ICS 111
Introduction to Computer Science I

Nikki Manuel
Leeward Community College

`if`, `if-else`, and `switch`

Week 5
Fall 2019

`if` Statements

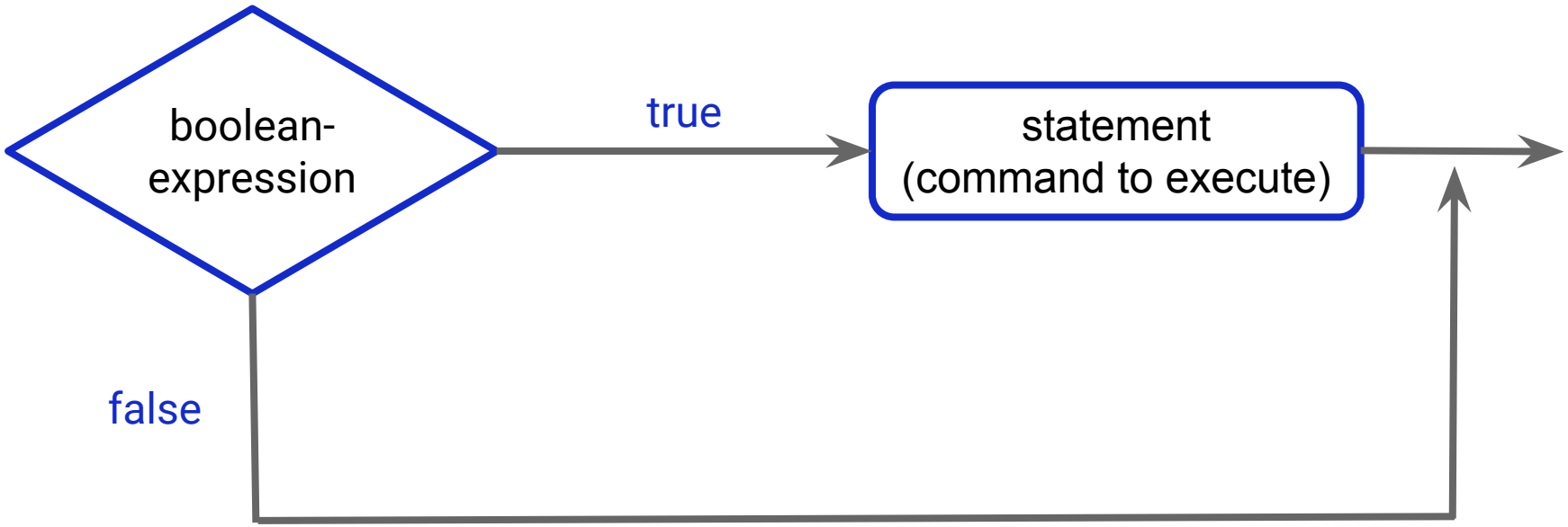
`if` Statements & Syntax

An **`if`** statement is a construct that enables a program to specify alternative paths of execution.

It executes an action if, and only if, the condition is **`true`**.

```
if (boolean-expression) {
   // do this code if the boolean expression = true
}
```

`if` Statement Flowchart



`if` Statements & Syntax

An `if` statement is made up of three parts:

1. The keyword `if`
2. The condition -- a boolean expression
3. Statements to be executed if the condition is true

```
if (boolean-expression) {
    // do this code if the boolean expression = true
}
```

Remember this? Comparison Operators

Comparison	*Symbol*
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=
Equal	==
Not equal	!=

(from the Week 2 slides)

Example: Using `if`

```
Scanner reader = new Scanner(System.in);

double moneyInWallet = 0;    // How much you have
double billTotal = 0;        // The bill total

System.out.print("What is the bill total? ");
billTotal = reader.nextDouble();
System.out.print("How much money ya got? ");
moneyInWallet = reader.nextDouble();

if (billTotal > moneyInWallet) {
    System.out.println("You in trouble.");
}
```

Example: Using `if`

```
if (billTotal > moneyInWallet) {
    System.out.println("You in trouble.");
}
```

The condition (a boolean expression):

```
billTotal > moneyInWallet
```

The statement to be executed:

```
System.out.println("You in trouble.");
```

`if` Statements : Coding Standard

one space before and after

```
if (boolean-expression) {
    // do this code if the boolean expression = true
}
```

code inside the `if` is indented

`if` and the closing bracket line up

`if` Statements : Coding Standard

one space before and after comparison operator

```
if (billTotal > moneyInWallet) {
    System.out.println("You in trouble.");
}
```

Test Yourself: Using `if`

Write an `if` statement that assigns 1 to `x` if `y` is greater than 10.

Write an `if` statement that prints “You get an A!” if `score` is greater than 90.

Write an `if` statement that asks a user for an integer and if the user’s number is even, prints “Your number is even.”

`if-else` Statements

`if-else` Statements

An **`if-else`** statement decides the execution path based on whether the condition is true or false.

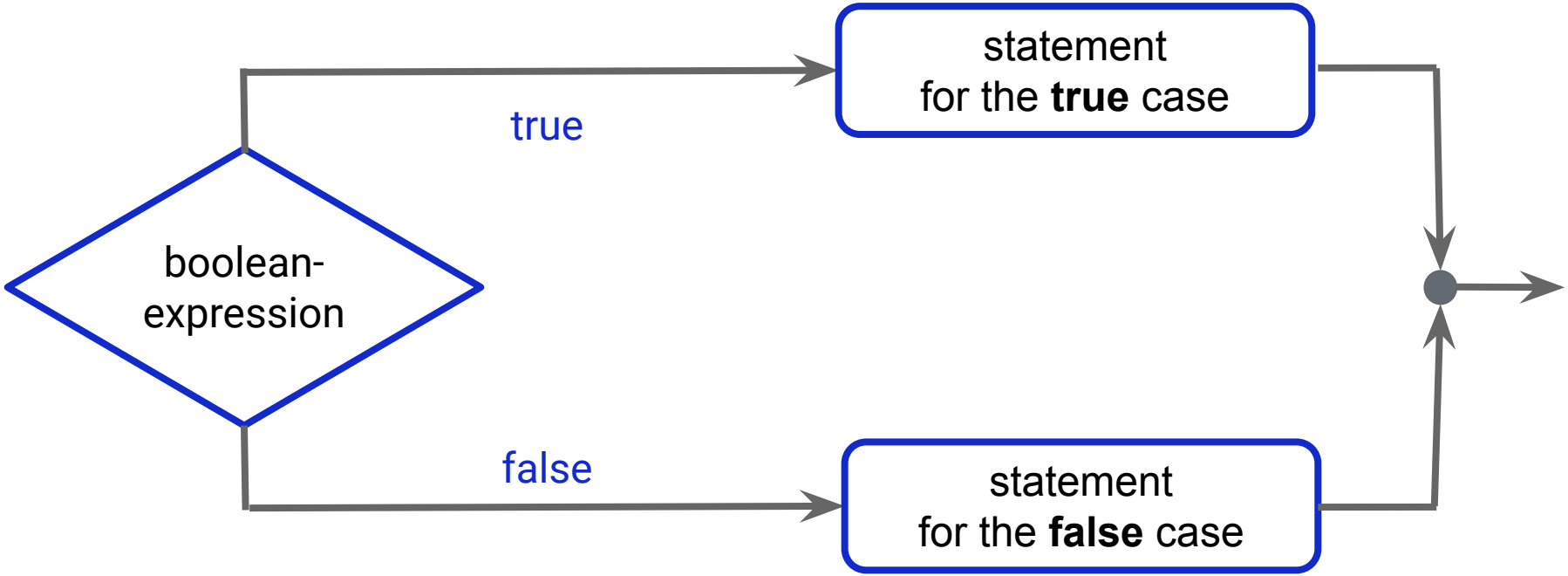
It executes one action if the condition is **`true`**.
It executes a different action if the condition is **`false`**.

`if-else` Syntax

Executes one action if the condition is **`true`**.
Executes a different action if the condition is **`false`**.

```
if (boolean-expression) {
    // do this code if the boolean expression = true
}
else {
    // do this code if the boolean expression = false
}
```

`if-else` Statement Flowchart



Example: Using `if-else`

```
if (billTotal > moneyInWallet) {
    System.out.println("You in trouble.");
    double moMoney = billTotal - moneyInWallet;
    System.out.println("You need $" + moMoney);
}
else {
    System.out.println("You good.");
}
```

Test Yourself: Using `if-else`

Write an **`if`** statement that increases **`pay`** by 500 if **`overtimeHours`** is greater than 10, otherwise increase **`pay`** by 50.

Write a program that generates a random integer and asks a user to guess the number, with an **`if`** statement that prints “Correct!” if the user’s guess is correct and otherwise prints “Sorry, incorrect!”

Nested `if` Statements

Nested `if` Statements

What if you want to check more than one condition?

An `if` or `if-else` statement can be inside another `if` statement to form a **nested** `if` statement.

There is no limit to the depth of nesting.

Example: Using Nested `if` Statements

```
if (a > x) {
    if (b > x) {
        System.out.println("a and b are greater than x");
    }
    else {
    System.out.println("a is greater than x, b is not");
    }
}
else {
    System.out.println("a is less than or equal to x");
}
```

Test Yourself: Nested `if` Statements

Suppose `x = 2` and `y = 3`. What is the output, if any?
What if `x = 3` and `y = 2`? What if `x = 3` and `y = 3`?

```
if (x > 2) {
    if (y > 2) {
        int z = x + y;
        System.out.print("z is " + z);
    }
}
else
    System.out.print("x is " + x);
```

Multi-Way `if-else` Statements

Multi-Way `if-else` Statements

You can also nest **`if-else`** statements in the else statement!

```
if (boolean-expression) {
    // do this code if the boolean expression = true
}
else {
    if (boolean-expression) {
        // do this if true
    }
    else {
        // otherwise do this
    }
}
```

Example: Using Multi-Way `if` Statements

```
if (grade >= 90.0)
    System.out.print("A");
else
    if (grade >= 80.0)
        System.out.print("B");
    else
        if (grade >= 70.0)
            System.out.print("C");
        else
            if (grade >= 60.0)
                System.out.print("D");
            else
                System.out.print("F");
```

BETTER example: Multi-Way `if` Statements

```
if (grade >= 90.0)
    System.out.print("A");
else if (grade >= 80.0)
    System.out.print("B");
else if (grade >= 70.0)
    System.out.print("C");
else if (grade >= 60.0)
    System.out.print("D");
else
    System.out.print("F");
```

This is the preferred format.

- avoids deep indentation
- easier to read

Note that a condition is tested only when all of the conditions that come before it are **`false`**.

Test Yourself: Multi-Way `if` Statements

What is wrong with the following code?

```
if (grade >= 60.0)
    System.out.print("D");
else if (grade >= 70.0)
    System.out.print("C");
else if (grade >= 80.0)
    System.out.print("B");
else if (grade >= 90.0)
    System.out.print("A");
else
    System.out.print("F");
```

Logical Operators

Logical Operators (aka Boolean operators)

Logical operators can combine conditions to form a compound Boolean expression.

For example:

IF you are hungry AND you like french fries
    Go to McDonald's

IF (you're hungry OR you're thirsty) AND you <u>don't</u> like french fries
    Go to Taco Bell

Logical Operators (aka Boolean operators)

Operator	*Name*
!	not
&&	and
||	or

Logical Operators (aka Boolean operators)

IF you are hungry && you like french fries

Go to McDonald's

IF (you're hungry || you're thirsty) && !(you like french fries)

Go to Taco Bell

Truth Table: `&&` (AND)

`X`	`Y`	`X && Y`
True	True	**True**
True	False	False
False	True	False
False	False	False

Example: Using `&&`

Let's go back to our first example:

```
if (billTotal > moneyInWallet) {
    System.out.println("You in trouble.");
}
else {
    System.out.println("You good.");
}
```

What about this: If you have friends, then they'll help cover your bill.
But if you don't have enough money and you don't have friends,
you in trouble.

Example: Using `&&`

If you don't have enough money and you don't have friends, then you in trouble.

How do we translate that into Java code?

- You don't have enough money:

```
billTotal > moneyInWallet
```

- You don't have friends:

```
boolean haveFriends = false;
haveFriends == false
```

Example: Using `&&`

If you have friends, then they'll help cover your bill.
But if you don't have enough money and you don't have friends,
you in trouble.

```
if (billTotal > moneyInWallet) && (haveFriends == false) {
    System.out.println("You in trouble.");
}
else {
    System.out.println("You good.");
}
```

Truth Table: `||` (OR)

`X`	`Y`	`X || Y`
True	True	**True**
True	False	**True**
False	True	**True**
False	False	False

Example: Using ||

If the bill is less than or equal to the amount of money you have in your wallet, or you have friends to cover you, then you're good.

How do we translate that into Java code?

- The bill is less than or equal to the amount of money in your wallet:

```
billTotal <= moneyInWallet
```

- You have friends:

```
haveFriends == true
```

Example: Using ||

If the bill is less than or equal to the amount of money you have in your wallet, or you have friends to cover you, then you're good.

```
if (billTotal <= moneyInWallet) || (haveFriends == true) {
    System.out.println("You good.");
}
else {
    System.out.println("You in trouble.");
}
```

Important!: Incompatible Operands

In math, this is okay:

```
1 <= numberOfDaysInMonth <= 31
```

In Java, it isn't okay!

1 <= numberOfDaysInAMonth is evaluated first as a boolean value.
Then, Java would try to compare: (true/false) <= 31

Instead, write it like this:

```
(1 <= numberOfDaysInMonth) && (numberOfDaysInMonth <= 31)
```

Test Yourself: Logical Operators

If **`x`** is **`1`**, state the result of the following Boolean expressions:

```
(true) && (3 > 4)
```

```
!(x > 0) && (x > 0)
```

```
(x > 0) || (x < 0)
```

```
(x != 0) || (x == 0)
```

```
(x >= 0) || (x < 0)
```

```
(x != 1) == !(x == 1)
```

Short Circuit Evaluation

The `&&` and `||` are short-circuit operators.

The evaluation stops when the condition is guaranteed.

What...?

Short Circuit

`&&`

`&&` : All criteria must be `true` !

Let's say you have:

```
criteria_1 && criteria_2 && …
```

If criteria_1 is false, then why bother evaluating the rest?

If criteria_2 is false, then why bother evaluating the rest?

Short Circuit `&&`

Once a condition is false, the rest of it can be ignored.

For example:

```
false && some_really_complex_expression_with_lots_of_parentheses
```

```
false && (true || false && false || false && true)
```

These will not be evaluated because we already know the condition is guaranteed to be false.

Short Circuit
||

|| : At least 1 must be **`true`** !

Let's say you have:

```
criteria_1 || criteria_2 || …
```

If criteria_1 is true, why bother with the rest?

Short Circuit ||

Once a condition is true, the rest of it can be ignored.

For example:

```
true || some_really_complex_expression_with_lots_of_parentheses

true || (true || false && false || false && true)
```

These will not be evaluated because we already know the condition is guaranteed to be true.

Short Circuit Evaluation

Keep parentheses in mind.

`&&` has a higher priority than `||`

Test Yourself: Evaluate These Conditions

```
((false && true) || true) && false
```

```
true || (false && false)
```

```
true || true || true && false
```

```
false || true && false || false
```

Test Yourself: Evaluate These Conditions

```
((false || true) && (false && false))
```

```
false || (true || false && true)
```

```
true && (false || true || true)
```