

Conditional Statements

15-110 Summer 2010

Margaret Reid-Miller

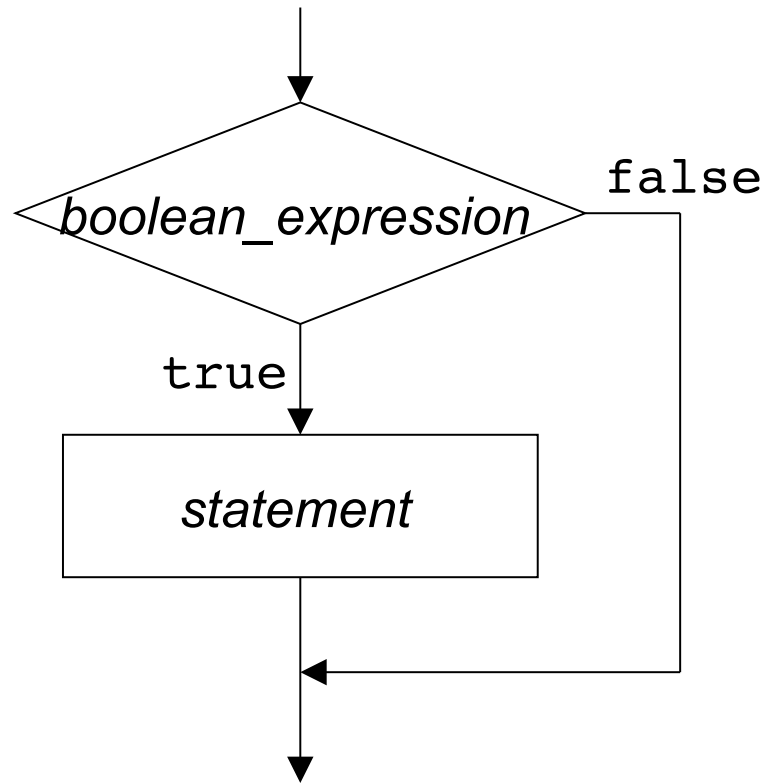
Conditional statements

- Within a method, we can alter the *flow of control* (the order in which statements are executed) using either conditionals or loops.
- The *conditional statements* `if`, `if-else`, and `switch` allow us to choose which statement will be executed next.
- Each choice or decision is based on the value of a boolean expression (also called the *condition*).

The `if` statement

- If we have code that we sometimes want to execute and sometimes we want to skip we can use the `if` statement.
- The form of the `if` statement is
`if (boolean_expression)`
`statement`
- If `boolean_expression` evaluates to `true`, then `statement` is executed.
- If `boolean_expression` evaluates to `false`, then `statement` is skipped.
- Note that the `boolean_expression` enclosed in parentheses must evaluate to `true` or `false`.

The `if` Flowchart



if-Statement Examples

```
if (count > 0)
    average = total / count;
```

```
if (age >= 26)
    if (hasLicense == true)
        System.out.println("You may rent a car.");
```

Or simply
hasLicense

```
daysInFeb = 28;
if (isLeapYear) {
    daysInFeb = 29;
    System.out.println(year + " is a leap year.");
}
```

The `if` Statement

- The *statement* in the `if` statement can be any Java statement:
 - A *simple* statement
 - A *compound* statement, such as an `if` statement
 - A *block* statement, a group of statements enclosed in braces `{ }`

```
if (zipcode == 15213) {  
    city = "Pittsburgh";  
    state = "PA";  
}
```

Proper indentation
becomes essential!

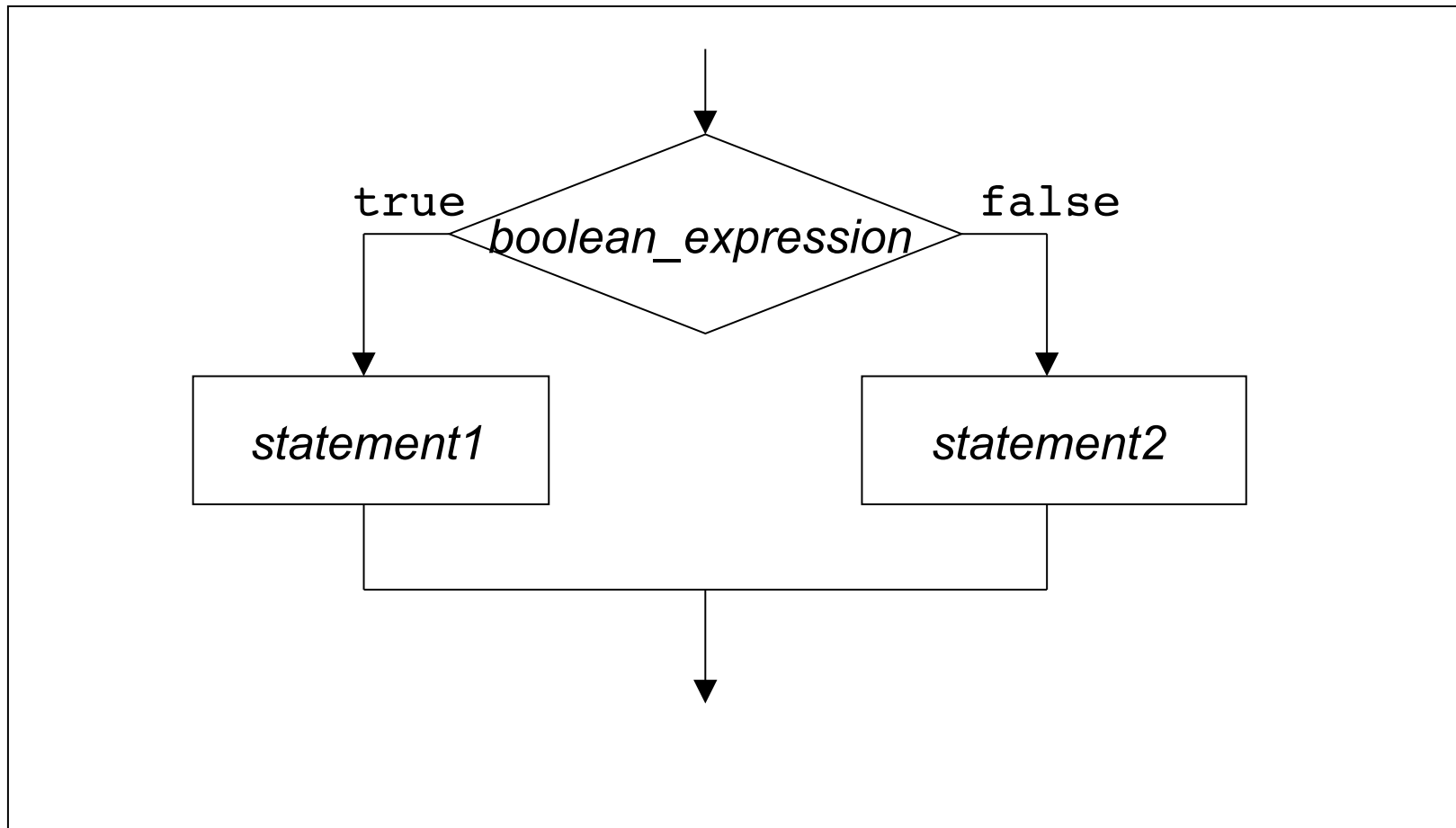
The `if-else` Statement

- If we want to choose between two alternative we use the `if/else` statement:

```
if (boolean_expression)
    statement1
else
    statement2
```

- If `boolean_expression` evaluates to `true`, then `statement1` is executed.
- If `boolean_expression` evaluates to `false`, then `statement2` is executed.

The `if-else` Flowchart



if-else Statement Examples

```
if (temperature <= 32.0) {  
    forecast = "SNOW"; ← The then clause  
}  
else {  
    forecast = "RAIN"; ← The else clause  
}  
  
if (count > 0) {  
    average = total / count;  
}  
else {  
    System.out.println("No data to average.");  
}
```

Common Error 1

- When you want to test if the value of a variable is in a range.

```
if (0 < temperature < 100) {  
    state = "LIQUID";  
}
```

WRONG!!

```
if (0 < temperature && temperature < 100) {  
    state = "LIQUID";  
}
```

Correct

Common Error 2

- When you want to test if the value of a variable is one of two alternates.

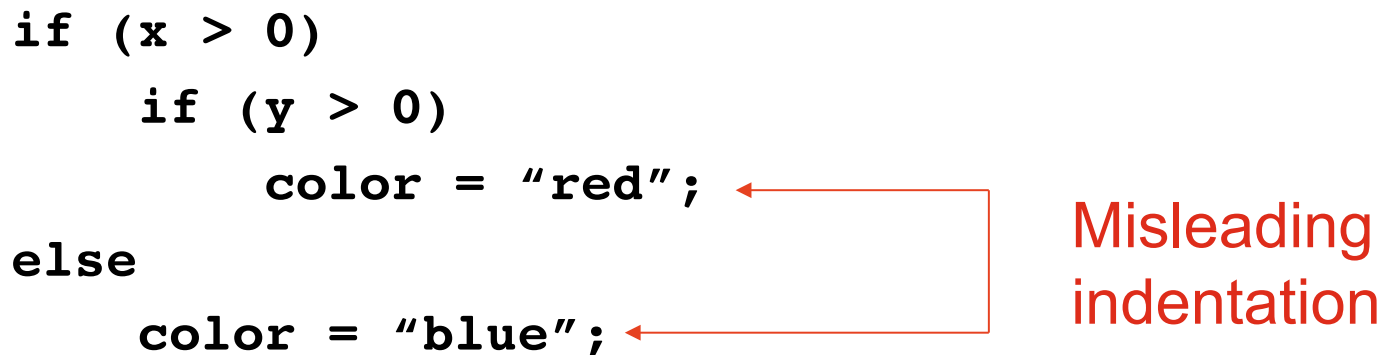
```
if (choice == 'M' || 'L') { WRONG!!  
    System.out.println("You're correct!");  
}
```

```
if (choice == 'M' || choice == 'L') {  
    System.out.println("You're correct!");  
} Correct
```

The *Dangling else* Problem

- When an `if` statement is nested inside the `then` clause of another `if` statement, the `else` clause is paired with the closest `if` statement without an `else` clause.

```
if (x > 0)
    if (y > 0)
        color = "red";
else
    color = "blue";
```

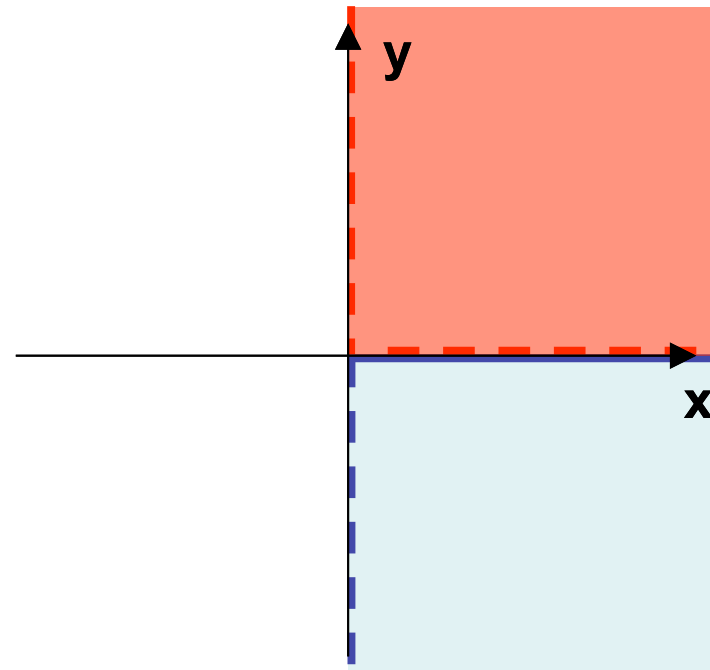


Misleading indentation

The *Dangling else* Problem

- In reality it is

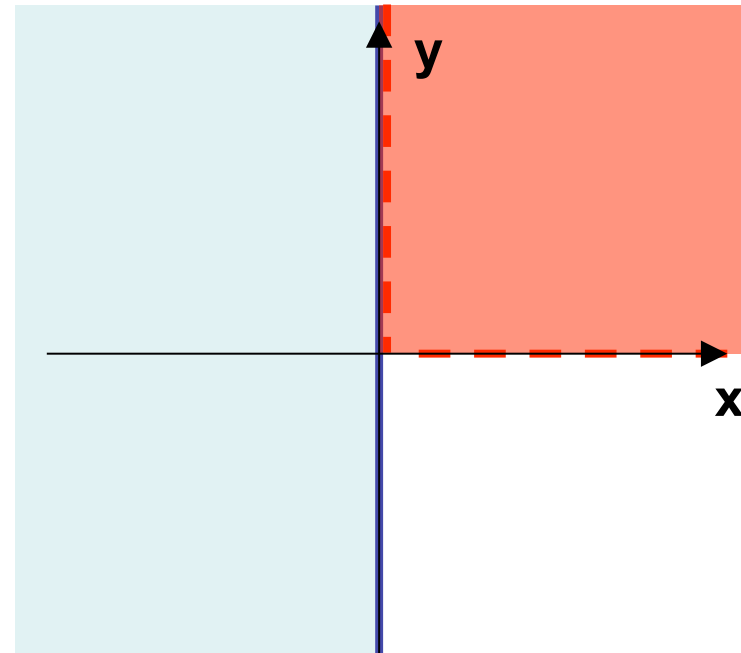
```
if (x > 0)
  if (y > 0)
    color = "red";
else
  color = "blue";
```



The *Dangling else* Problem

- Use braces to pair else with the outer if

```
if (x > 0) {  
    if (y > 0)  
        color = "red";  
}  
else {  
    color = "blue";  
}
```



- Compare flowcharts!

Multiple Alternatives

- Determine if a number is positive, negative, or zero:

```
if (value < 0) {  
    System.out.println("Value is negative.");  
}  
if (value == 0) {  
    System.out.println("Value is zero.");  
}  
if (value > 0) {  
    System.out.println("Value is positive.");  
}
```

*Computer thinks **any combination** of the three statements can be executed.*

Multiple Alternatives

- Determine if a number is positive, negative, or zero

```
if (value < 0) {
    System.out.println("Value is negative.");
}
else {
    if (value == 0) {
        System.out.println("Value is zero.");
    }
    else {
        if (value > 0) {
            System.out.println("Value is positive.");
        }
    }
}
```

*At most one statement is executed.
Leads to lots of indentation.*

Multiple Alternatives

- Determine if a number is positive, negative, or zero

```
if (value < 0) {  
    System.out.println("Value is negative.");  
}  
else {  
    if (value == 0) {  
        System.out.println("Value is zero.");  
    }  
    else {  
        if (value > 0) {  
            System.out.println("Value is positive.");  
        }  
    }  
}
```

*Remove unnecessary
brackets and re-indent*

Multiple Alternatives

- Determine if a number is positive, negative, or zero:

```
if (value < 0) {  
    System.out.println("Value is negative.");  
}  
else if (value == 0) {  
    System.out.println("Value is zero.");  
}  
else if (value > 0) {  
    System.out.println("Value is positive.");  
}
```

*At most one statement is executed.
Each choice, however, is at same indentation.*

Multiple Alternatives

- Determine if a number is positive, negative, or zero:

```
if (value < 0) {  
    System.out.println("Value is negative.");  
}  
else if (value == 0) {  
    System.out.println("Value is zero.");  
}  
else { // value must be positive  
    System.out.println("Value is positive.");  
}
```

*It is clear, **exactly one** statement is executed.*

Multiple Alternatives: Assignments

- Determine the fare: \$2 for a child (no more than 11 years), \$3 for a senior (at least 65 years), or \$5 for an adult.

```
int fare;  fare must be defined  
if (age _____) { before the if statement  
    fare = 2;  
}  
else if (age _____) { // _____  
    fare = 5;  
}  
else { // _____  last clause must be  
    fare = 3; else with no if  
}  
System.out.println("Your fare is $" + fare);
```

Exercise

- Write a method that prints how many of n1, n2, and n3 are odd:

```
public void printNumOdd(int n1, int n2, int n3) {
```

```
}
```

Exercise

- Write a method that print whether die1 and die2 are doubles, cat's eyes (two 1's) or neither of these.

```
public void printDoubles(int die1, int die2) {
```

Programming Style

- Single-line `if` statement:

```
if (y > 0) color = "red";
```

- Multi-line `if` statement:

```
if (zipcode == 15213) {  
    city = "Pittsburgh";  
    state = "PA";  
}
```

- The `if-else` statement:

```
if (temperature <= 32.0) {  
    forecast = "SNOW";  
}  
else {  
    forecast = "RAIN";  
}
```

- Multiple alternatives:

```
if (value < 0) {  
    valueType = "negative";  
}  
else if (value == 0) {  
    valueType = "zero";  
}  
else { // no if here!!  
    valueType = "positive";  
}
```

Testing For Equality

- For **primitive values** use `==` for equality testing.
- For **objects**, use the `equals` method for testing equal contents.
 - The argument must be the same type as the object on which `equals()` is called. The method returns true or false depending on whether both objects are “equal” or not.
- For example, let `day` be an `int` variable and `month` be a `String` variable.

```
if (day == 1 && month.equals("APRIL")) {  
    System.out.println("It's April Fool's Day");  
}
```

*Two String objects are equal if they have **exactly** the same characters, including case and number of characters.*

Testing for Equality with doubles

- Which statement will Java print?

```
double x = Math.sqrt(2.0);
double y = x * x;

if (y == 2.0) {
    System.out.println("sqrt(2) * sqrt(2) is 2");
}
else {
    System.out.println("sqrt(2) * sqrt(2) "
        + "is not 2. It is " + y);
}
```

Never test for exact equality
with floating point numbers!

Testing for Equality with doubles

- Because of round-off errors, you should test if the numbers are close.

```
double tolerance = 1.0e-10;
double x = Math.sqrt(2.0);
double y = x * x;

if (Math.abs(y - 2.0) < tolerance) {
    System.out.println("sqrt(2) * sqrt(2) is 2");
}
else {
    System.out.println("sqrt(2) * sqrt(2) "
        + "is not 2. It is " + y);
}
```

Short-Circuit Evaluation

- *Short circuit evaluation* (or *lazy evaluation*) : If the first conditional in an `&&` expression is `false`, Java does not execute the second conditional.

Example:

```
if (liters > 0 && total/liters > threshold) {  
    System.out.println("WARNING: Exceeds threshold");  
}
```

What if the expression was an `||` expression?

The switch statement

- If an `if/else` statement with multiple alternatives compares an `int` or `char` variable or expression against several constants you can use a `switch` statement.

Example:

```
switch (suitAsChar) {  
    case 'C': suitAsName = "Clubs"; break;  
    case 'D': suitAsName = "Diamonds"; break;  
    case 'H': suitAsName = "Hearts"; break;  
    case 'S': suitAsName = "Spades"; break;  
    default: suitAsName = "Unknown";  
}
```