
  - **Narrowing a type example:**
    - double x = 3.45;
    - int y = (int) x; //notice the (int), this is how we explicitly narrow the type

TYPE CASTING

- **Narrowing a type** can cause us to lose information, so be mindful of this when you use it.

A cast disguises the value to be a different type.

x = (int) 7.2;

*by putting the cast (int) it chops the double at the decimal and takes the integer portion only

*java doesn't know rounding so even if it were 7.9, x would still get 7 after casting.

Besides the differing sizes is there any other reason java would not do auto casting for us here?

TYPE CASTING EXAMPLES

```
//Java doesn't round!
int x = (int)25.76;
int y = (int)127.92;

double z = 7.89;
int i = (int)z;

System.out.println(x);
System.out.println(y);
System.out.println(z); // z is still a double, i is an integer
System.out.println(i);
```

Output

```
25
127
7.89
7
```

AN EXAMPLE…

- Let's look at how to determine **BMI**. How could we create a program that lets us output the health state of a person's BMI?

A person's health state with respect to weight is roughly rated as follows:

BMI < 18.5: **Underweight**
18.5 ≤ BMI ≤ 25: **Normal**
25 < BMI ≤ 30: **Overweight**
BMI > 30: **Obese**

AN EXAMPLE…

- We can use **conditional statements!** (A.K.A. selection statement). We'll first look at the **if statement**.
- This is where our knowledge of boolean data types and logical operators come into play (&&, ||). Let's first do a recap with boolean symbols…

BOOLEAN RECAP

- A boolean is a binary type that can be either **true** or **false** in Java.
- These symbols are important for conditional statements

Symbol	Meaning
==	Equals
!	Not
!=	Not equals
<, <=	Less than, less than or equal
>, >=	Greater than, greater than or equal
&&	And
||	Or

IF STATEMENT FLOW

- Every line of code will run until the conditional is hit. If the condition is met, the code inside will run. Else, Java will skip it.

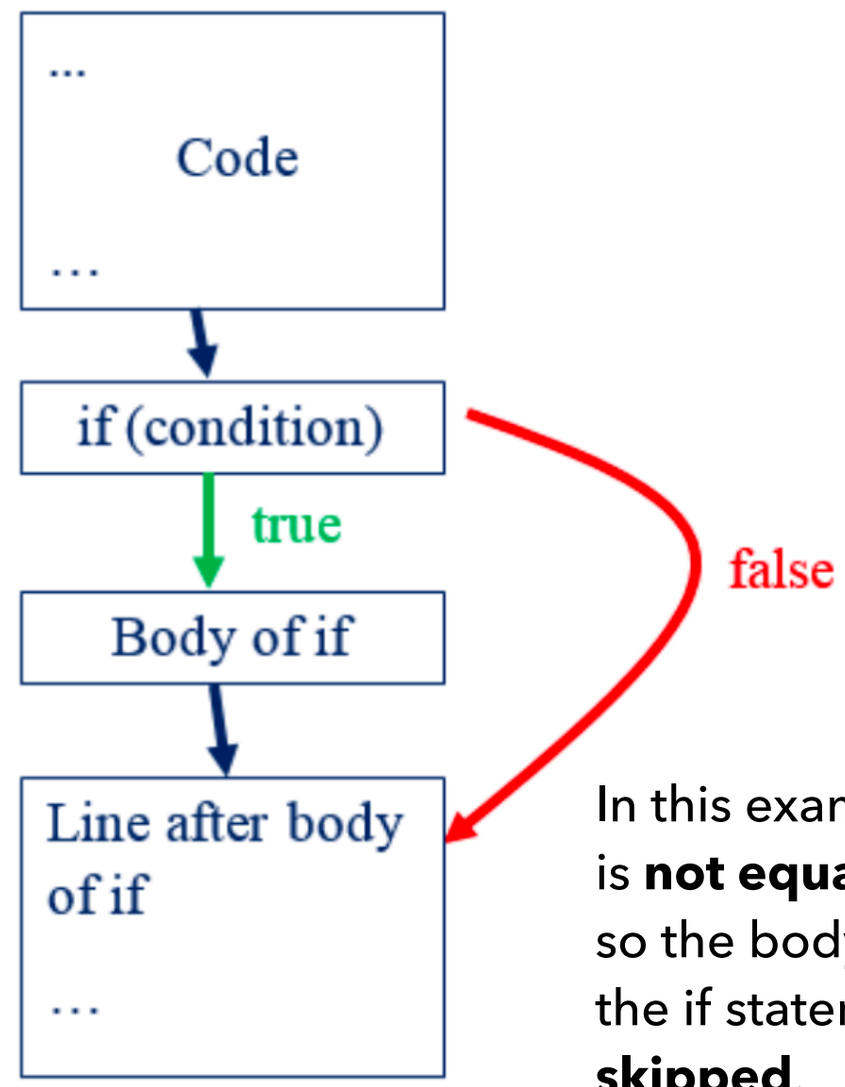
```
int x = 5;

if(x == 3) {
    System.out.println("equal!");
}

System.out.println("Hi there.");
```

Output:

```
Hi there.
```



In this example, x is **not equal** to 3, so the body of the if statement is **skipped**.

IF STATEMENT FLOW

- Every line of code will run until the conditional is hit. If the condition is met, the code inside will run. Else, Java will skip it.

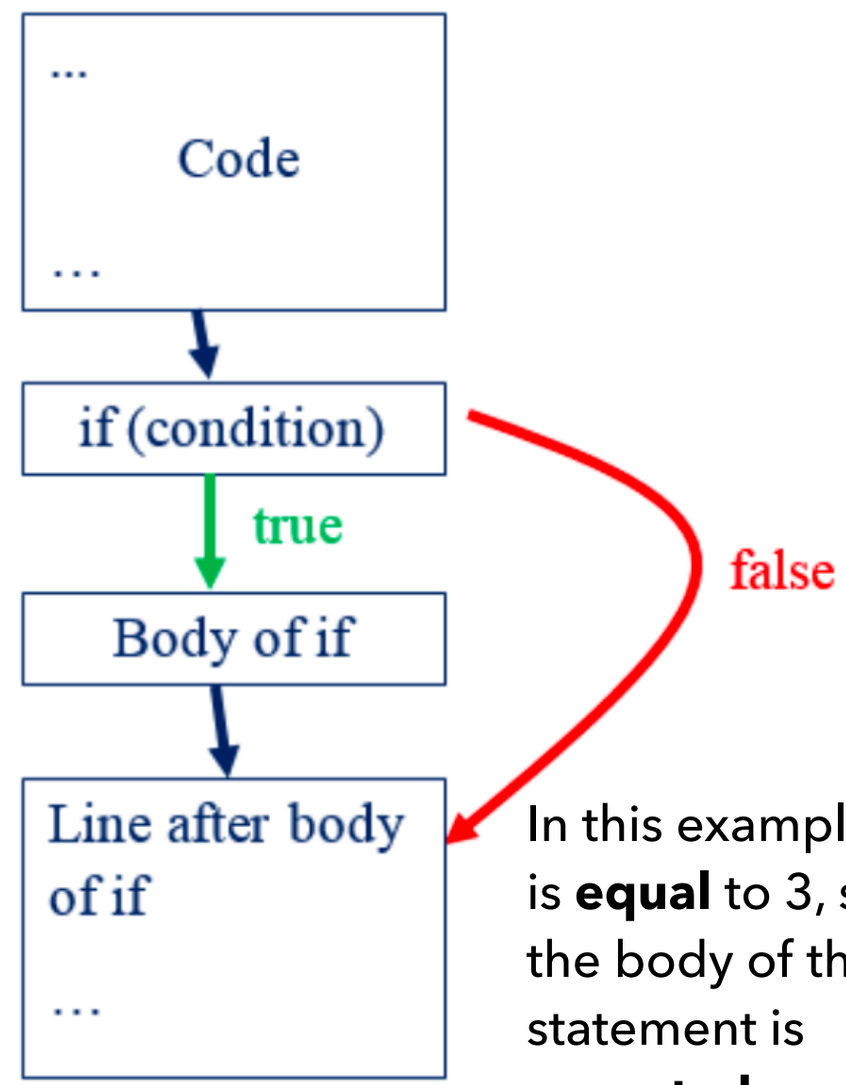
```
int x = 3;

if(x == 3) {
    System.out.println("equal!");
}

System.out.println("Hi there.");
```

Output:

```
equal!
Hi there.
```



In this example, x is **equal** to 3, so the body of the if statement is **executed**.

IF/ELSE STATEMENT

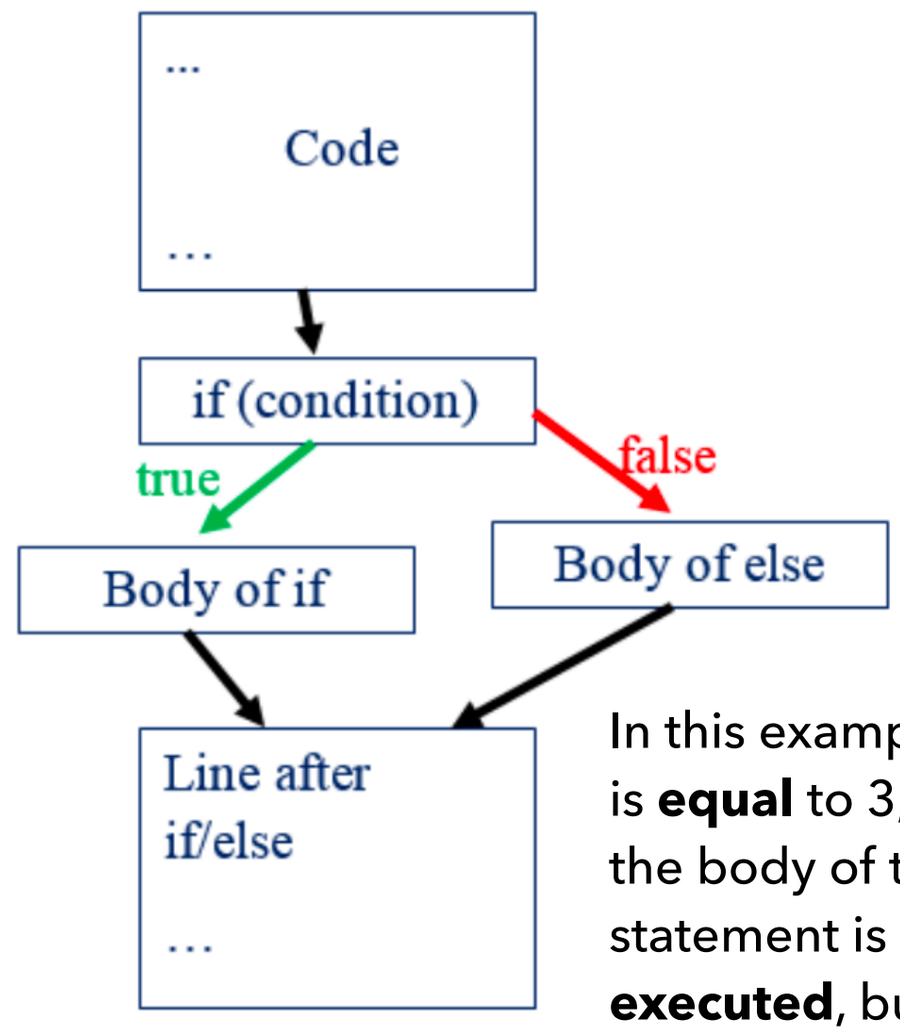
- What if we want to have a statement for when the first condition is **not met**? We can use the **else** statement along with the if.
- **NOTE:** You **CANNOT** use an else statement without an if statement first! An error will be thrown.
- The else statement will only run if the the condition above it **is not met**. If the condition above is met in the if statement. It will not run.
- Example…

IF/ELSE STATEMENT FLOW

- If the condition is **met** inside the first condition, the code inside will run and the code inside the else body will be **skipped**.

```java
int x = 3;

if(x == 3) {
    System.out.println("equal!");
} else {
    System.out.println("Not equal!");
}
```



Output:

```
equal!
```

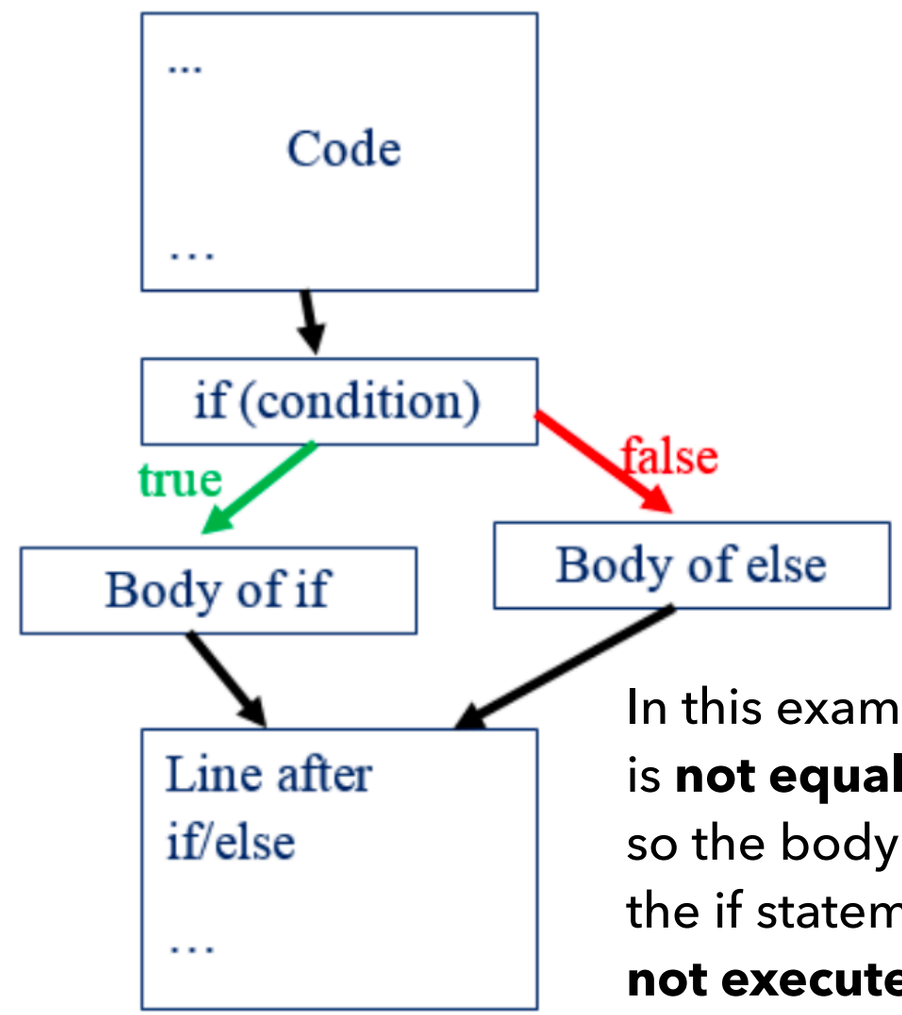
In this example, x is **equal** to 3, so the body of the if statement is **executed**, but the body of the else **is not**.

IF/ELSE STATEMENT FLOW

- If the condition is **not met** inside the first condition, the code inside will be skipped, and the code inside the else body will **run**.

```java
int x = 5;

if(x == 3) {
    System.out.println("equal!");
} else {
    System.out.println("Not equal!");
}
```



Output:

```
Not equal!
```

In this example, x is **not equal** to 3, so the body of the if statement is **not executed**, but the body of the else **is executed**.

IF/ELSE IF/ELSE STATEMENT

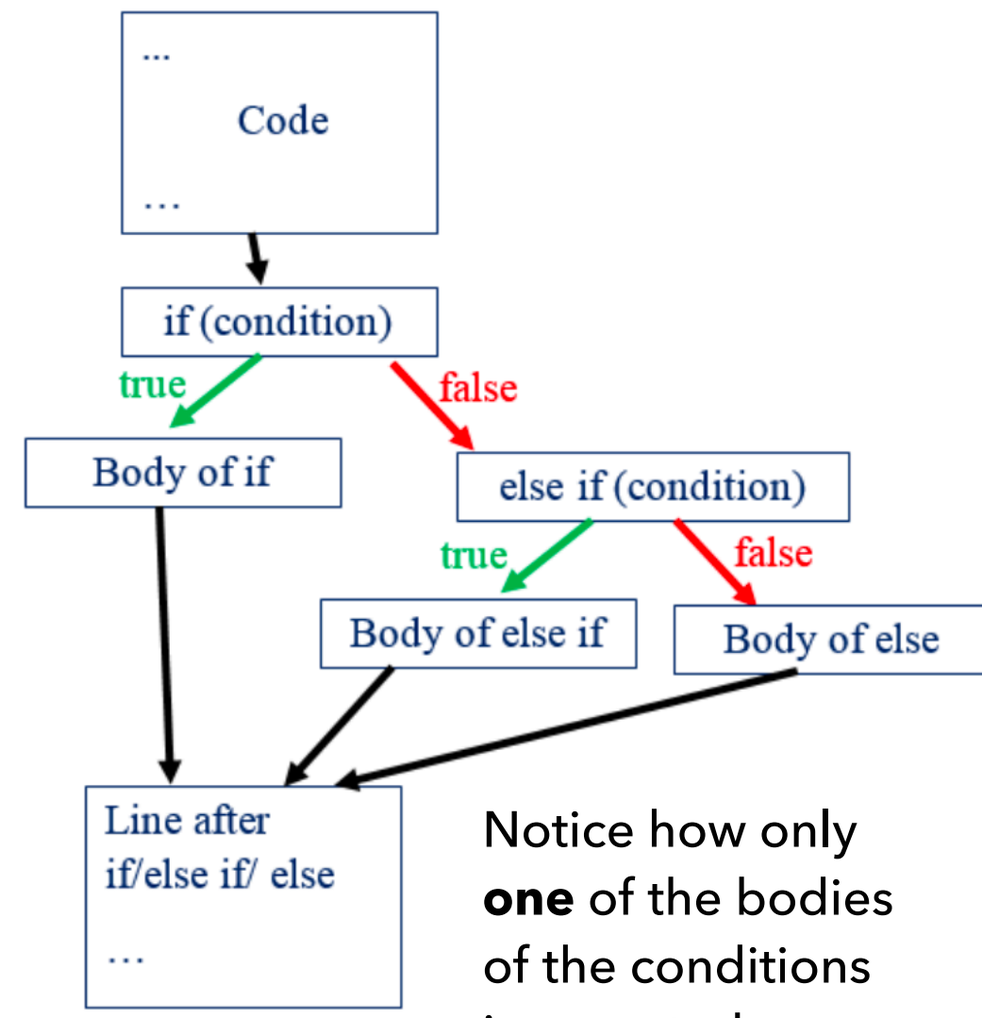
- What if we want to have several conditions to check for?? We can use the **else if** statement along with the if.
- **NOTE:** You **CANNOT** use an else if statement without an if statement first! An error will be thrown. However, you don't need an else statement.
- Like before, the else if statement will only run if the the condition above it **is not met**. If the condition above it is met, it will not run, and the rest of the else if/else conditional statements will be skipped.
- Example…

IF/ELSE IF/ELSE STATEMENT FLOW

- If the condition is **met** inside the first condition, the code inside will run and the code inside the rest of the conditional else if/else statements will be **skipped**. If the first condition is not met, the next condition will be checked, and so on and so on.

```
int x = 5;

if(x == 3) {
    System.out.println("x is equal to 3");
} else if(x == 4){
    System.out.println("x is equal to 4");
} else {
    System.out.println("x is not equal to 3 or 4");
}
```



Notice how only **one** of the bodies of the conditions is executed.

Output:

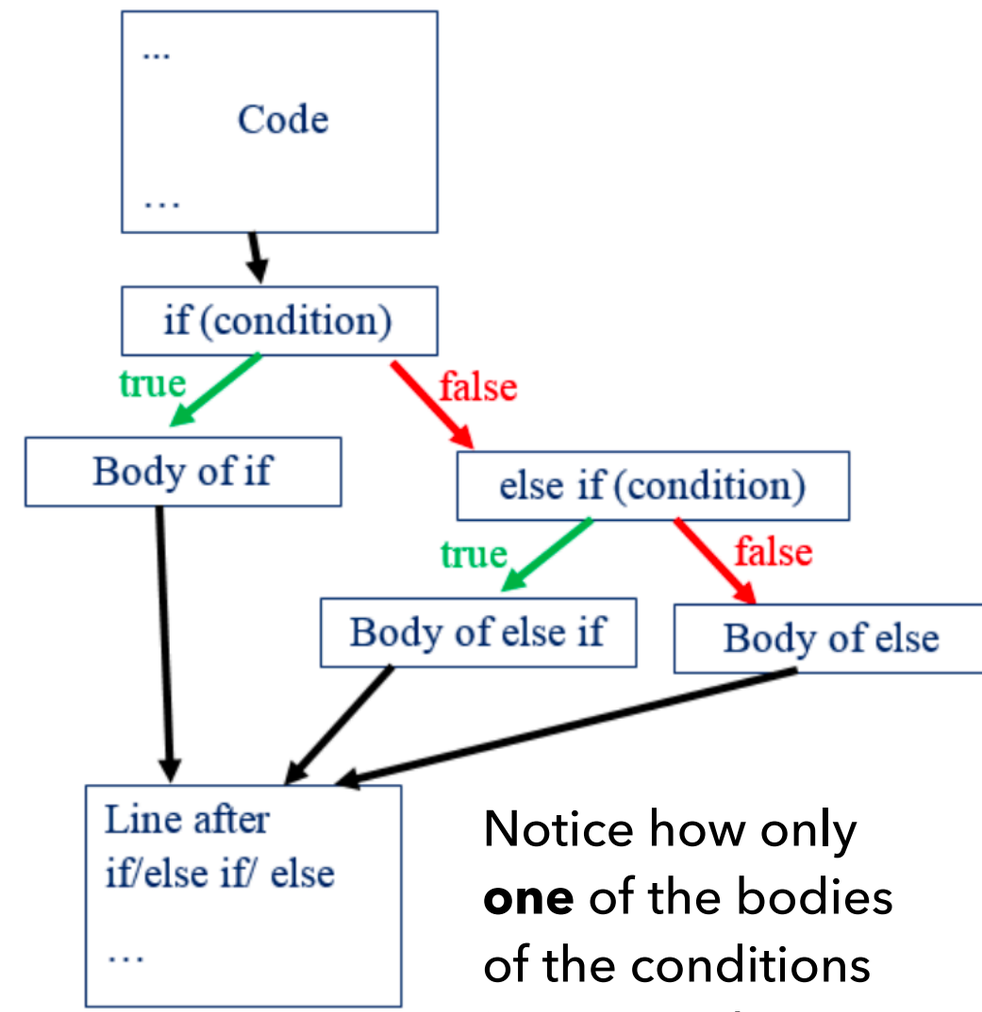
```
x is not equal to 3 or 4
```

IF/ELSE IF/ELSE STATEMENT FLOW

- If the condition is **met** inside the first condition, the code inside will run and the code inside the rest of the conditional else if/else statements will be **skipped**. If the first condition is not met, the next condition will be checked, and so on and so on.

```
int x = 4;

if(x == 3) {
    System.out.println("x is equal to 3");
} else if(x == 4){
    System.out.println("x is equal to 4");
} else {
    System.out.println("x is not equal to 3 or 4");
}
```



Output:

```
x is equal to 4
```

Notice how only **one** of the bodies of the conditions is executed.

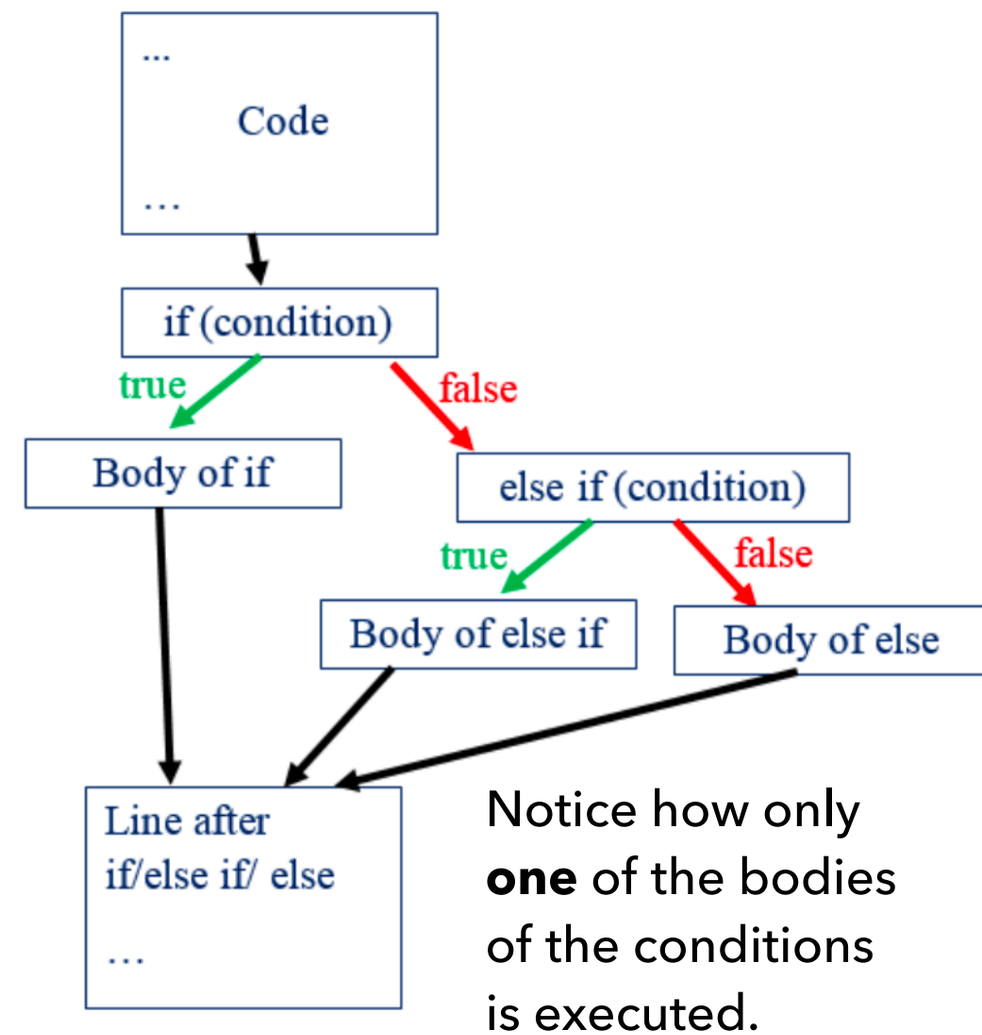
THE ELSE STATEMENT AGAIN

- The else statement is a great catchall for when none of the conditions are met! Even though it's not necessary, it's a good idea to include it.
- This is also an example of using a logical operator inside one of our conditions

```
int id = 223;
String occupation;

if(id > 300 && id < 500) {
    occupation = "software developer";
} else if(id < 300){
    occupation = "QA tester";
} else {
    occupation = "intern";
}

System.out.println("Occupation is " + occupation);
```



Notice how only **one** of the bodies of the conditions is executed.

Output:

```
Occupation is QA tester
```

OUR BMI QUESTION AGAIN

- How could we create a program that lets us output the health state of a person's BMI? (Lets say **height** is 64.5 inches and **weight** is 145 pounds)

A person's health state with respect to weight is roughly rated as follows:

BMI < 18.5: **Underweight**
18.5 ≤ BMI ≤ 25: **Normal**
25 < BMI ≤ 30: **Overweight**
BMI > 30: **Obese**

$$BMI = \frac{(\text{Weight in Pounds }) * 703}{(\text{Height in inches})^2}$$

Lets do this in Java

OUR BMI QUESTION AGAIN – BREAK IT UP IN STEPS

```java
// 1. determine variables needed
String healthState;
int weight = 145;
double height = 64.5;

// 2. solve for BMI:
double bmi = (weight * 703) / (height * height);

// 3. determine the health state
if(bmi < 18.5) {
    healthState = "Underweight";
} else if(bmi <= 25) {
    healthState = "normal";
} else if(bmi <= 30) {
    healthState = "Overweight";
} else {
    healthState = "Obese";
}

// 4. display the health state
System.out.printf("%s%.2f%s%s", "BMI: ", bmi, ", Health State: ", healthState);
```

Output:

```
BMI: 24.50, Health State: normal
```

BREAK THE PROBLEM DOWN

- Whenever you have a programming question, stop, and **write out** your plan on paper. This is what I like to do:

  1. Determine your variables needed and declare/initialize them
  2. If you need to solve some kind of equation, solve it first and store it in a variable.
  3. Write out your conditions and determine what should be **executed** if the condition is met.
  4. Display your result to check your work.

MORE EXAMPLES

- How do we check if a number is **positive**?
- How do we check if a number is **even**?
- How do we check if a number is between **0 and 10 inclusive** (inclusive means we include 0 and 10, exclusive means we exclude them).

NOTE: 0 is **NOT** positive and **NOT** negative. A positive number is a number *greater* than 0. A negative number is a number *less* than 0.

MORE EXAMPLES – THINK IT THROUGH

- How do we check if a number is **positive**?

  1. Determine our variables – need one for our number, one for our true/false
  2. Think about the logic: "A positive number is any number above 0, a negative number is any number below 0)."
  3. Create our conditional statements

MORE EXAMPLES – THINK IT THROUGH

- How do we check if a number is **positive**?

```
int num = 345;
boolean positive;

if(num > 0) {
    positive = true;
} else {
    positive = false;
}

System.out.println(num + " is positive: " + positive);
```

```
345 is positive: true
```

MORE EXAMPLES

- How do we check if a number is **even**?

  1. Determine our variables – need one for our number, one for our true/false
  2. Think about the logic: “a number is even if it is divisible by, odd if it is not.”
  3. Create our conditional statements

MORE EXAMPLES

- How do we check if a number is **even**?

```
int num = 345;
boolean even;

if(num % 2 == 0) {
    even = true;
} else {
    even = false;
}

System.out.println(num + " is even: " + even);
```

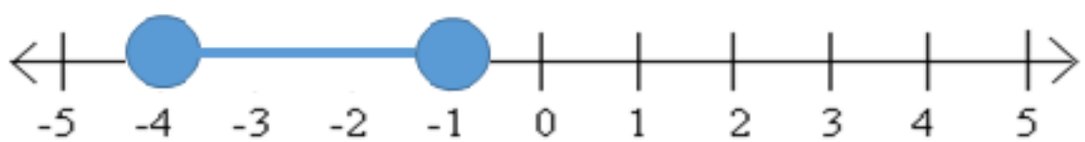
```
345 is even: false
```

MORE EXAMPLES

- How do we check if a number is between **-4 and -1 inclusive** (inclusive means we include -4 and -1, exclusive means we exclude them).
  - For intervals, we need to use logical operators (&&, | |)

In Math: $\mathbf{-4 \leq x \leq -1}$

- Visually:



- Interval Notation: **[–4,–1]**
- In Java we do this by:

```
(x>=-4)  &&  (x<=1)
```

MORE EXAMPLES

- How do we check if a number is between **-4 and -1 inclusive** (inclusive means we include -4 and -1, exclusive means we exclude them).

```
int num = -3;
boolean b;

if(num >= -4 && num <= -1) {
    b = true;
} else {
    b = false;
}

System.out.println(num + " is between -4 and -1 inclusive: " + b);
```

```
-3 is between -4 and -1 inclusive: true
```

TRY IT YOURSELF

- How can we give feedback based off the following (let's say the grade we are checking is 82):

Letter Grade	Number Grade
A	90-100
B	80-89.9
C	70-79.9
D	60-69.9
F	<=59.9

TRY IT YOURSELF

- How can we give feedback based off the following (let's say the grade we are checking is 82):

```
double grade = 82;
char letter;

if(grade <= 100 && grade >= 90) {
    letter = 'A';
} else if(grade >= 80) {
    letter = 'B';
} else if(grade >= 70) {
    letter = 'C';
} else if(grade >= 60) {
    letter = 'D';
} else {
    letter = 'F';
}

System.out.println("Letter Grade is " + letter);
```

Output:

```
Letter Grade is B
```

SIMPLIFYING CODE FOR MULTIPLE CONDITIONS

- A **switch statement** executes statements based on the value of a variable or an expression.
- Java provides a switch statement to simplify coding for multiple conditions.

```
switch(variable)
{
    case value1: //statements
            break;
    case value2: //statements
            break;
    case value3: //statements
            break;
    .
    .
    .
    default: //statements
            break;
}
```

SIMPLIFYING CODE FOR MULTIPLE CONDITIONS

- **Example:** Suppose we want to convert the letter of the day of the week to word form:

```
char day='t';
switch(day)
{
    case 'm': System.out.println("Monday");
              break;
    case 't': System.out.println("Tuesday");
              break;
    case 'w': System.out.println("Wednesday");
              break;
    case 'r': System.out.println("Thursday");
              break;
    case 'f': System.out.println("Friday");
              break;
    case 's': System.out.println("Saturday");
              break;
    default: System.out.println("Sunday");
             break;
}
```

SIMPLIFYING CODE FOR MULTIPLE CONDITIONS

- **Example 2:** Suppose we want to use a switch statement to determine if a variable x is even or odd (in this case, an if/else statement would suffice):

```
int x=10;
switch(x%2)
{
    case 0: System.out.println("even");
            break;
    default: System.out.println("odd");
            break;
}
```

SIMPLIFYING CODE FOR MULTIPLE CONDITIONS

- **Example 3:** What do you think the **output** will be?

```
int y=9;
switch(y)
{
    case 1:
    case 2:
    case 3: System.out.println("Hello");
            break;
    case 4: System.out.println("4");
    case 5:
    case 9:
    case 7: System.out.println("ABC");
            break;
    default: System.out.println("Goodbye");
             break;
}
```

SIMPLIFYING CODE FOR MULTIPLE CONDITIONS

- **Example 3:** What do you think the **output** will be?

```
int y=9;
switch(y)
{
    case 1:
    case 2:
    case 3: System.out.println("Hello");
            break;
    case 4: System.out.println("4");
    case 5:
    case 9:
    case 7: System.out.println("ABC");
            break;
    default: System.out.println("Goodbye");
             break;
}
```

Output:

ABC