Conditionals (*if*)

Overview

- Like humans, programs should be able to make decisions based on conditions
- In this module, we will learn how to express conditions in a program
- The program will decide to execute some code if a condition is `true` and another part if it is `false`
- Example:
    - `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

Boolean expressions

- Boolean expressions evaluate to `true` and `false`
- **Relational operators** (less than, equals to, greater than, etc.) are used in `boolean` expressions
- Relational operators compare numeric values or arithmetic expressions
- *`compareTo()`* and *`equals()`* methods are used to compare String variables

Relational Operators

- Relational operators are **binary operators** (they take two operands)

```
operand1  operator  operand2
```

○	< Less Than	5	<	7	➔ true
○	> Greater Than	6	>	1	➔ true
○	<= Less than or equal to	6	<=	1	➔ false
○	>= Greater than or equal to	3	>=	5	➔ ??
○	== Equals	4	==	4	➔ ??
○	!= Does not equal	0	!=	10	➔ ??

Logical expressions

- Return a `boolean` value
- Uses **logical operators**
  - `&&` logical conjunction (**and**)
    - both expressions must be true for conjunction to be true
  - `||` logical disjunction (**or**)
    - either expression must be true for disjunction to be true
  - `!` logical negation (**not**)
    - True → false, false → true

Truth table

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

Compound Boolean Expressions

```
int score = 10;
if (score < 0 || score > 100)
{
    System.out.println("Score has an illegal value.");
}
if (score >= 0 && score <= 100)
{
    System.out.println("Score is in the range 0-100");
}
```

false

false false

true

true true

Conditionals

- Programs execute one statement after another
- Conditionals allow us to **control the flow of a program**
- `If statement` is a **flow control structure**

`If` statement

- The `if` statement
  - Evaluate a `boolean` expression
  - If `true`, execute some statements
  - If `false`, skip those statements

`If` statement flowchart

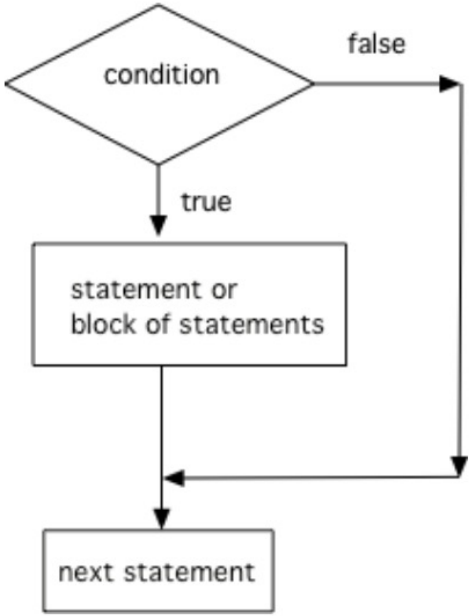


Figure 2: The order that statements execute in a conditional

Structure of `If` statement

```
// A single if statement
if (boolean expression)
    Do statement;
// Or a single if with {}
if (boolean expression)
{
   Do statement;
}
// A block if statement: { } required
if (boolean expression)
{
   Do Statement1;
   Do Statement2;
   ...
   Do StatementN;
}
```

Boolean expressions and `if` statement

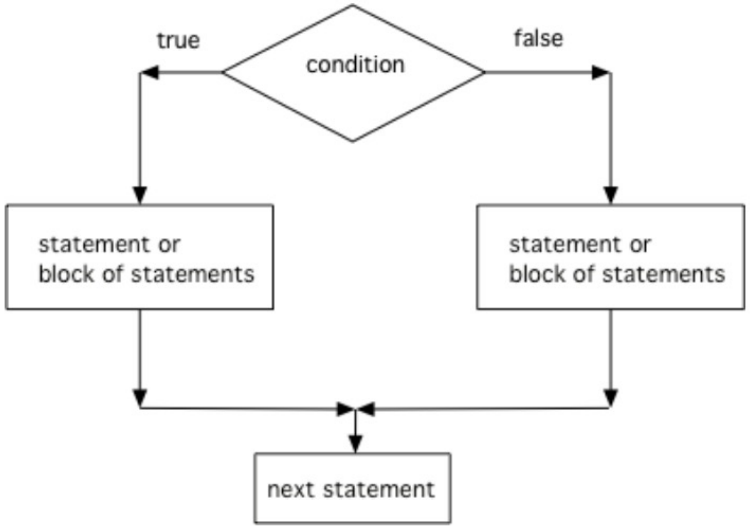
- **Boolean expressions** use **relational operators** (==, !=, <, >)

```
int age = 16;
if(age > 17) {
  System.out.println(“Eligible to vote”);
}
```

```
if ( true ) { System.out.println(”Apple”); }
if ( 10 > 10 ) { System.out.println(“Banana”); }
if ( 10 >= 10 ) { System.out.println(“Orange”); }
```

`If-else` statement

- A conditional with two options



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

Structure of `If-else` statement

```
// A block if/else statement
if (boolean expression)
{
   statement1;
   statement2;
}
else
{
   do other statement;
   and another one;
}
```

```
// A single if/else statement
if (boolean expression)
    Do statement;
else
    Do other statement;
```

Can you go to see your friend at the park?

```
boolean isNearby = true;
boolean haveHomework = true;
if(isNearby && !haveHomework) {
  System.out.println(“Yes!”);
} else {
  System.out.println(“No.”);
}
```

What values of these variables cause the program to print “yes?”

Nested if statements

```
// Nested if with dangling else
if (boolean expression)
    if (boolean expression)
        Do statement;
    else  // belongs to closest if
        Do other statement;
```

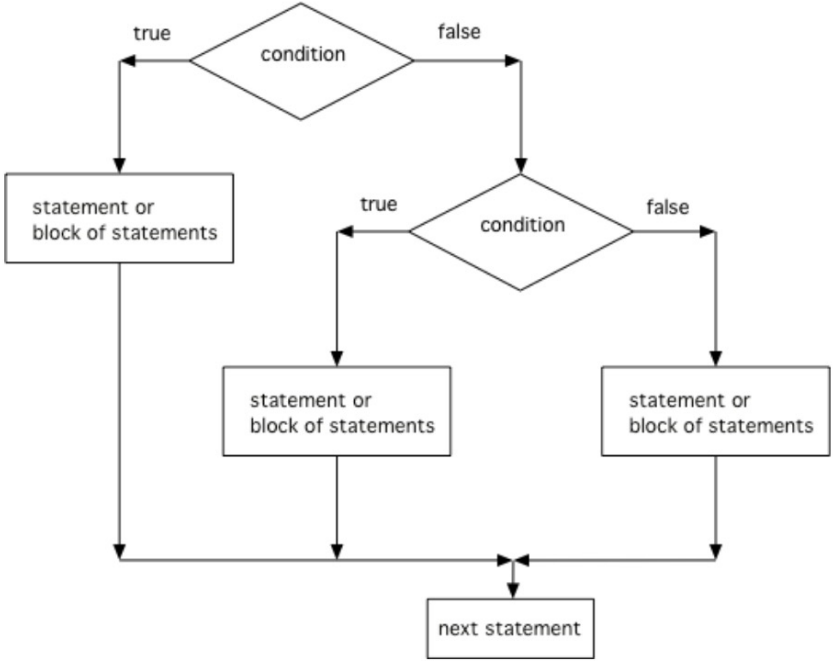
```
// Nested if with dangling else
if (boolean expression)
{
    if (boolean expression)
        do this;
}
else  // belongs to first if
  do that statement;
```

Can you go to see your friend at the park?

```
boolean isNearby = true;
boolean haveHomework = true;
if(isNearby) {
  if (!haveHomework) {
     System.out.println(“Yes!”);
  } else {
     System.out.println(“No, do HW.”);
  }
} else {
  System.out.println(“No, too far.”);
}
```

`If-else-if` statement

- A conditional with three or more options



The order that statements execute in a conditional with 3 options: if, else if, and else

Structure of `If-else-if` statement

```
// 3 way choice with else if
if (boolean expression)
{
   statement1;
}
else if (boolean expression)
{
   statement2;
}
else
{
   statement3;
}
```

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”);
}
```

Same as the previous nested version.

Comparing Strings

- The `equals` method is used to compare two strings letter by letter
- the operator `==` is used to compare if two variables ***refer to the same object***
- Two variables that refer to the same object are called **aliases** for the same object
- Always use `equals()` to compare strings

```
String a = new String("hi");
String b = new String("bye");
String c = b;       // c is now an alias for b
```

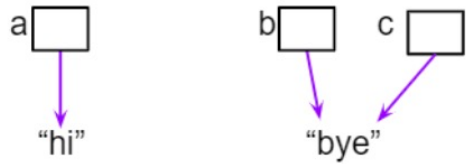


Figure 1: String aliases

Live Coding: Parking Sign