Conditionals

Overview

Like humans, programs should be able to make decisions based on *conditions*

- *Conditions* are the states of the data in your program
  - Values stored in variables (*today*)
  - User input like mouse clicks, key presses (*soon*)
- The program will decide to execute some code if a condition is `true` and another part if it is `false`
  - "if the light is green *walk* else *stop*"

Learning Objectives

- To be able to create and evaluate boolean expressions
- To be able to use if statements to control the flow of a program
- To be able to use if-else statements to control the flow of a program

Building Blocks of Boolean Expressions

Boolean expressions evaluate to `true` and `false` and allow us to test conditions.

Relational operators (less than, equals to, greater than, etc.) are used in boolean expressions to compare numeric values or arithmetic expressions

- `compareTo()` and `equals()` methods are used to compare String variables

Logical operators (`&&`, `||`, `!`) are used to combine boolean expressions to build more complex and detailed boolean expressions.

The Boolean Expression Toolkit

Relational Operators

Operator/method	Input Types	Description
`< / <=`	`int & double`	less than / less than or equal to
`> / >=`	`int & double`	greater than / greater than or equal to
`== / !=`	`int, double, boolean`	equal to / not equal to
`.equals()`	`String`	equal to
`.compareTo()`	`String`	returns -ve, `0`, or +ve `int`, not a `boolean`!

The Boolean Expression Toolkit

Logical Operators

Operator/method	Input Types	Description
`&&`	`boolean`	logical "and", evaluates to `true` only if both inputs are `true`
`||`	`boolean`	logical "or", evaluates to `true` as long as at least one input is `true`
`!`	`boolean`	logical "not", negates a single `boolean` value to its opposite

Truth Tables

P	Q	P && Q	P || Q	!P
`true`	`true`	`true`	`true`	`false`
`false`	`true`	`false`	`true`	`true`
`true`	`false`	`false`	`true`	`false`
`false`	`false`	`false`	`false`	`true`

Poll Time!

Conditionals

- So far, programs have always executed one statement after another, top to bottom.
- Conditionals allow us to **control the flow of a program** based on the values in the program
    - We say that the `if` statement is a **flow control structure**

If statement

The `if` statement:

- Evaluates a `boolean` expression
- If `true`, executes some statements
- If `false`, skips those statements

> “Choose whether or not to execute a set of statements.”

Flow of Control Using `if`

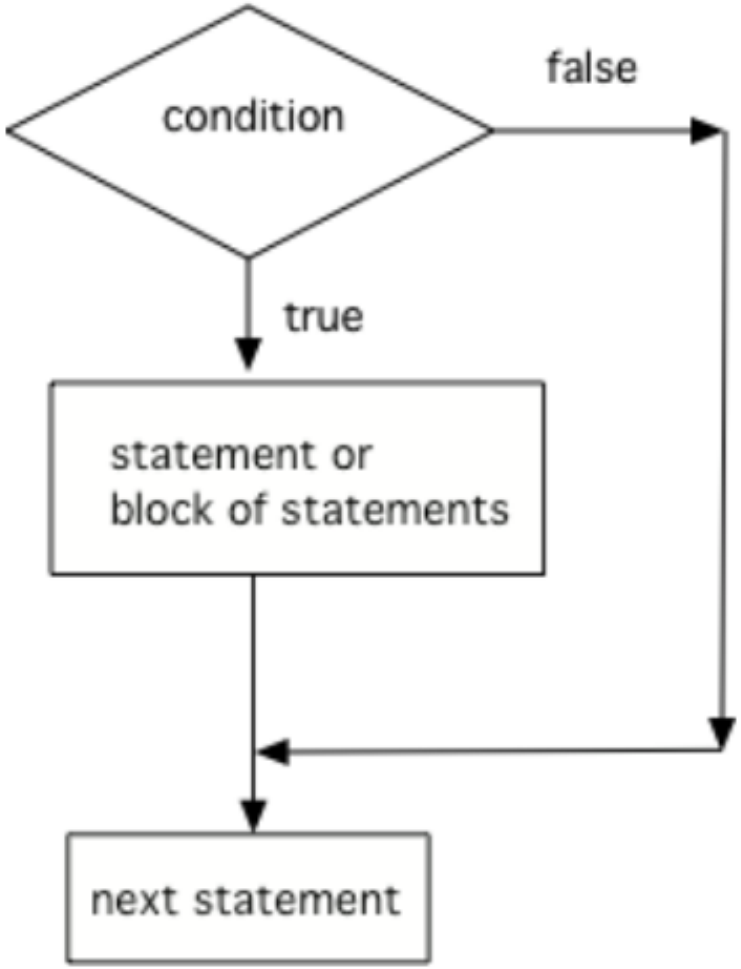


Figure 2: The order that statements execute in a conditional

Code Blocks

Code blocks are associated groups of statements that are executed together and that have the same level of *scope*.

- Curly Braces (`{}`) denote the start and end of code blocks.
- Scope refers to the region of the program where variables are able to be accessed after declaring

Structure of If statement

```
// a single if statement
if (4 < 5 && !"Harry".equals("Smith")) { // start a new code block
  System.out.println("Drawing a circle if condition is true");
  PennDraw.circle(0.5, 0.5, 0.1);
}
// statements outside of if are run no matter what!
System.out.println("Drawing a square no matter what.");
PennDraw.square(0.5, 0.5, 0.1);
```

Exercise: What Gets Printed?

```
public static void main(String[] args) {
   int age = 16;
   if(age > 17) {
      System.out.println("Eligible to vote");
   }
}
```

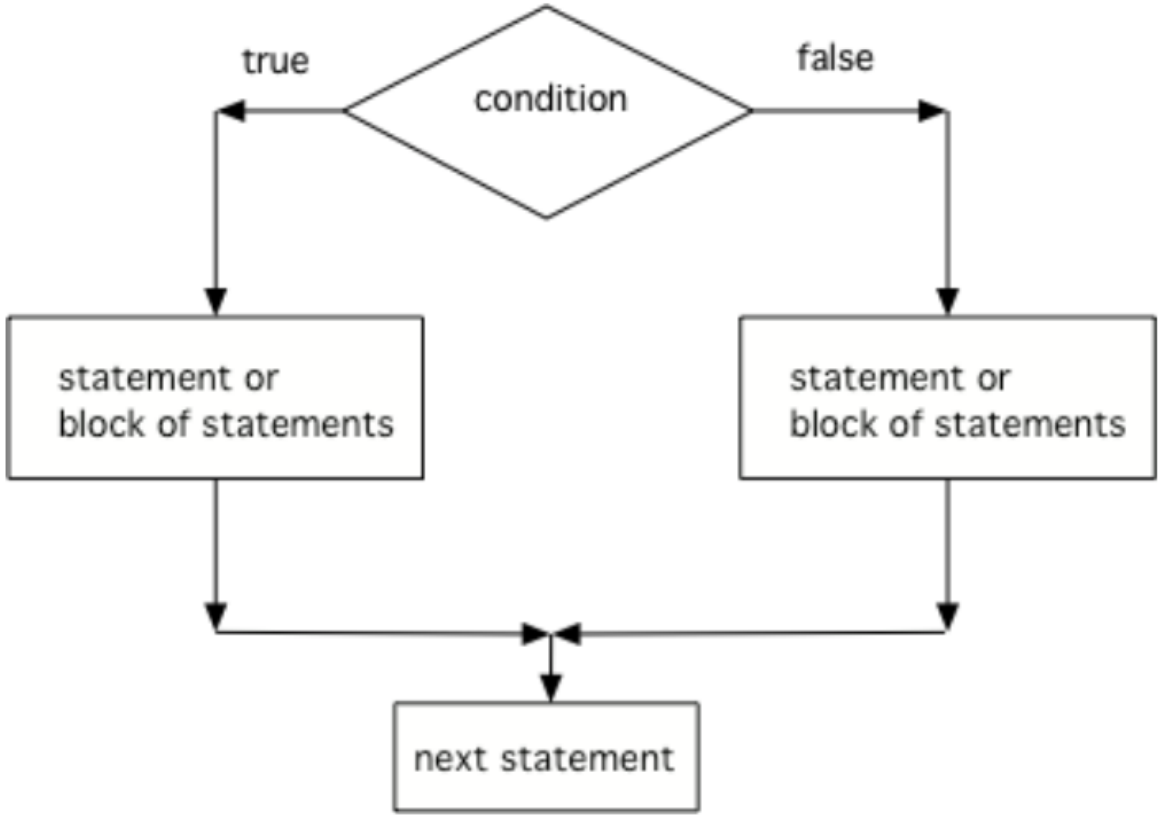
Exercise: What Gets Printed?

```
if (true) {
  System.out.println("Apple");
}
if (10 > 10) {
  System.out.println("Banana");
}
if (10 >= 10) {
  System.out.println("Cherry");
}
```

Poll Time!

The `else` Statement

"Which **one** of these two options should I pick?"



The order that statements execute in a conditional with 2 options: if and else

Structure of If-else statement

```
if (4 < 5 && !"Harry".equals("Smith")) {
  // start a new code block to run if condition is true
  System.out.println("Drawing a circle if condition is true");
  PennDraw.circle(0.5, 0.5, 0.1);
} else {
  // start a new code block to run if condition is false
  System.out.println("Drawing a line if condition is false");
  PennDraw.line(0, 0, 1, 1);
}
// statements outside of if-else are run no matter what!
System.out.println("Drawing a square no matter what.");
PennDraw.square(0.5, 0.5, 0.1);
```

Poll Time!

Nested if statements

The body of a conditional contains a sequence of statements

The **if** statement is, itself, a statement!

- So: you can put a conditional inside of another.
- “Only If **X** is true, then I’ll check if **Y** is true…”

Nested if statements

```
public static void main(String[] args) {
  if (expression A) {
    if (expression B) {
      // run when A is true and B is true
    }
    // run when A is true regardless of B
  } else {
    // run when A is false regardless of B
  }
}
```

Nested if statements

Follow the curly braces to figure out which “if” the “else” belongs to!

(in this case, it’s the first one)

```java
public static void main(String[] args) {
  if (expression A) {
    if (expression B) {
      // run when A is true and B is true
    }
    // run when A is true regardless of B
  } else {
    // run when A is false regardless of B
  }
}
```

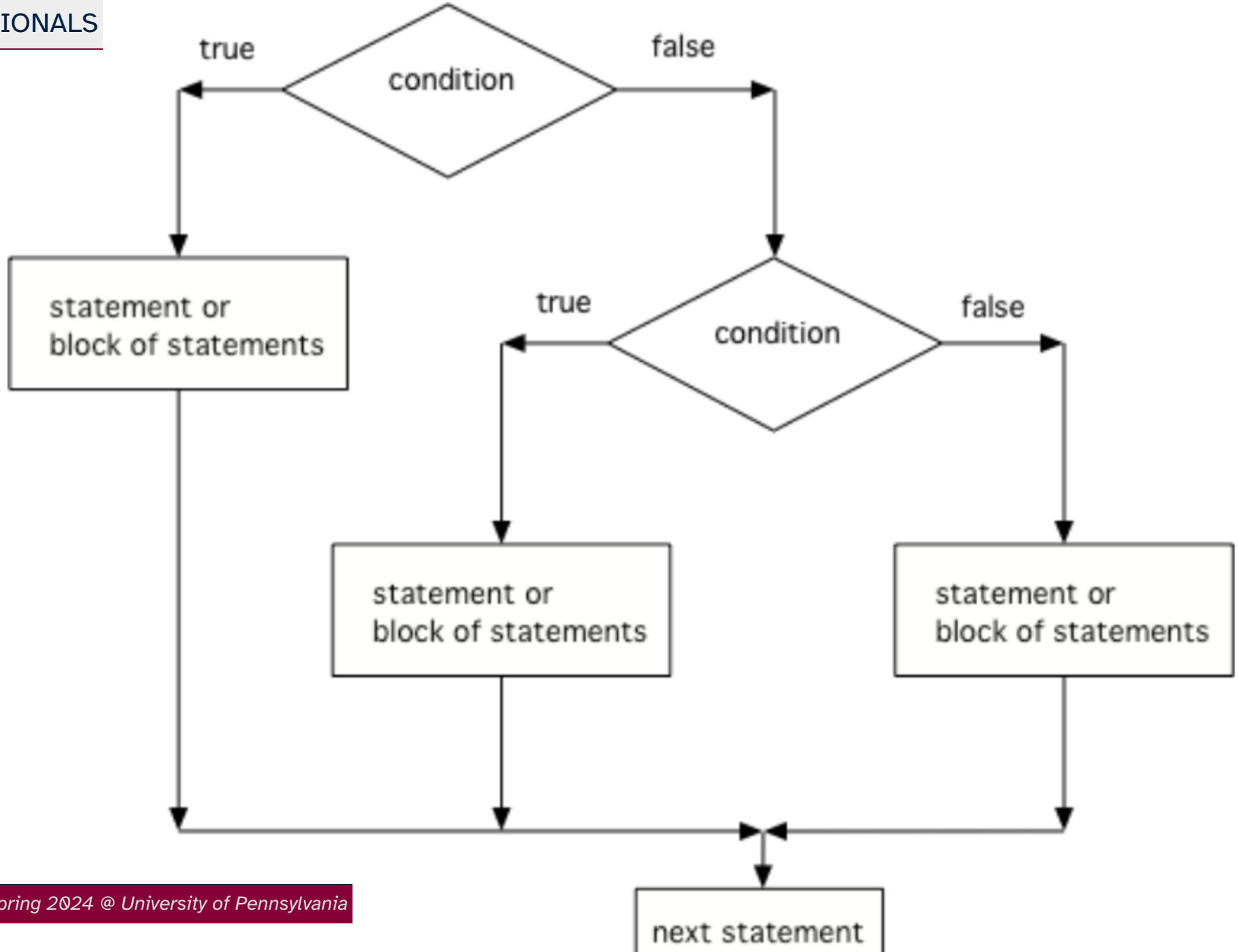
Poll Time!

`if-else if-else` statements

A conditional with three or more mutually exclusive options

Of the statement blocks, exactly one will execute

- (if you leave off the last **else**, then exactly 0 or 1 will execute)


true
condition
false
statement or
block of statements
true
condition
false
statement or
block of statements
statement or
block of statements
next statement

Can you go to see your friend at the park?

```
boolean isNearby = true;
boolean haveHomework = true;
if(!isNearby) {
  System.out.println("no, too far");
} else if (haveHomework) {
  System.out.println("no, do HW");
} else {
  System.out.println("yes, go see them");
}
```

The Grammar of the Conditional

A conditional statement consists of one essential part—the `if`—and several optional parts.

1. Begin with an `if` statement. The `if` statement must include a boolean expression to test.
2. Optionally, include any number of `else if` statements. Each `else if` statement must include a boolean expression to test. Any conditional may include zero or more `else if` statements.
3. Optionally, include an `else` statement. The `else` statement does not include a boolean expression to test. Any conditional may include zero or one `else` statements.

Some Examples

```
if (x > y && x > z) {
  System.out.println("x is the largest");
} else if (y > x && y > z) {
  System.out.println("y is the largest");
} else if (z > x && z > y) {
  System.out.println("z is the largest");
} else {
  System.out.println("two or more variables are tied for largest");
}
```

```
if (a > b && a % b == 0) {
  System.out.println("a is divisible by b");
} else if (b > a && b % a == 0) {
  System.out.println("b is divisible by a");
}
```

Live Coding: Parking Sign