Java™ Methods

Lecture slides from Larry Baker
IB Computer Science

An Introduction
to Object-Oriented Programming

if (chapter == 7)

Boolean Expressions and if-else Statements

7.1 - Prologue

- Normal control flows sequentially from one statement to the next in a program
- This sequence can be altered
  - How?
    - Calling a method
    - Iterative statements (loops)
    - Conditional statements (if-else)
    - Switch statements
    - Exceptions

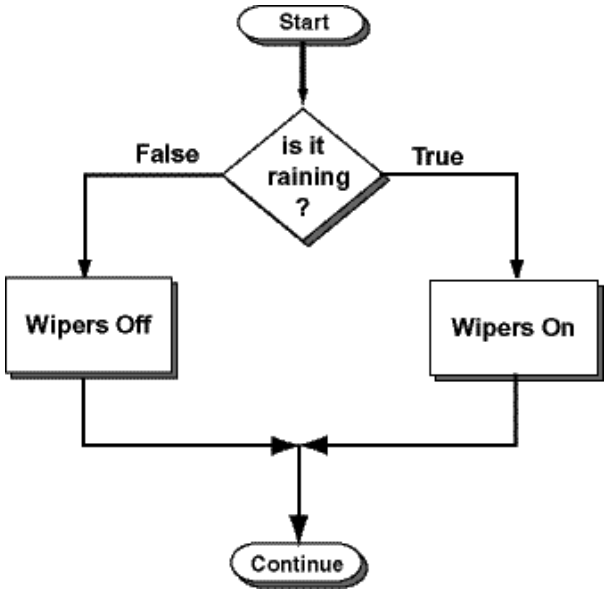
Conditional branching

- Program instructions go in a step-by-step order.
  - But many times programs need to execute different instructions based on the conditions of when the program is run.

7.2 – *if-else* Statements

Decision Making

- You are driving in your car and it starts to rain.
- The rain falls on your windshield and makes it hard to see.
- Should your windshield wipers be on or off?



Example

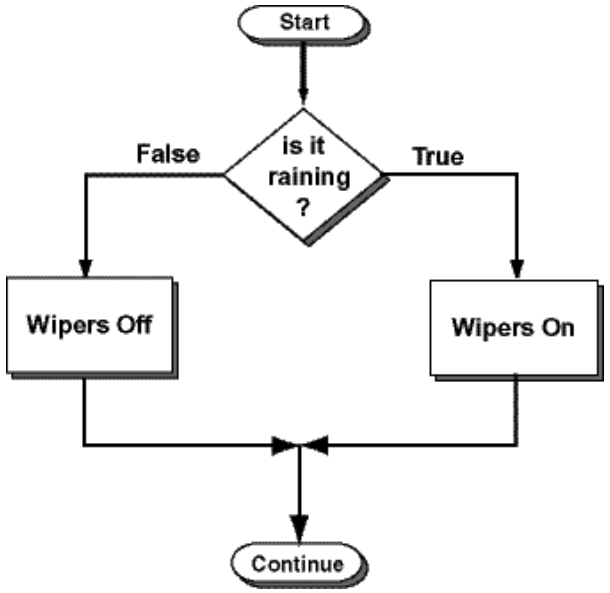
```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
        System.out.println("Application accepted.");
}
```

Example

```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
        System.out.println("Application accepted.");
} else {
        System.out.println(“Application denied.”);
}
```

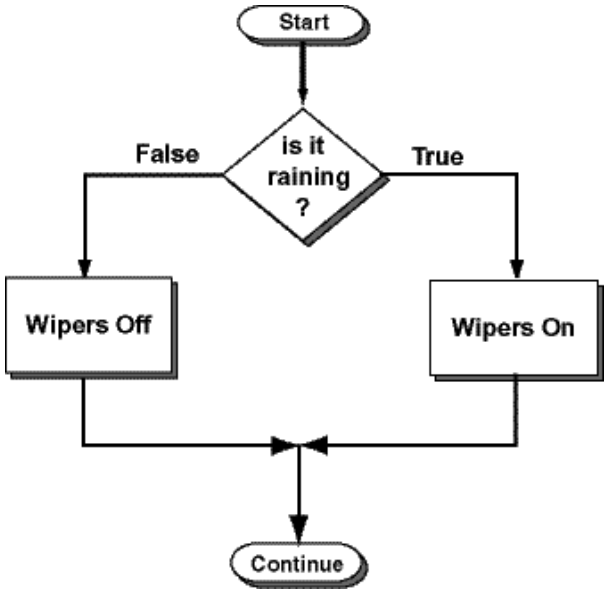
Decisions

- The words if and else are markers that divide the decision into two sections.
  - The else divides the "true branch" from the "false branch".
- The if statment always asks a question (often about a variable.)
- If the answer is "true" only the true-branch is exectued.



Decisions

- If the answer is "false" only the false-branch is executed.
- No matter which branch is chosen, execution continues with the statement after the false-branch.
- Notice that a two-way decision is like picking which of two roads to take to the same destination.



if – else statement

The form of the if-else is very straight forward:

```
if (condition)
{
    statements executed if condition is true;
} else {
   statements executed if condition is false;
}
```

The else statement is not required but optional.

IF – ELSE statement

Do these statements before.

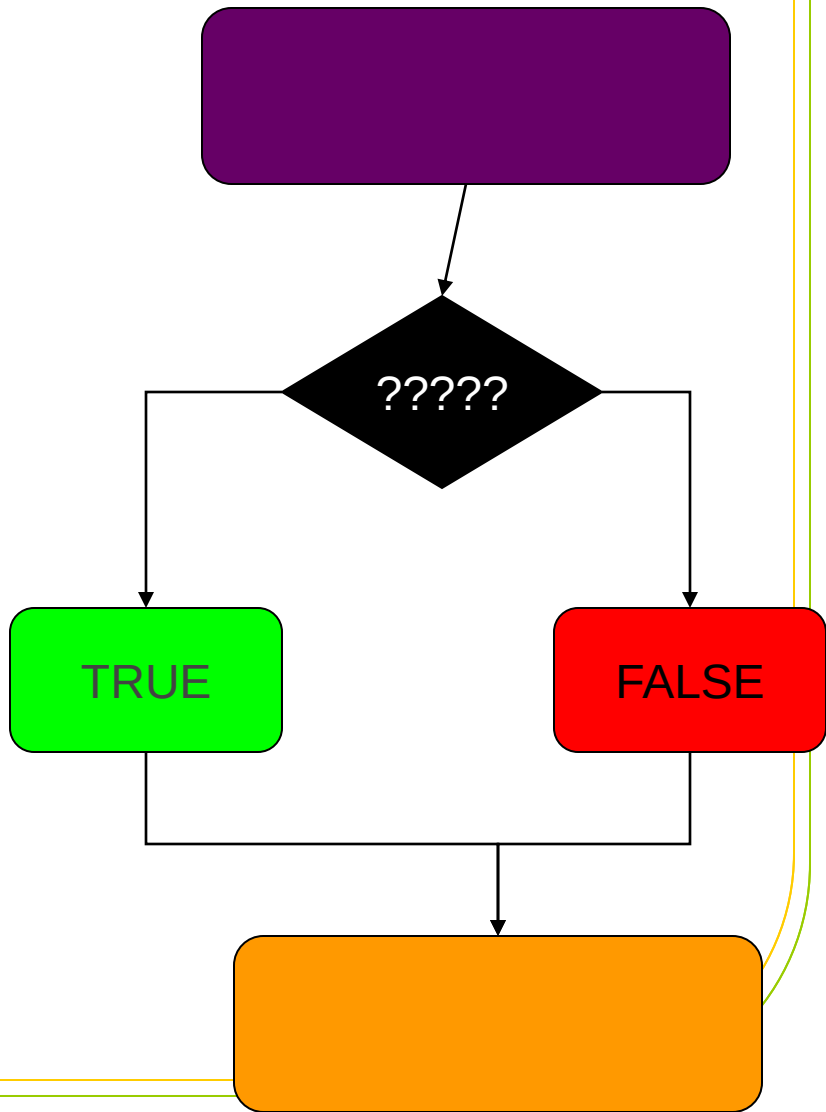
if (*condition*)

{ *Do this clause if the condition is true* }

else

{ *Do this clause if the condition is false* }

Do these statements after.



7.2 – *if-else* Statements

- **Should you always use braces?**
  - When the `if` or `else` clause contains only one statement, the braces are optional, but it is safer to always keep them.
  - This is a safer practice because any later addition of a statement to one of the clauses will require braces.
  - If you don't have the braces with multiple statements, the compiler may not give any error message, but your code will not do what was expected.

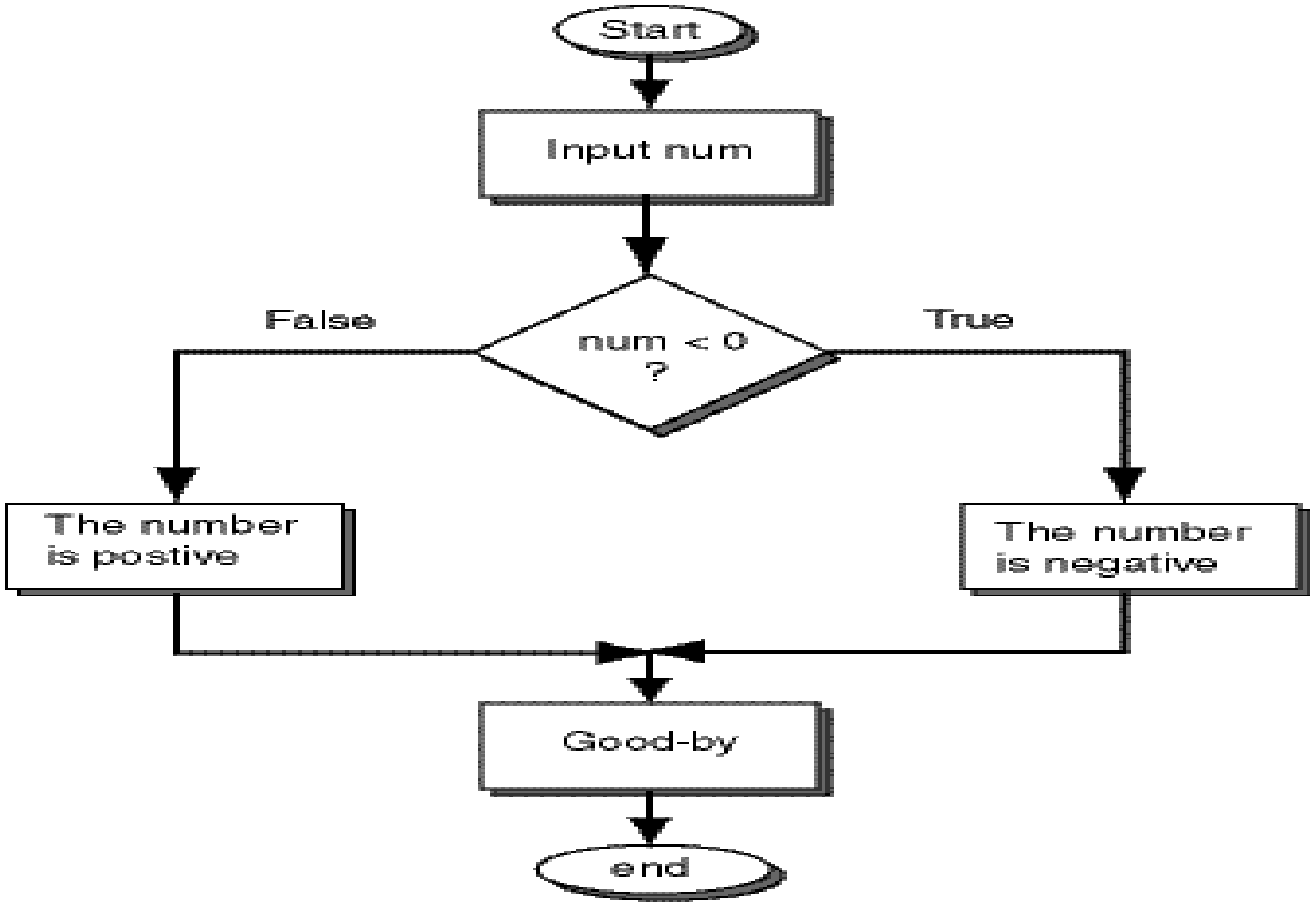
```
Do these statements before.
if (condition)
    { Do this clause if the condition is true
    }
else
    { Do this clause if the condition is false
    }
Do these statements after.
```

7.2 – *if-else* Statements

- The else clause is optional
  - If condition is true
    - Execute the true part of the statement
  - If condition is false
    - Program skips the true part of the statement and moves on

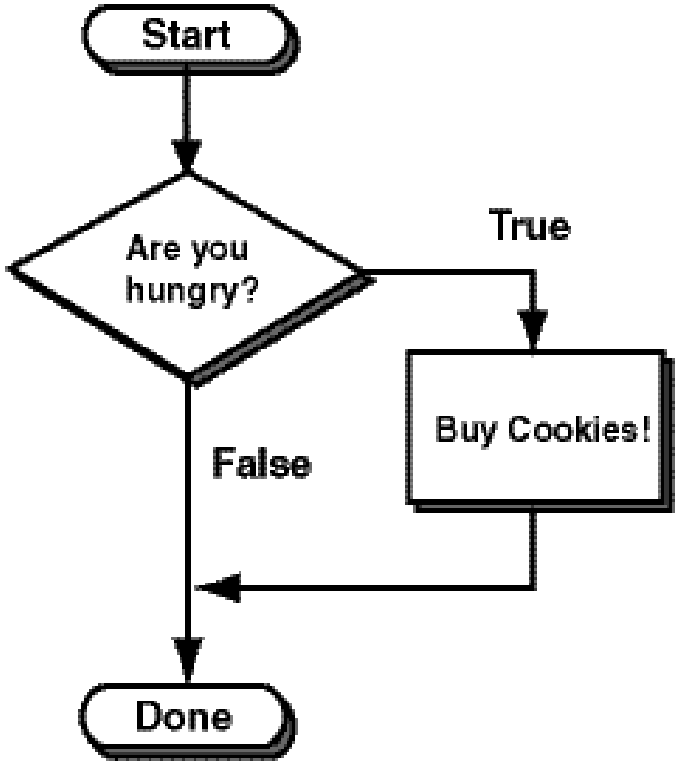
```
Do these statements before.
if (condition)
{
    Do this clause if the
    condition is true
}
Do these statements after.
```

- Example of IF THEN program



Single-block if Statements

- The decision is about whether to buy cookies or just stroll on by.
  - In the picture of a decision, you start at the top, then follow the line to the question:
  - Are you hungry? The answer to the question is either *true* or *false*.
  - If the answer is *true*, follow the line labeled *true*, do the directions in the box "buy cookies", follow the line to "done"
  - If the answer is *false* follow the line to "done"



if-else Statement

```
if ( <condition> )
{
    < statements >
}
else
{
    < other statements >
}
```

TRUE

FALSE

If Statement

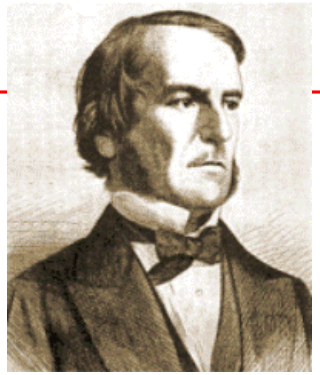
```
if ( <condition> )
{
    < statements >
}
```

TRUE

FALSE

else clause is optional

boolean Data Type



- George Boole (1815 - 1864)
- boolean variables may have only two values, true or false.
- You define boolean fields or boolean local variables the same way as other variables.

```
private boolean hasMiddleName;
boolean isRolling = false;
```

boolean
true
false

Reserved words

Boolean Expressions

- In if (<*condition*>)
  - <*condition*> is a *Boolean* expression.
- A Boolean expression evaluates to either true or false.

Boolean Expressions

- You can assign the result of a Boolean expression to a `boolean` variable. For example,

```
Limit = 10;
boolean exceedsLimit = x > limit;
```

Relational Operators

- The result of every comparison is *boolean* (true or false).
- *operator* *meaning*

<	less than
<=	less than or equal to
==	equal to
>=	greater than or equal to
>	greater than
!=	not equal

Relational Operators

- **Confusion between equals (=) and equals (==)**
  - **=**
    - **Is for assignment**
    - **This is not equals; this is assignment**
      - **X = X + Y;**
        - **"Take what is on the right of the assignment operator and assign it to the variable on the left of the assignment operator"**
  - **==**
    - **Is for comparison**
- The result of a relational operator expression will always be either true or false

Common Errors

- **0 < x < 100**
  - Comparison operators can be used with *two* numbers.
    - Although you can write 0 < x < 100 in mathematics, it is illegal in Java.
    - You must write this as the *and* of two comparisons:
      - 0<x && x<100
- **= instead of ==**
  - Using the assignment operator instead of equality will produce a compiler error, which is easy to fix.
- **== with floating-point**
  - Because floating-point numbers are not exact, you should always use >= or <= instead of ==.
    - For example, because the decimal number 0.1 can not be represented exactly in binary,
      - (0.1 + 0.1 + 0.1) is *not* equal to 0.3!

Relational Operators (cont'd)

- Apply to numbers or chars:

```
if  ( x <= y ) ...
if  ( a != 0 ) ...
if  ( letter == 'Y' ) ...
```

- Do not use == or != with doubles
  - Why?
    - Because they may have rounding errors

Relational Operators (cont'd)

- Be careful using == and != with objects (e.g., Strings): they compare references (addresses) rather than values (the contents)

```
String cmd = console.readLine();
if ( cmd == "Help" ) ...
```

Wrong!
(always false)

Relational Operators (cont'd)

- Use the **equals** method to compare Strings:

```
String cmd = console.readLine();
if ( cmd.equals ("Help")  )   ...
```

or

```
if (  "Help".equals (cmd)  )   ...
```

7.5 – Logical Operators

- 3 logical operators
  - "Truth tables" in math classes

What are the logical operators?

Logical Operators (cont'd)

- AND → &&
  - (*condition1* **&&** *condition2*) is true if and only if both *condition1* and *condition2* are true
- OR → ||
  - (*condition1* **||** *condition2*) is true if and only if *condition1* or *condition2* (or both) are true
- **NOT → !**
  - **!***condition1* is true if and only if *condition1* is false

OR Operator (||)

- Here is how || works:
  - true || true = true
  - false || true = true
  - true || false = true
  - false || false = false

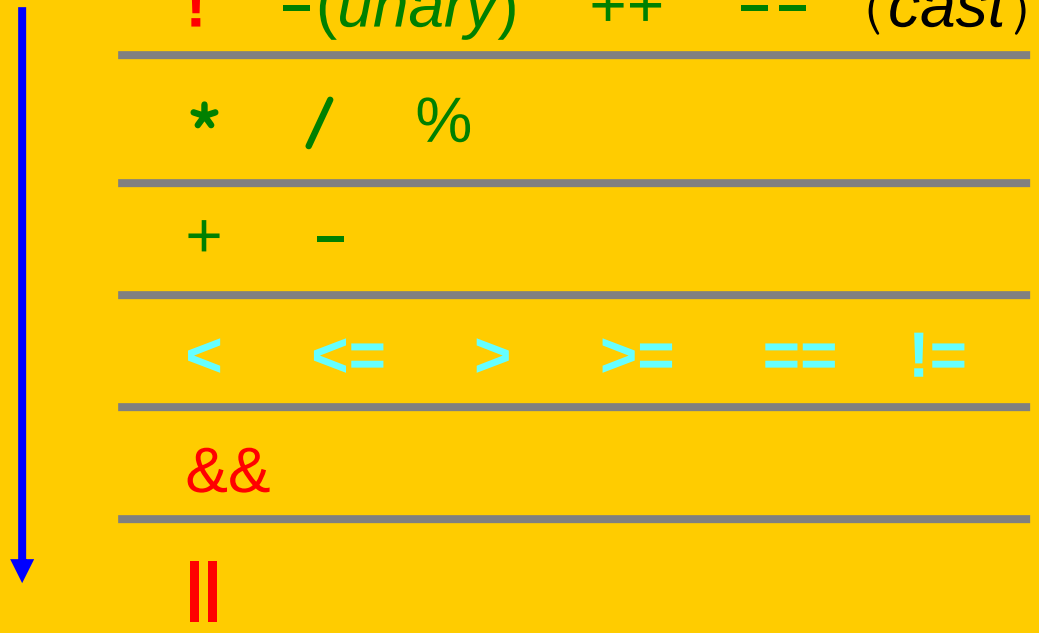
AND Operator (&&)

- Here is how && works:
  - true && true = true
  - false && true = false
  - true && false = false
  - false && false = false

NOT Operator (!)

- **!** (exclaimation point.)
  - The NOT operator
    - Changes *true* to *false*
    - Changes *false* to *true*

Ranks of Operators (Order of operations)



1. MATHEMATICAL
2. RELATIONAL
3. LOGICAL

7.7 – Short–Circuit Evaluation

LOGICAL OPERATOR AND
&&
FALSE &&
if (*condition1* **&&** *condition2*) ...
If *condition1* is false, *condition2* is not evaluated (the result is false anyway)

LOGICAL OPERATOR OR
||
OR
TRUE ||
if (*condition1* **||** *condition2*) ...
If *condition1* is true, *condition2* is not evaluated (the result is true anyway)

7.8 – if-else-if and Nested if-else

The general syntax of a nested if statement is:

```
if (condition1)
{
    block1
}
else if (condition2)
{
    block2
} else {
    block3
}
```

Example

```
if (x > 0) {
        System.out.println("Positive");
} else if (x < 0) {
        System.out.println("Negative");
} else {
        System.out.println("Zero");
}
```

Common if-else Errors

What's wrong?

```
if (...) ;
{
   statements;
   ...
}
```

Extra semicolon

It is safer to always use braces in if-else

What's wrong?

```
if (...)
   statement1;
   statement2;
...
```

What's wrong?

```
if (...)
   if (...)

      statement1;
else
   statement2;
...
```

Missing braces

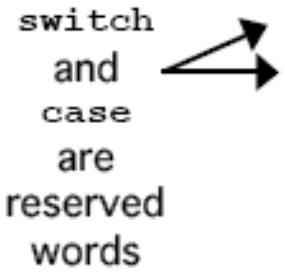
Chapter 7 : Boolean Expressions and *if-else* statements

- **7.10 : SWITCH statement**

SWITCH statement

The switch Statement

- The general syntax of a **switch** statement is:




```
switch ( expression )
{
    case value1 :
       statement-list1
    case value2 :
       statement-list2
    case value3 :
       statement-list3
    case  ...
}
```

If *expression* matches *value2*, control jumps to here

Exercise

- Determine if a person is short, medium, or tall.

```
1  public class MyClass {
2    public static void main(String[] args) {
3      double height = 51; // enter height in inches here
4
5      // code here
6    }
7  }
```

Short is less than 63 inches.

Medium is more than 63 and less than 71.

Tall is greater than 71 inches.

Exercise

- Determine if a number is divisible by 2, 3, and/or 5.

```
1  public class MyClass {
2    public static void main(String[] args) {
3      int number = 51; // enter the number here
4
5      // code here
6    }
7  }
```

If divisible by 2, print “Divisible by 2”

If divisible by 3, print “Divisible by 3”

If divisible by 5, print “Divisible by 5”

Review:

- What are the possible values of a boolean variable?
- What operators are used to compare values of numbers?
- How can you test whether two Strings have the same values?
- Which binary operators have higher rank (are executed first), relational or logical?

Review (cont'd):

- Can you have an if statement without an else?
- How long can an if-else-if sequence be?