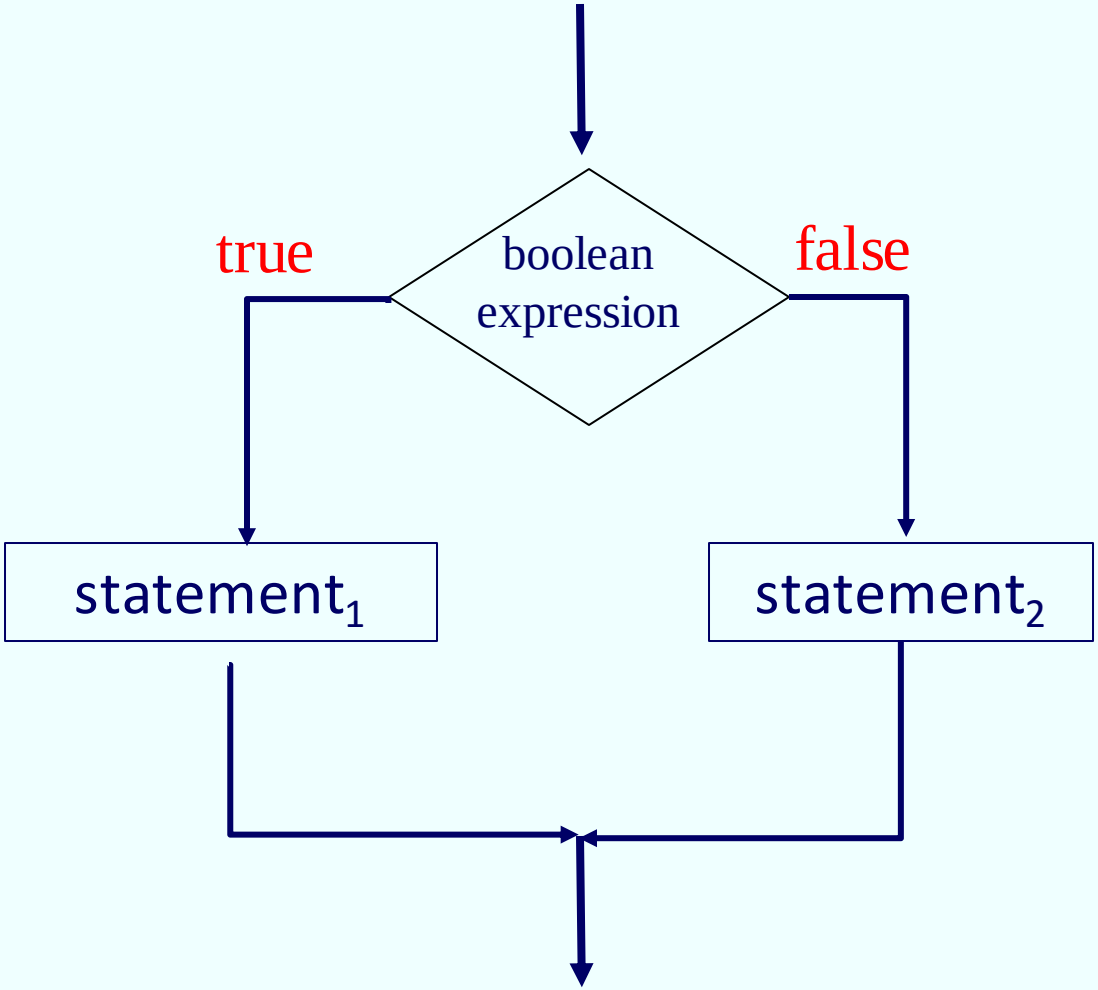
Decision structures : the **if** statement

We say the if is a *decision structure*

… based on the outcome of evaluating the logical expression one of possibly two statements is executed.

Decision structures : the **if** statement



Decision structures : the **if** statement

An **if** statement is coded with a logical expression and either

one statement (statement$_1$)

or

two statements (statement$_1$ and statement$_2$)

written according to syntax as:

if (logical expression)
statement$_1$
else
statement$_2$

The else *clause* is optional

Decision structures : the if statement

if (logical expression)
    statement$_1$
else
    statement$_2$

The else *clause* is optional

When an **if** executes the logical expression is evaluated and :

- when the expression evaluates to true statement$_1$ executes, and then execution of program statements continue at the statement following the **if** ;

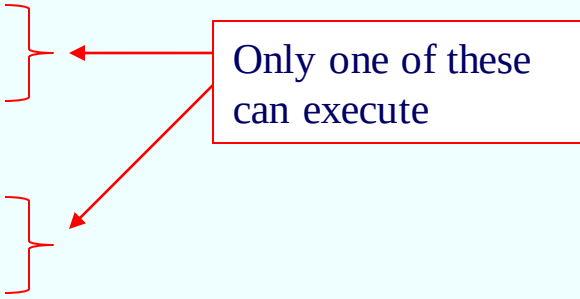
- when the expression evaluates to false statement$_2$ executes, and then execution of program statements continue at the statement following the **if** .

Decision structures : the if statement

Example. `PositiveOrNot`

Gets a number from the user and displays *`positive`* or *`not positive`* accordingly

```
int number = keyboard.nextInt();
System.out.print("the number "+number+" is ");
// Display a message if number is positive or not
if (number > 0) {
   System.out.println("positive");
}
else {
   System.out.println("not positive");
}
```



Strictly speaking, the { } are not required when there is just a single statement
… but it is a common practice to always use compound statements in control structures.

Decision structures : the if statement

Another **example**:

Background:

The Canadian SIN can be tested for a proper check digit.
Part of the process involves multiplying individual digits by either 1 or 2.
When a digit is multiplied by 2 and the result is greater than 9 then the two digits of this product must be added.
For example, if the digit is 8 then the product 2*8 is 16. 16 is greater than 9 and so the sum of its digits is calculated → 1+6 is 7

Along these lines consider a simpler program that:

1. gets a digit from the user
2. multiplies that digit by 2
3. if the product is <= 9 then displays the digit
   otherwise displays the sum of the digits of the product

See next slide →

Decision structures : the if statement

```
import java.util.Scanner;
/**
 * Get a digit from the user and multiply by 2
 * If the result is larger than 9 add its two digits
 */
public class TestGreaterThanNine
{
   public static void main(String[] args)
   {
      Scanner kb = new Scanner(System.in);
      System.out.println("Please enter a single digit:");
      int digit = kb.nextInt();
      int result = digit*2;
      if (result > 9) {
         result = result/10 + result%10;
      }
      System.out.println(" result is "+result);
   }
}
```

`else` clause is not necessary… and so omitted

Decision structures : nested ifs

$statement_1$ or $statement_2$ can be any Java statement

$statement_1$, $statement_2$ can be ifs, whiles, etc.

We can nest one control structure inside another control structure

Example. R**ound Cost Up Down** pages 82-83

Suppose we must handle a purchase transaction in one of two ways:

- If the customer pays cash we must round up/down as there are no pennies.
- If the customers pays by electronic means there is a surcharge of 25 cents.

Decision structures : nested ifs

In *pseudocode*:

1. obtain the type and amount of the purchase
2. **if** the purchase is cash then round the cost up/down
3. **otherwise** add on the surcharge
4. display the total cost for the customer

Steps 2 and 3 can be handled with an if-else

Rounding up/down can be handled with a nested if-else

Code for this is shown on next slide →

Decision structures : the if statement

```
if (typePayment.equals("cash")) {
    if (originalCost % 5 < 3)
       actualCost = originalCost - originalCost%5;
    else
       actualCost = originalCost + (5 - originalCost%5);
}
else
    actualCost = originalCost + 25;
```

Nested `if` to handle rounding for a cash payment

Decision structures : nested ifs

Example. Consider the example beginning on page 84
A table for converting alphabetic grades to a numeric value:

Letter grade	Numeric grade
A	4
B	3
C	2
D	1
F	0

When you need to convert some letter grade to its equivalent numeric value you would look for which row the letter appears in and read off the numeric value in that same row.

E.g. the numeric value corresponding to B is 3

Decision structures : nested ifs

To program this conversion nested ifs are useful

Letter grade	Numeric grade
A	4
B	3
C	2
D	1
F	0

We can structure this code in many ways
We will consider three ways to write the required logic →

Decision structures : nested ifs

Option 1:

```
if (letterGrade.equals("A"))
    numericGrade = 4.0;
if (letterGrade.equals("B"))
    numericGrade = 3.0;
if (letterGrade.equals("C"))
    numericGrade = 2.0;
if (letterGrade.equals("D"))
    numericGrade = 1.0;
```

This style requires the evaluation of expressions that would not be necessary (for example, when the grade is A the other `ifs` do not need to execute, but they do)

Decision structures : nested ifs

Option 2:

```
if (letterGrade.equals("A"))
    numericGrade = 4.0;
else
    if (letterGrade.equals("B"))
        numericGrade = 3.0;
     else
        if (letterGrade.equals("C"))
            numericGrade = 2.0;
         else
            if (letterGrade.equals("D"))
                numericGrade = 1.0;
            else
                numericGrade = 0.0;
```

This style uses indentation properly … each `if` is indented within the outer `if` … but the code easily goes off the page/screen and gets hard to read

Decision structures : nested ifs

Option 3:

```
if (letterGrade.equals("A"))
    numericGrade = 4.0;
else if (letterGrade.equals("B"))
    numericGrade = 3.0;
else if (letterGrade.equals("C"))
    numericGrade = 2.0;
else if (letterGrade.equals("D"))
    numericGrade = 1.0;
else
    numericGrade = 0.0;
```

Due to the similarity of the logical conditions, this style would be favoured by many … it shows the choices at the same level of indentation