# Selection

**Programs with various alternative paths depending on conditions**

# Learning Objectives

**To introduce and become familiar with:**

- **Nature of Algorithms**
- **Concept of Sequence**
- **Selection based on decisions**
- **Decisions in Java**
- **The ‘if’ statement**
- **The ‘if-else’ statement**
- **Nested statements**
- **The ‘switch’ statement**
- **Boolean Operators**
- **Boolean Conditions**

# Algorithms

- A computer program is:

  "**a set of steps or instructions**
  **by which a problem is solved**
  **or a process is carried out**"

- A sequence of steps of this kind is called an **ALGORITHM**

- Previous programs have consisted of simple lists of instructions which were carried out one after the other
  - We say they were carried out in a sequential fashion

- This is the simplest form of algorithm

# Selection / Repetition

- Most problems however, are too complex to be solved by a simple sequence of steps

- Frequently it is necessary to stop and decide - which is the best way to proceed - and often it is necessary, at points in an algorithm, to choose between a range of alternatives

- Choosing between two or more alternative ways of proceeding is referred to as a **SELECTION**

As an aside, sometimes it is necessary to REPEAT a particular process until some desired state of affairs is produced. This is known as **REPETITION** (**ITERATION**)

# Computer Programs

- A computer program is an algorithm and as such it incorporates the THREE structures of:
  - **SEQUENCE**
    - Order
    - Perform one instruction after another
  - **SELECTION**
    - Perform a set of instructions depending on some condition
  - **REPETITION**
    - Perform a set of instructions over and over again

# Decisions …. decisions

- Programming languages provide a facility to:
  - Perform a true/false **TEST**
  - Make a **DECISION** based on the outcome of the TEST, and
  - Direct the computer to jump from one point in a program to another depending on the outcome of the test

- A computer can thus be programmed to:
  - Jump **forward** in a program and skip a sequence of instructions, or
  - jump **backwards** in a program and RE-DO a sequence of instructions

# Decisions …. Decisions

- Decisions are normally based on the outcome of a **COMPARISON** between 2 data values

- Data values may be:
  - Actual (or explicit) values — e.g. 24, ‘A’, 12.99
  - Values held in variables — e.g. myNumber, letter, price
  - Constants/finals — e.g. TARGET, LETTERA, MINCHARGE

- The result of a mathematical expression and these can be compared, using what are called Equality and **Relational Operators**

# Equality & Relational Operators (1)

| COMPARISON | RELATIONAL OPERATOR | EXAMPLE | RESULT |
|---|---|---|---|
| Equal to | == | 6 == 3 | false |
| Not Equal to | != | 6 != 3 | true |
| Greater than | > | 6 > 3 | true |
| Less than | < | 6 < 3 | false |
| Greater than or equal to | >= | 6 >= 3 | true |
| Less than or equal to | <= | 6 <= 3 | false |

# Equality & Relational Operators (2)

| COMPARISON | RELATIONAL OPERATOR | EXAMPLE | RESULT |
|---|---|---|---|
| Equal to | == | 3 == 3 | true |
| Not Equal to | != | 3 != 3 | false |
| Greater than | > | 3 > 3 | false |
| Less than | < | 3 < 3 | false |
| Greater than or equal to | >= | 3 >= 3 | true |
| Less than or equal to | <= | 3 <= 3 | true |

# Format

- A simple comparison has the form:

| ( first value | operator | second value ) |
|---|---|---|

- In Java COMPARISONS ARE ALWAYS placed inside round brackets e.g.

```
(count == 100)                  //   these comparisons will
(days <= 31)                    //   be either true or false
( (x - y) < 0.1)                //   depending on the
( (side1 + side2) > side3)      //   values of the variables
```

# Comparisons

- Comparisons are known as **BOOLEAN expressions**
- **Boolean expressions** return either the value **true** or the value **false**
- Where the individual expressions are numeric the outcome of a comparison will obviously depend on their numeric values
- Also possible to compare characters
- Comparison of 2 characters is based on the numeric value of their ASCII codes
- Letters will always be in alphabetical order so that:

"A' < 'B' < 'C' < 'D' < ..... < 'Y' < 'Z'

**'a' < 'b' < 'c' < 'd' < ..... < 'y' < 'z'**

- So: **'A' < 'N' is true** **'*' < 'T' is true** **'X' <= 'x' is true**

# Examples

- Write different Boolean expressions to check if:
  - A variable called **age** has a value greater than **21**

```
(age > 21)
```

  - A variable called **count** has a value less than **18**

```
(count < 18)
```

  - A variable called **cost** has a value less than or equal to **35.5**

```
(cost <= 35.5)
```

  - A variable called **hours** has a value equal to **50**

```
(hours == 50)
```

  - A variable called **letter** has a value that is not equal to 'C'

```
(letter != 'C')
```

# if Statement

- The simplest form of **SELECTION** is the **if-statement**

- This is used when a line of a program, or a section of a program, is to be executed only if a specified condition is **true**
  - Just as in English
  - IF (you see someone you know) wave at them

- A condition is tested
  - IF it is true then the statement (or sequence of statements) is executed
  - OTHERWISE the statement (or sequence of statements) is skipped

# if Statement - Format

- An **if statement** takes the following format:

```
if (boolean expression) {          //     if the boolean
                                   //     expression is true
   sequence of instructions;       //     execute i.e.
                                   //     PERFORM THE SEQUENCE
   one per line;                   //     OF STATEMENTS
                                   //
   grouped together;               //     OTHERWISE skip the
                                   //     whole sequence
}//if
```

# if Statement

- **E.g. If someone is under 18 years of age, print out a statement saying "You are not an adult“**

```
if (age < 18) {
   System.out.println("You are not an adult");
}//if
```

- **E.g. If you achieve 40 or more in an exam, award a PASS**

```
if (mark >= 40) {
   System.out.println(mark + " is a PASS");
}//if
```

# Single & Compound Statements

- If the 'if' statement controls only 1 statement the curly brackets **{ ... }** may be omitted

```
if (mark >= 40)
    System.out.println("This mark is a pass");
```

- What happens in the following when **mark** has a value 69?

```
if (mark >= 70) {
    System.out.println("This mark is a pass");
    System.out.println("This mark is a grade A");
    System.out.println("Congratulations!");
}//if
```

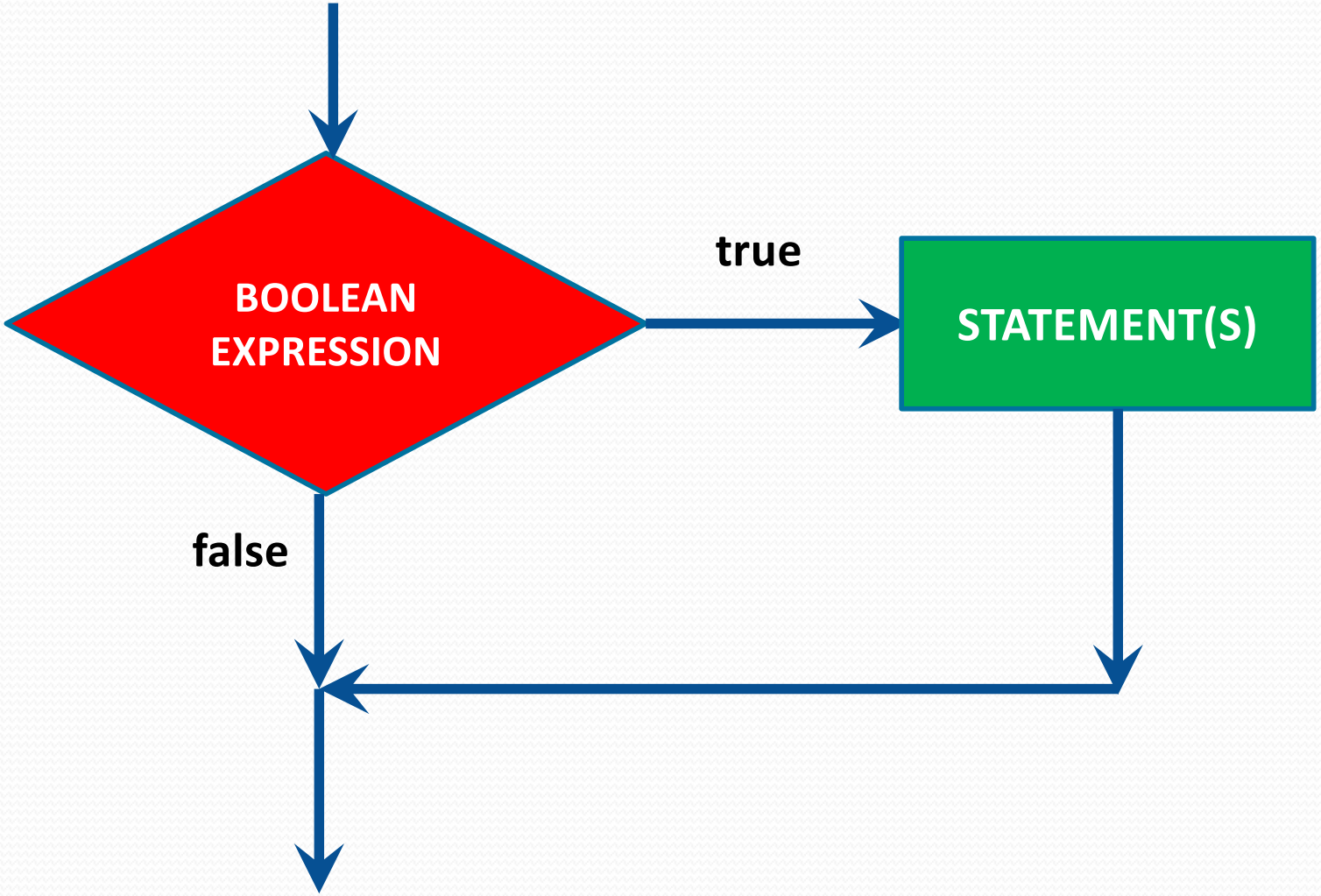
# booleans

- A variable of type '**boolean'** can hold either of the two values **true** or **false.** E.g.

```
boolean found;
…
//somewhere assign found a value eg true

found = true;
…
if (found) {
  …
}//if
```

# if - Diagrammatically



# Example (Tax1.java)

Write a program (Tax1.java) to read in a person's gross pay.

The program should then calculate the person's tax liability and output their gross pay, tax and net pay.

Tax is calculated as follows:

- Up to and including £10,000 – pay no tax
- Pay 20% tax on all earnings over £10,000

# Example (Tax1.java)

Prompt for the gross pay

Read grossPay from the keyboard

```
IF (grossPay > TAXLEVEL)
   Calculate tax = (grossPay – TAXLEVEL) * TAXRATE
```

Calculate netPay = grossPay – tax

Output "Gross Pay: £" + grossPay to 2 decimal places (dps).

Output "Tax: £" + tax to 2 decimal places (dps).

Output "Net Pay £" + netPay to 2 decimal places (dps).

```
double grossPay

final int TAXLEVEL = 10000
double tax = 0
final double TAXRATE = 0.20
double netPay
```

# Example (Tax1.java)

```
18   final int TAXLEVEL = 10000;
19   final double TAXRATE = 0.20;
20   double grossPay, tax = 0, netPay;
21
22   // Prompt for and read in the gross pay
23   System.out.print("Pleas enter your Gross Pay: £");
24   grossPay = keyboard.nextDouble();
25
26   // Calculate tax and net pay
27   if (grossPay > TAXLEVEL){
28      tax = (grossPay - TAXLEVEL) * TAXRATE;
29   }//if
30   netPay = grossPay - tax;
31
32   // Output gross pay, tax and net pay
33   System.out.println("Gross Pay:\t\t£" + df.format(grossPay));
34   System.out.println("Tax:\t\t\t£" + df.format(tax));
35   System.out.println("Net Pay:\t\t£" + df.format(netPay));
```

# QUESTION?

- What is the outcome of the following section of code?

```
int number = 3;

if (number > 1)
   System.out.println ("Number is greater than one");

System.out.println ("I'm always executed … Why?");
```

# The if … else Statement

- Used if a program has to choose between 2 alternative sections of program

- Essentially:
  - A condition is tested
  - if the condition is **true** then one statement or sequence of statements is executed
  - if the condition is **false** then an alternative statement or sequence is executed

# if … else format

```
if (boolean expression) {
   // This is the if sequence of statements;
   // if the above boolean expression is true we execute
   // these statements one after another in the correct
   // sequence - each terminated by a semicolon;
}//if
else {
   // This is the else sequence of statements;
   // If the above boolean expression is false we execute
   // these statements one after another in the correct
   // sequence - each terminated by a semicolon;
}//else
```

# Examples

- If someone is 18 or over, print "You are an adult", otherwise print "You are a child“

```
if (age >= 18) {
        System.out.println("You are an adult");
    }//if
else {
        System.out.println("You are a child");
}//else
```

- If you achieve 40 or more in an exam, award a PASS, otherwise award a FAIL

```
if (mark >= 40) {
        System.out.println(mark + " is a PASS");
    }//if
else {
        System.out.println(mark + " is a FAIL");
}//else
```

# Examples

- If someone is 18 or over, print "You are an adult", otherwise print "You are a child“

```
if (age >= 18)
        System.out.println("You are an adult");
else    System.out.println("You are a child");
```

- If you achieve 40 or more in an exam, award a PASS, otherwise award a FAIL

```
if (mark >= 40)
        System.out.println(mark + " is a PASS");
else    System.out.println(mark + " is a FAIL");
```

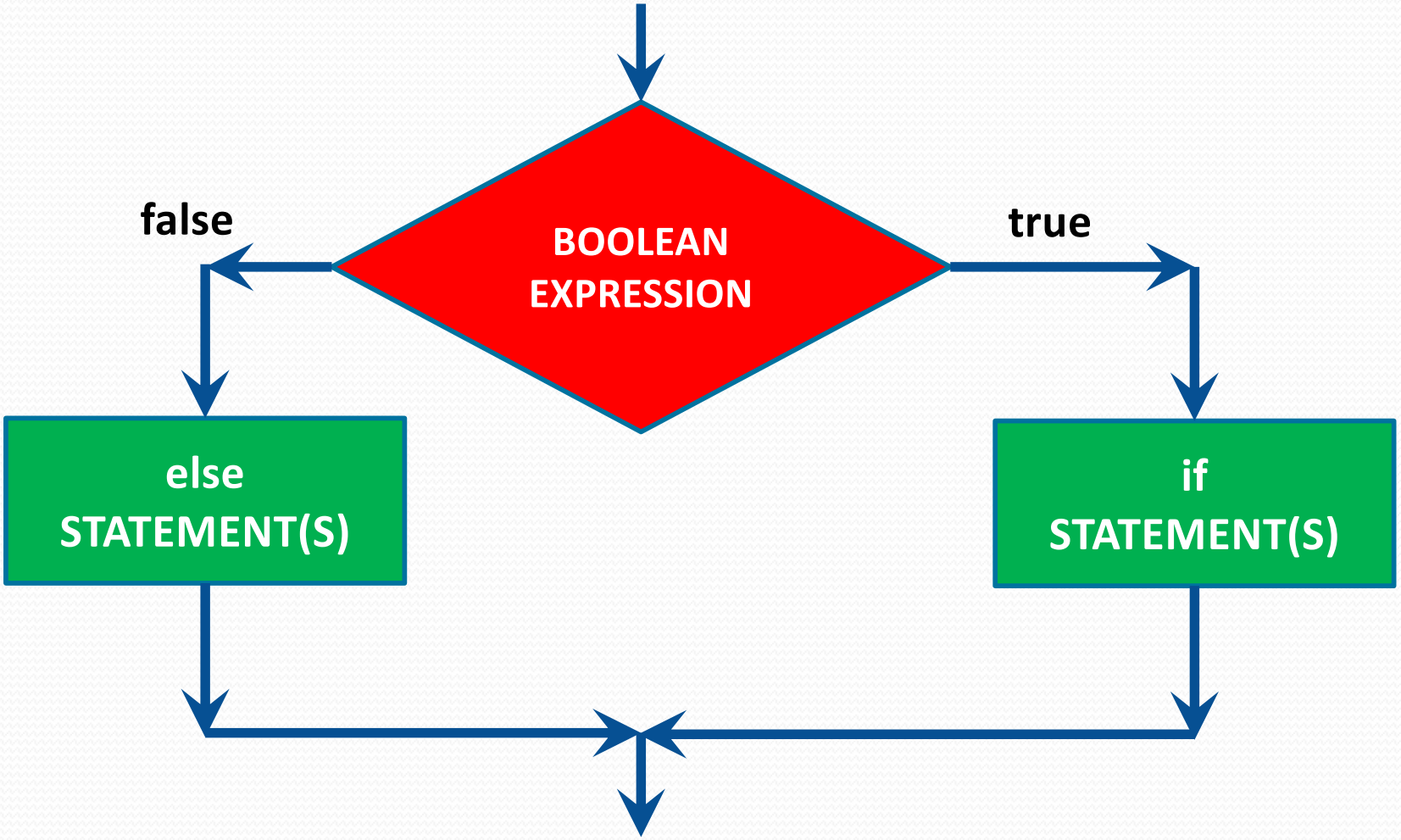
# Example

```
if (mark >= 40) {
   System.out.println ("Mark is a pass");
   System.out.println ("Proceed to Year 2");
   System.out.println ("Congratulations! ");
}//if
else {
   System.out.println("Mark is a fail");
   System.out.println("You must resit");
   System.out.println("Ruins your summer!");
}//else
```

# Blocks

- A **block** (or compound statement) is a sequence of statements enclosed in curly brackets {...}
- Blocks can refer to more than one statement
- Blocks can thus be treated as a single entity

# if … else - Diagrammatically



# Example (Tax2.java)

Prompt for the gross pay

Read grossPay from the keyboard

double grossPay

IF (grossPay > TAXLEVEL)

    Calculate tax = (grossPay – TAXLEVEL) * TAXRATE

ELSE

    Calculate tax = 0

final int TAXLEVEL = 10000

double tax

final double TAXRATE = 0.20

Calculate netPay = grossPay – tax

double netPay

Output "Gross Pay:  £" + grossPay to 2 d.p.

Output "Tax:£" + tax to 2 d.p.

Output "Net Pay  £" + netPay to 2 d.p.

# Example (Tax2.java)

```
18   final int TAXLEVEL = 10000;
19   final double TAXRATE = 0.20;
20   double grossPay, tax, netPay;
21
22   // Prompt for and read in the gross pay
23   System.out.print("Please enter your Gross Pay: £");
24   grossPay = keyboard.nextDouble();
25
26   // Calculate tax and net pay
27   if (grossPay > TAXLEVEL){
28      tax = (grossPay - TAXLEVEL) * TAXRATE;
29   }//if
30   else {
31      tax = 0;
32   }//else
33   netPay = grossPay - tax;
34
35   // Output gross pay, tax and net pay
36   System.out.println("Gross Pay:\t\t£" + df.format(grossPay));
37   System.out.println("Tax:\t\t\t£" + df.format(tax));
38   System.out.println("Net Pay:\t\t£" + df.format(netPay));
```

# Nested ifs

- Can be very messy
- Can be difficult to read
- Can be difficult to code
- Is there a better way?
  - Sometimes …. YES!

# Nested if statement

- **Example: Grades are awarded as follows:**

| | |
|---|---|
| 70+ | DISTINCTION |
| 40 - 69 | PASS |
| < 40 | FAIL |

```
IF (mark >= 70)
        Output  mark + " is a DISTINCTION"
    ELSE
        IF (mark >= 40)
            Output  mark + " is a PASS"
        ELSE
            Output  mark + " is a FAIL"
```

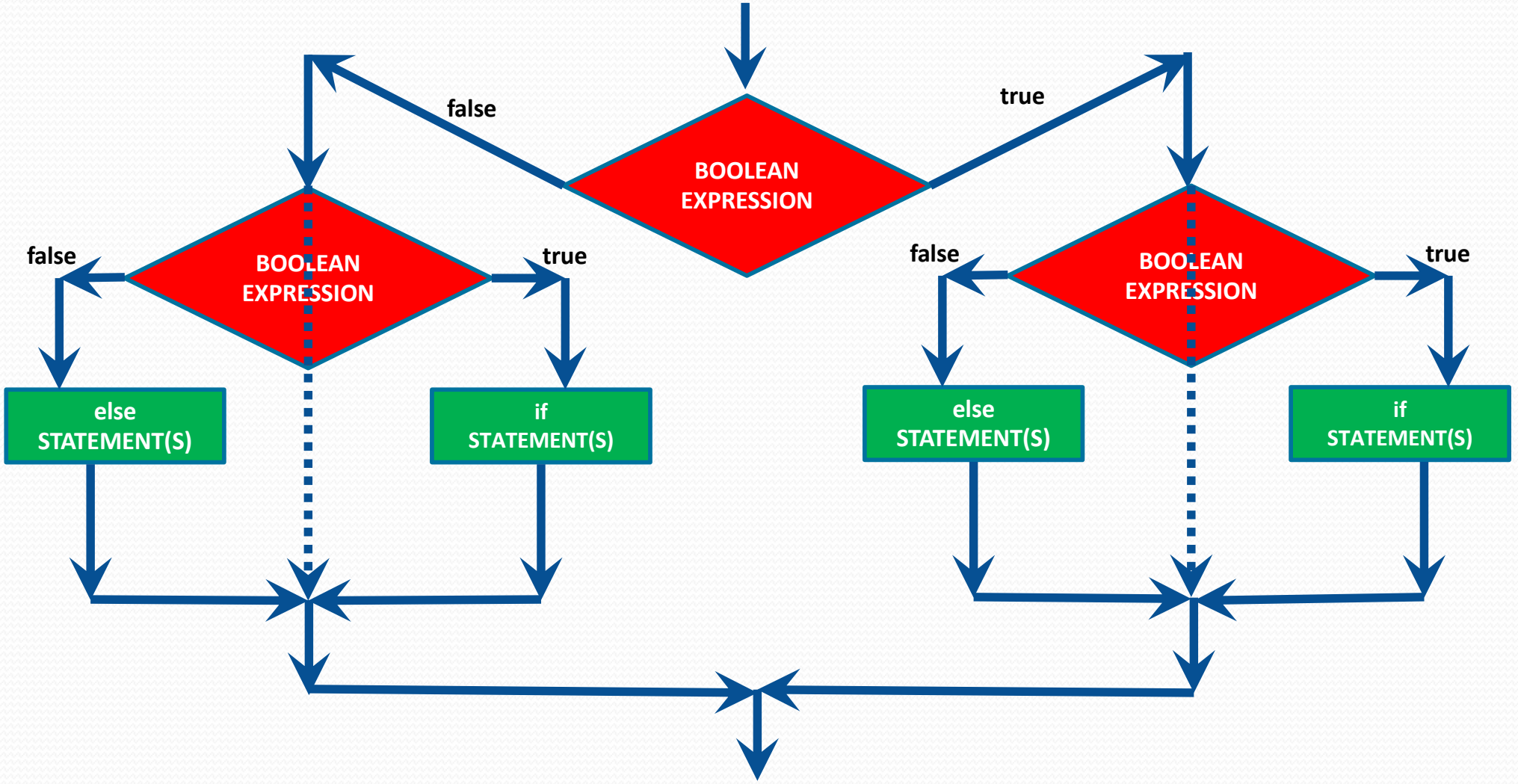
# Nested if statement

- **Example:** **Grades are awarded as follows:**

| | |
|---|---|
| **DISTINCTION** | **70+** |
| **PASS** | **40 to 69** |
| **FAIL** | **< 40** |

```
if (mark >= 70)
   System.out.println(mark + " is a DISTINCTION");
else  {
   //   If we get here it implies mark is < 70
   if (mark >= 40)
      System.out.println(mark + " is a PASS");
   else
     System.out.println(mark + " is a FAIL");
}
```

# Nested if … else - Diagrammatically



# Example (Tax3.java)

Write a program (Tax3.java) to read in a person's gross pay then calculate their tax and output their gross pay, tax and net pay.

Tax is calculated as follows:

- Up to £10,000 – no tax
- 20% tax on all earnings greater than £10,000, up to £40,000
- 40% tax on all earnings greater than £40,000

# Example (Tax3.java)

Prompt for the gross pay

Read grossPay from the keyboard

IF (grossPay > HIGHTAXLEVEL)

    Calculate tax = ((grossPay – HIGHTAXLEVEL) * HIGHTAXRATE) +

        ((HIGHTAXLEVEL – LOWTAXLEVEL) * LOWTAXRATE)

ELSE

    IF (grossPay > LOWTAXLEVEL)

        Calculate  tax = (grossPay – LOWTAXLEVEL) * LOWTAXRATE

Calculate netPay = grossPay – tax

Output "Gross Pay:  £" + grossPay to 2 d.p.

Output "Tax:  £" + tax to 2 d.p.

Output "Net Pay  £" + netPay to 2 d.p.

double grossPay

final int HIGHTAXLEVEL = 40000

double tax = 0

final double HIGHTAXRATE = 0.40

final int LOWTAXLEVEL = 10000

final double LOWTAXRATE = 0.20

double netPay

# Example
## (Tax3.java)

```
18   final int LOWTAXLEVEL = 10000, HIGHTAXLEVEL = 40000;
19   final double LOWTAXRATE = 0.20, HIGHTAXRATE = 0.40;
20   double grossPay, tax = 0, netPay;
21
22   // Prompt for and read in the gross pay
23   System.out.print("Pleas enter your Gross Pay: £");
24   grossPay = keyboard.nextDouble();
25
26   // Calculate tax and net pay
27   if (grossPay > HIGHTAXLEVEL){
28      tax = ((grossPay - HIGHTAXLEVEL) * HIGHTAXRATE) +
29         ((HIGHTAXLEVEL - LOWTAXLEVEL) * LOWTAXRATE);
30    }//if
31   else {
32      if (grossPay > LOWLEVELTAX){
33         tax = (grossPay - LOWTAXLEVEL) * LOWTAXRATE;
34      }//if
35   }//else
36   netPay = grossPay - tax;
37
38   // Output gross pay, tax and net pay
39   System.out.println("\nGross Pay:\t\t£" + df.format(grossPay));
40   System.out.println("Tax:\t\t\t£" + df.format(tax));
41   System.out.println("Net Pay:\t\t£" + df.format(netPay));
```

# Notes on if … else structure

- There are two forms of if statement:

```
if (logical expression)
   statement(s)
else
   statement(s)
```

```
if (logical expression)
   statement(s)
```

- The expression in an **`if`** or **`if … else`** structure is a logical expression

- There is NO stand-alone **`else`** statement in Java
  - Every **`else`** has a related **`if`**

- An **`else`** is paired with the most recent **`if`** that has not been paired with any other **`else`**

# switch Statements ….

- A series of nested if-then statements can be used to choose between a number of alternative processes BUT this involves stepping through the various decisions to reach the sequence of statements to be executed

- An alternative method of choosing between a range of options is provided by what is called in Java or C as a **‘SWITCH statement’**

- This involves choosing one of a selection of processes depending upon the value of an expression called the **‘switch expression’**

# switch (case ) Statement

Look at the following section of code and determine what it is trying to do:

```
System.out.println ("1. \t Hockey");
System.out.println ("2. \t Football");
System.out.println ("3. \t Rugby");
System.out.println ("4. \t Exit System");

System.out.print ("\nEnter a menu choice: ");
int choice = keyboard.nextInt();
```

What does the above menu system offer to the user?

What would the user enter?

# switch Statement Format

```
switch (option) {

   case option1:
        sequence of instructions;
        break;
   case option2:
        sequence of instructions;
        break;
   …
   default:
       sequence of instructions;
}//switch
```

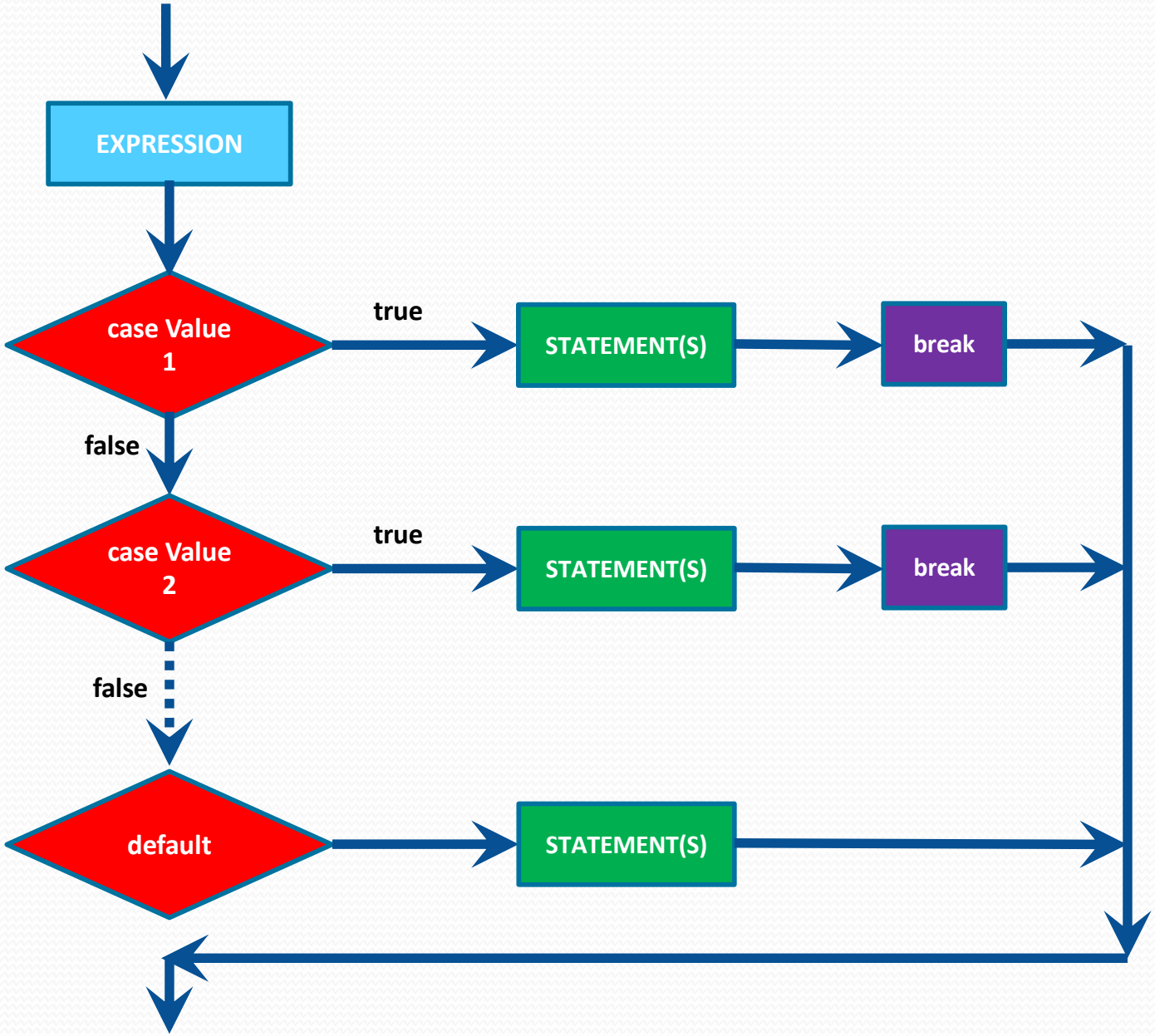
# Menu Example

```
switch (choice) {

   case 1: System.out.println("World Cup Hockey (Argentina) -
2015");
           …
           break;
   case 2: System.out.println("World Cup Football (Rio) - 2014");
           …
           break;
   case 3: System.out.println("World Cup Rugby (England) - 2015");
           …
           break;
   case 4: System.out.println("You have chosen to EXIT the system");
           …
           break;

   default:System.out.println("You have not entered 1, 2, 3 or 4");
```

# switch- Diagrammatically



# Example

**Read in a month number (1-12), then output number of days in a month (assume non-leap year)**

```
Prompt user to "Enter the month (1 – 12) : "
Read month from keyboard                                    int month

   SWITCH (month)

        1, 3, 5, 7, 8, 10, 12:
              set noOfDays = 31                             int noOfDays = 0
        2:
              set noOfDays = 28
        4, 6, 9, 11:
              Set noOfDays = 30

 Output "There are " + noOfDays + " days in month " + month);
```

# Example

**Read in a month number (1-12), then output number of days in a month (assume non-leap year)**

```
int month, noOfDays  = 0;
System.out.print("Enter the month (1 - 12) : ");
month = keyboard.nextInt();

switch (month) {
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
       noOfDays = 31;
       break;
    case 2:
       noOfDays = 28;
       break;
    case 4: case 6: case 9: case 11:
       noOfDays = 30;
       break;
}//switch
System.out.println("There are " + noOfDays + " days in month "
                   + month);
```

# Using the previous example ….

- **Note 1:**
  - We inspect the contents of the variable called `month` to give us the **switch expression**
  - The **value** of the *switch expression* determines which **case value** is chosen and hence which sequence of statements is executed

- **Note 2:**
  - There can be a single case value or a series of case values
  - If there are a series of case values, the various case values must be separated separated by colons **(:)**

# More details

- **Note 3:**
  - The switch expression is usually *either* an integer *or* a character
  - NEVER A String
- **Note 4:**
  - Each case value must be of the same data type as the value returned by the switch expression
  - In our example they must be integers
- **Note 5:**
  - The **break** command causes the program to jump out of the switch statement at that point and continue with the rest of the program

# Default Option

- Our example works correctly in most cases
- It does not take into account what would happen if the switch expression (month) did not correspond to any of the case values
- We can include an option in the list of case values to take account of any other value
- We do this by using the word **`default`** followed by the sequence of statements to be executed should the situation require it

# if … else versus switch

- There are no fixed rules that can be applied to decide whether to use an **`if … else`** or **`switch`** structure

- A **`switch`** statement is an elegant way to implement multiple selections

- If the range of values is infinite and you cannot reduce them to a set containing a finite number of values, you must use the **`if … else`** structure

# Boolean Variables

- A variable of type ‘boolean’ can hold either of the two values true or false
- The boolean variable can be used in the if statement

```
int number;  boolean found;

System.out.print("Enter a number: ");
number = keyboard.nextInt();
found = (number == 100);
if (found) {
   …
}//if
else {
   …
}//else
```

# Logical Operators

- Two or more boolean expressions can be combined using the logical operators '**and**' and '**or**'

- An expression can also be converted from **true** to **false** or vice versa using the operator '**not**'

| Logical Operator | Java Symbol |
|---|---|
| and | && |
| or | \|\| |
| not | ! |

# Logical Operator - &&

- If two boolean expressions are combined using **&&** (meaning **and**) the result will be **true** if and only if the two individual boolean expressions are **true**

- It either or both the individual expressions are **false** then the whole expression will be **false**

| Condition1 | Condition2 | Condition1 && Condition2 |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

# Example

A section of program to read in an examination mark and check if it is in the range 0..100

```
Prompt for mark
Read mark from the keyboard                    int mark

IF ( (mark >=0) AND (mark <= 100) )
    Output "Mark entered is valid"
ELSE
    Output "Not a valid mark"
```

# Example

A section of program to read in an examination mark and check if it is in the range 0 ..100

```
int mark;
System.out.print("Enter your mark: ");
mark = keyboard.nextInt();

if ( (mark >=0) && (mark <= 100) ) {
   System.out.println("Mark entered is valid");
}//if
else {
   System.out.println("Not a valid mark");
}//else
```

# Logical Operator - ||

- If two boolean expressions are combined using **||** (meaning **or**) the result will be true if either or both the individual expressions are true

- If both the individual expressions are **false** then the whole expression will be **false**

| Condition1 | Condition2 | Condition1 \|\| Condition2 |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

# Example

A section of program to read in an examination mark and check if it is in the range 0 ..100

```
int mark;
System.out.print("Enter your mark: ");
mark = keyboard.nextInt();

if ( (mark < 0) || (mark > 100) ) {
    System.out.println("Not a valid mark");
}//if
else {
   System.out.println("Mark entered is valid ");
}//else
```

# Example

A section of program to read in a character and check if the character is ‘x’ or ‘y’:

```
Prompt for a character
Read character from the keyboard                char letter

IF ( (letter = ‘x’) OR (letter = ‘y’) )
    Output "The character is x or y"
ELSE
    Output "The character is NOT x or y"
```

# Example

A section of program to read in a character and check if the character is ‘x’ or ‘y’:

```
char letter;
System.out.print("Enter a character: ");
letter = keyboard.nextLine().charAt(0);

if ( (letter == 'x') || (letter == 'y') ) {
   System.out.println("The character is x or y");
}//if
else {
   System.out.println("The character is NOT x or y");
}//else
```

# Logical Operator - !

- If the logical operator **!** (meaning **not**) is placed in front of a boolean expression it will convert **true** into **false** and vice versa

| Condition | !Condition |
|---|---|
| true | false |
| false | true |

# Example

We can also put the not symbol (!) in front of a boolean variable. For example

```
boolean found = false;
…
// Assume number and searchValue have been assigned values
if (number == searchValue) {
    found = true;
}//if
        …
if (!(found)) {
    System.out.println("Sorry, you have not found the number");
}//if
```

Could have used :

```
if (!found)
```

# Example

```
boolean found, finished;

….


if (!(found) && !(finished)) {

   // statements executed if not found
   // and not finished

}//if
```

# Order of Precedence

- Where there is a mixture of mathematical operators the normal laws of operator precedence apply:

| | | | | |
|---|---|---|---|---|
| **Unary** | **!** | | | |
| **Multiplication & Division** | ***** | **/** | **%** | |
| **Addition & Subtraction** | **+** | **-** | | |
| **Relational** | **<** | **<=** | **>** | **>=** |
| **Equality** | **==** | **!=** | | |
| **AND** | **&&** | | | |
| **OR** | **\|\|** | | | |

# Order of Precedence

- Example:

(17 < 4 * 3 + 5) || (8 * 2 == 4 * 4) && !(3 + 3 == 6)

= (17 < 12 + 5) || (16 == 16) && !(6 == 6)

= (17 < 17) || true && !(true)

= false || true && false

= false || false

= false

# Questions

- What are the 2 most common activities provided by control structures?
- What are the 4 relational operators in Java?
- What are the 2 equality operators in Java?
- **What is the result of a logical (boolean) expression?**
- **What are the 3 logical operators in Java?**
- **What are the 3 selection structures in Java?**

**CHALLENGE**

**What is the difference between && and & (also || and |)?**