More on `IF`

COMP163 Introduction to Computer Programming

"In any moment of decision, the best thing you can do is the right thing, the next best thing is the wrong thing, and the worst thing you can do is nothing."

Theodore Roosevelt

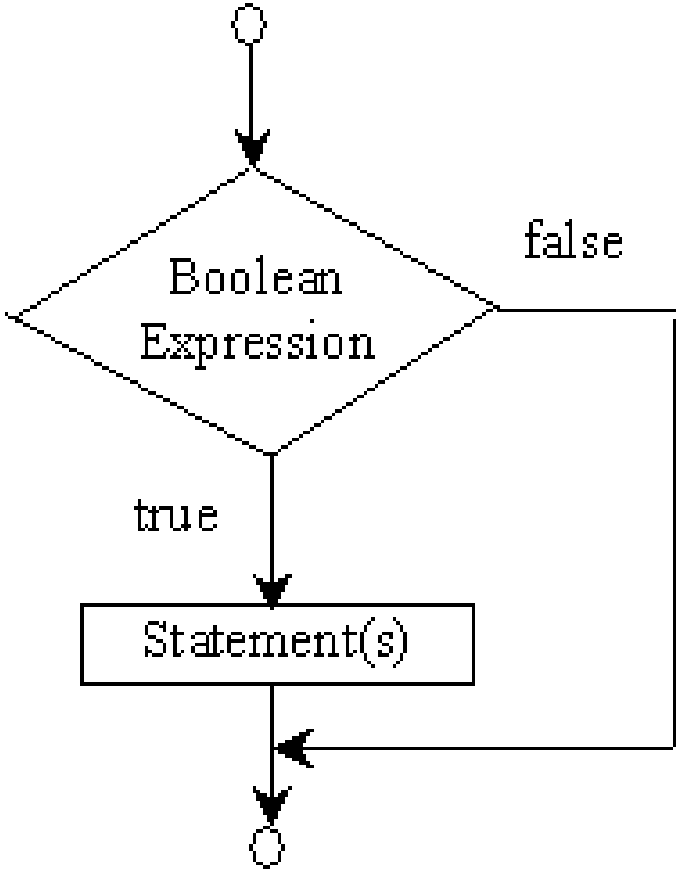
ZyBooks Assignment

- Read chapter 6 of the ZyBooks text
- Answer all of the participation questions except sections 6.11 and 6.17
- Due by midnight on Thursday, September 27

Programming Assignment

- The weekly programming assignment has been posted on Blackboard
- Either part 1 or part 3 must be implemented with a GUI
- Due by midnight on SUNDAY, September 31

if Logic



if Syntax

```
if ( true or false decision )
    next statement;  or  {next block}
```

- The program will execute the next statement if and only if the decision is true
- The next statement can be a single Java statement or a block
- Whitespace is optional

Comparison Operators

Operator	*Name*
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Watch Out for Double Equals

```
if (no = good)       /* Incorrect */

if (does == work)  /* Correct */
```

Indenting

- Although the Java compiler does not care, it is traditional to indent the statements that are executed only when the if is true
- Indenting is required for all assignments

```
if ( dog > cat ) {
    cow  = 5;
    bull = 37;
}
```

Blocks

- In Java, a block is a bunch of code surrounded by **{** curly brackets **}**
- Almost anyplace you might put a single statement, you can put a block of statements

Brackets Recommended

- Your instructor recommends you always use {brackets} around statements following an **`if`**
- This makes it clear what will be skipped
- Without brackets, a careless update later can make your program not work

```
if (dog == cat )
        cat = 47;

dog += cat;
```

```
if (dog == cat )
        update = 1;
        cat = 47;
dog += cat;
```

Brackets Recommended

- Your instructor recommends you always use {brackets} around statements following an **`if`**
- This makes it clear what will be skipped
- Without brackets, a careless update later can make your program not work

```
if (dog == cat ) {
        cat = 47;
}

dog += cat;
```

```
if (dog == cat ) {
        update = 1;
        cat = 47;
}
dog += cat;
```

Style

- It is important that your programs are easily readable by humans
- Academic research indicates that good style is important
- A clean, clear program looks good

- We are going to require good style

If and Brackets

- Always use {Curly Brackets} with an IF
- Put the left { curly bracket to the right of the right parenthesis on the same line

```
if (dog > cat) {
    // do something;
}
```

Compound Logical Statements

- You can combine relational expressions with logical or Boolean operators
- Expressions can be combined with AND, OR, XOR and NOT

Boolean Operators

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

George Boole
19th century British mathematician
inventor of Boolean logic

Boolean Logic

- A logical AND is true if both sides are true

```
if (cow > bull && bull == 17)
```

- A logical OR is true if at least one side is true

```
if (cow > bull || bull == 17)
```

What is displayed?

```
int bull = 3, cow = 5, cat = 7;
int dog = 9;
if((bull != cow) && (cat < cow)){
    dog = 10;
}
System.out.println(dog);
```

A. 3

B. 5

C. 7

D. 9

E. 10

Now what is displayed?

```
int bull = 3, cow = 5, cat = 7;
int dog = 9;
if((bull != cow) || (cat < cow)){
    dog = 10;
}
System.out.println(dog);
```

A. 3

B. 5

C. 7

D. 9

E. 10

Caution

- Adding a semicolon at the end of an `if` clause is a common mistake

```
if (radius >= 0) ;
{
        area = radius*radius*Math.PI;
}
```

Wrong

- **Students spend more time fixing this programming mistake than any other**
- This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error
- This error often occurs when you use the next-line indenting style

Try It

- Write an `if` statement that sets mongoose to 5 if weasel is bigger than mink or if mink is negative

```
int mongoose = 0, weasel, mink;
// assume values are given to weasel and mink
```

Possible Solution

- Write an `if` statement that sets mongoose to 5 if weasel is bigger than mink or if mink is negative

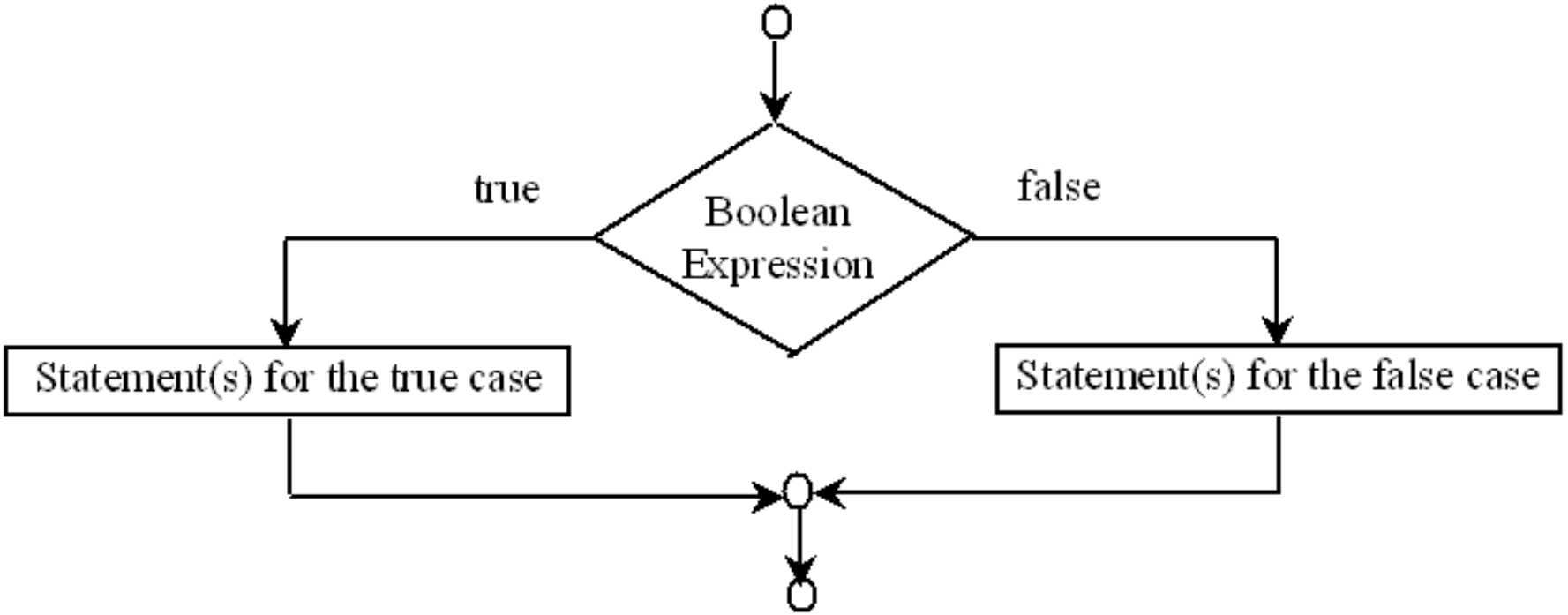
```
int mongoose = 0, weasel, mink;
// assume values are given to weasel and mink
if ( weasel > mink || mink < 0) {
        mongoose = 5;
}
```

else

- An if statement can have an else clause that is executed only when the if condition is false

```
if ( cat > 4 ) {
     System.out.println("cat is big");
} else {
     System.out.println("cat is small");
}
```

if else Logic

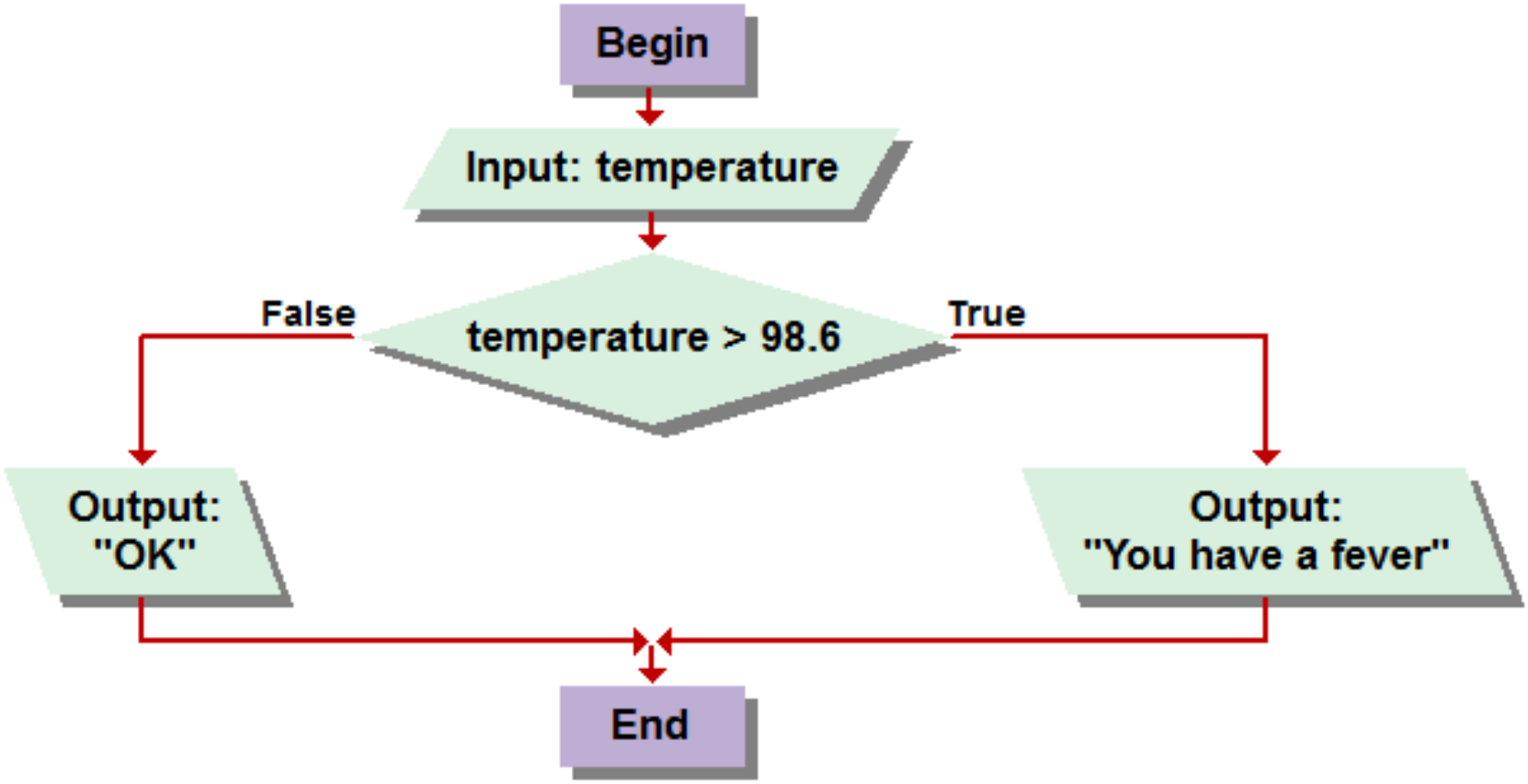


One or the Other

- With an if – else statement, either the if part or the else part are executed, but never both

```
if ( logical expression ) {
        Executed only if true
} else {
        Executed only if false
}
```

Flowchart to tell if a fever

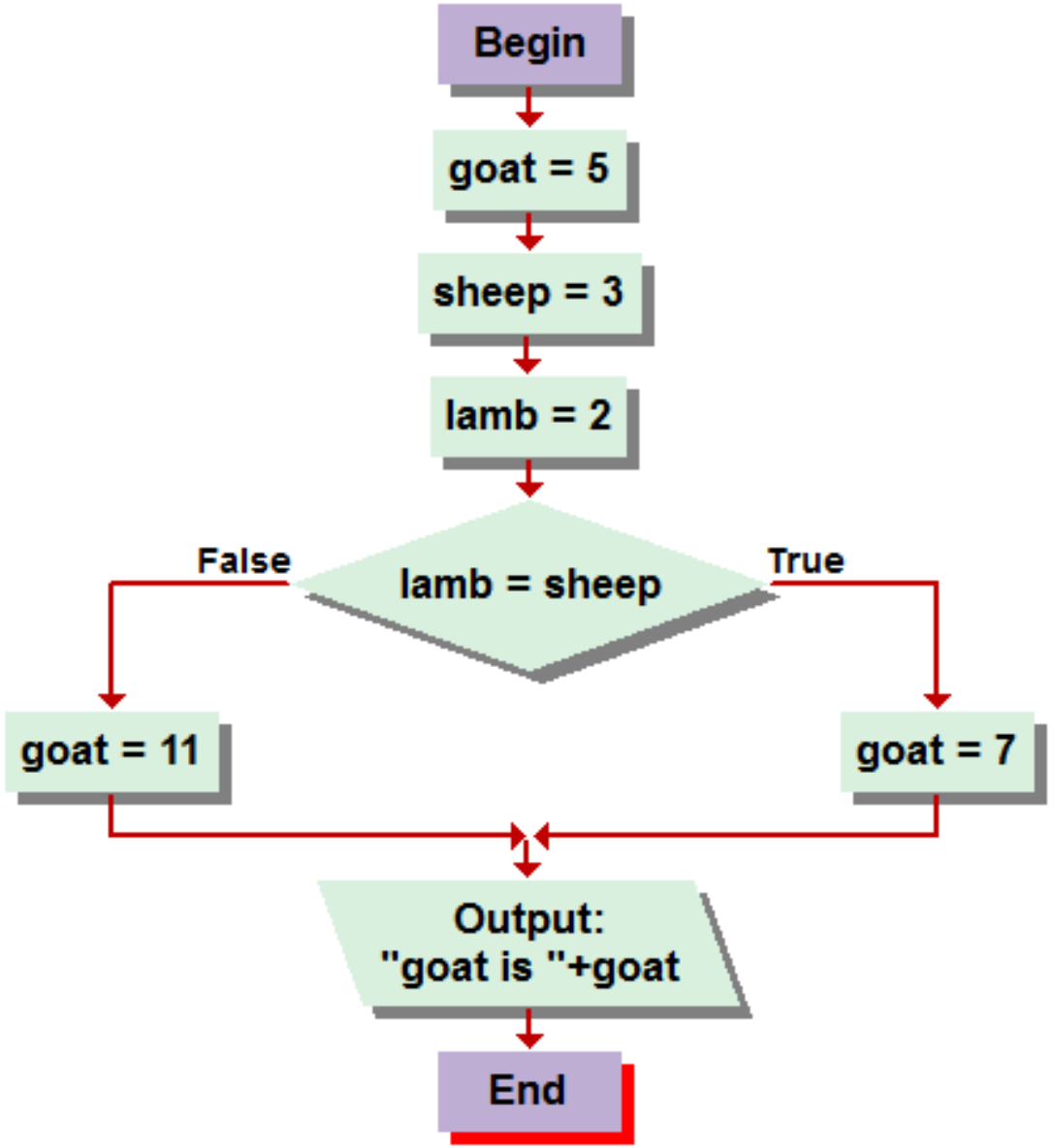


Java Version

```
public class Fever {
    public static void main(String[] unused) {
        double temperature;
        java.util.Scanner keyboard = new
                    java.util.Scanner(System.in);
        System.out.println("Enter temperature");
        temperature = keyboard.nextDouble();

        if (temperature > 98.6) {
            System.out.println("You have a fever");
        } else {
            System.out.println("OK");
        }
    }
}
```

What is output by this program?



A. goat is 2
B. goat is 3
C. goat is 5
D. goat is 7
E. goat is 11

A Recommendation

- Always put the clause following an **`if`** and **`else`** in curly brackets
- This avoids nested if errors
- This avoids errors when inserting an extra line in an if clause
- The brackets should line up with proper indenting

What is displayed?

```
int dog=6, cat=3, cow=5;
if (dog > cat) cow=7; else cow = 4; cow=9;
System.out.println( cow );
```

A. 4

B. 5

C. 7

D. 9

E. Compiler Error

Properly Formatted

```
int dog=6, cat=3, cow=5;
if (dog > cat)
        cow=7;
else
        cow = 4;
cow=9;
System.out.println( cow );
```

Even Better

```
int dog=6, cat=3, cow=5;
if (dog > cat) {
        cow=7;
} else {
        cow = 4;
}
cow=9;
System.out.println( cow );
```

Comparing Strings

- The comparison operators (such as **>**, **<**, **==**) do **not** work with Strings. They only work on primitive data types (int, float, double and char)
- You need to use the `equals` or `compareTo` methods to compare Strings

equals Method

- **`equals`** is a boolean method of String that is true if the parameter has exactly the same characters

String cat = "frog", dog = "toad", cow = "frog";

if (cat.equals(dog)) is false

if (cat.equals(cow)) is true

Comparison Examples

```
String dog = "dog", bull = "bull", bulldog="bulldog", cat;

cat = bull + dog;                    // cat = "bulldog"
if (cat == bulldog)   not true

if (cat.equals(bulldog))   true

cat = bulldog;
if (cat == bulldog)   true
```

compareTo method

```
String dog, cat;
dog.compareTo( cat )
```

- if dog is alphabetically before cat, then the compareTo method returns a positive number
- if dog is alphabetically after cat, then the compareTo method returns a negative number
- if dog is the same as cat, then the compareTo method returns zero

compareTo method

```
String dog, cat;
dog.compareTo( cat )
```

if dog < cat +

if dog == cat 0

if dog < cat -

compareTo example

```
String dog = "bull dog";
String cat = "puma";

if (dog.compareTo(cat) > 0) { // if positive
    System.out.println("dog comes before cat");
}
```

- The result of compareTo is zero if the Strings are equal, positive if dog < cat and negative if dog > cat

Upper and Lower Case

- When comparing Strings, upper and lower case characters are different

"Dog" is not equal to "dog"

- The **`equalsIgnoreCase`**(anotherString) method is true if both strings have the same letters regardless of case
- **`compareToIgnoreCase`**(anotherString) works like compareTo but ignores case

Complete the Program

```
public class Tax{
    public static void main(String[] unused) {
        double rate;
        String name;
        java.util.Scanner keyboard = new
                    java.util.Scanner(System.in);
        System.out.println("Enter name");
        name = keyboard.next ();

        /* Set rate to 0.01 if the name is Fred, otherwise set
           rate to 25.0  */
    }
}
```

Possible Solution

```
public class Tax{
    public static void main(String[] unused) {
        double rate;
        String name;
        java.util.Scanner keyboard = new
                    java.util.Scanner(System.in);
        System.out.println("Enter name");
        name = keyboard.next ();

        if( name.equalsIgnoreCase("Fred")) {
            rate = 0.01;
        } else {
            rate = 25.0;
        }
    }
}
```

boolean variables

- A boolean variable can only have the values **true** or **false**

```
boolean rabbit, err = false;
rabbit = true;
if (a > 5) {
    err = true;
}
```

Logical Equations

- A boolean variable can be set to the result of a logical expression

```
int x=3, y=5, z=7;
boolean bat;
bat = x > y;
bat  = (x != y) && (z > y);
```

- The expression is evaluated once and the boolean variable is then set to true or false
- Changing **x,y** or **z** will not change the value of **bat** after the above equations

boolean variables in IF

- You can use a boolean variable in an if statement without a comparison

```
boolean  problem = false;
     …
problem = true;
     …
if ( problem ) {
  System.out.println("look out");
}
```

boolean Methods

- A method can return a boolean (true or false) value

```
boolean close(double cat, double dog) {
        if ( Math.abs(cat – dog) < 0.01 ) {
                return true;
        }
        return false;
}
```

Using boolean Methods

- A boolean method can be used in an if

```
double cobra = 0.666,  mamba = 2.0 / 3.0;
if ( close( cobra, mamba ) ) {
        System.out.println("same");
}
```

What is displayed?

```
int dog=2, cat=3, cow=5;
boolean trout;
trout = dog < cat || cow != dog;
if (trout) {
   System.out.println(dog);
} else {
   System.out.println(cat);
}
```

A. 2

B. 3

C. 5

D. Compiler Error

ZyBooks Assignment

- Read chapter 6 of the ZyBooks text
- Answer all of the participation questions except sections 6.11 and 6.17
- Due by midnight on Thursday, September 27

Programming Assignment

- The weekly programming assignment has been posted on Blackboard
- Either part 1 or part 3 must be implemented with a GUI
- Due by midnight on SUNDAY, September 31