Universidad
Carlos III de Madrid
www.uc3m.es

Lesson 3
Control flow statements

Programming

Grade in Computer Engineering

Outline

1. Decision-making statements
   - if-else
   - switch
2. Looping statements
   - while
   - do-while
   - for
3. Branching statements
   - break
   - continue
   - System.exit

Outline

1. **Decision-making statements**
   - **if-else**
   - switch
2. Looping statements
   - while
   - do-while
   - for
3. Branching statements
   - break
   - continue
   - System.exit

Java program instructions are executed from top to bottom, in the order that they appear, starting from the first sentence inside the `main` method

Control flow instructions break up this sequence

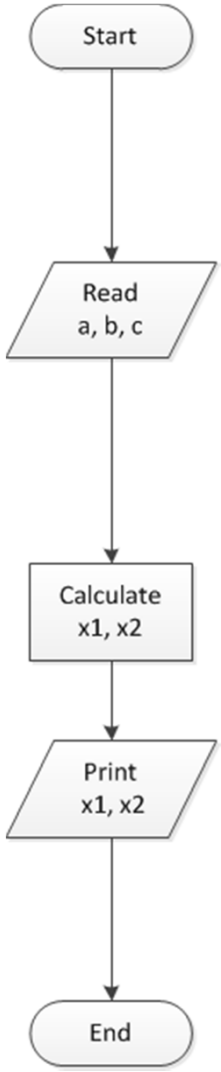
Blocks of code are executed or not depending on some *conditions*

Conditions are *boolean* expressions –relational and logical expressions

1. Decision-making statements

Second grade equation revisited

```java
import java.util.Scanner;

public class EquationSimple {

    public static void main(String[] args) {

        /* Declare variables */
        double a, b, c;

        /* Read values from the keyboard */
        Scanner sc = new Scanner(System.in);
        System.out.print("a? ");
        a = sc.nextDouble();
        System.out.print("b? ");
        b = sc.nextDouble();
        System.out.print("c? ");
        c = sc.nextDouble();

        /* Calculate solutions */
        double x1, x2;
        x1 = (-b + Math.sqrt(b*b - 4*a*c) ) / (2 * a);
        x2 = (-b - Math.sqrt(b*b - 4*a*c) ) / (2 * a);

        /* Print results on the screen */
        System.out.println("Equation: ");
        System.out.println(a + "x^2 + " + b + "x + " + c + " = 0");
        System.out.println();
        System.out.println("Solutions: ");
        System.out.println("x1: " + x1);
        System.out.println("x2: " + x2);
    }

}
```




```
Console
<terminated> EquationSimple [Java Application]
a? 1
b? 2
c? -3
Equation:
1.0x^2 + 2.0x + -3.0 = 0

Solutions:
x1: 1.0
x2: -3.0
```

EquationSimple.java

jgromero@inf.uc3m.es

1. Decision-making statements

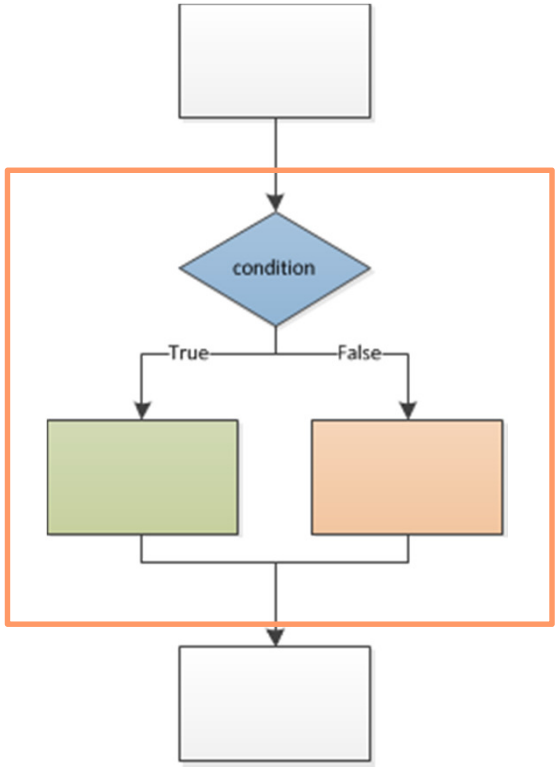
if-else

The most basic control flow instruction

If the condition is **true**, the block of code associated to the **`if`** part is executed

If the condition is **false**, the block of code associated to the **`else`** part is executed

`if-else` statements can be nested




```
if (<boolean expression>) {
  <statement(s)>
} else {
  <statement(s)>
}
```

1. Decision-making statements

Second grade equation revisited

EquationBetter.java

```
public static void main(String[] args) {

    /* Declare variables */
    double a, b, c;

    /* Read values from the keyboard */
    Scanner sc = new Scanner(System.in);
    System.out.print("a? ");
    a = sc.nextDouble();
    System.out.print("b? ");
    b = sc.nextDouble();
    System.out.print("c? ");
    c = sc.nextDouble();

    /* Calculate solutions */

    // (if discriminant is positive, real roots)
    // (otherwise, roots are complex)
    double discriminant;
    discriminant = b*b - 4*a*c;

    System.out.println("\nEquation: ");
    System.out.println(a + "x^2 + " + b + "x + " + c + " = 0");
    System.out.println();

    if(discriminant >= 0) {
        double x1, x2;
        x1 = (-b + Math.sqrt(discriminant)) / (2 * a);
        x2 = (-b - Math.sqrt(discriminant)) / (2 * a);

        /* Print results on the screen */
        System.out.println("Solutions (real roots): ");
        System.out.println("x1: " + x1);
        System.out.println("x2: " + x2);

    } else {
        System.out.println("The equation has no real roots");
    }

    System.out.println("\nEnd of example");
}
```

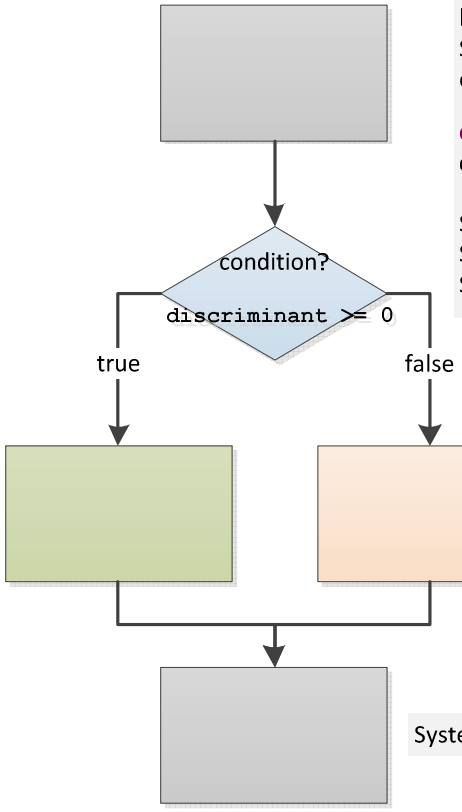
Console

```
<terminated> EquationBetter [Java
a? -2
b? 2
c? 1

Equation:
-2.0x^2 + 2.0x + 1.0 = 0

Solutions (real roots):
x1: -0.3660254037844386
x2: 1.3660254037844386

End of example
```

1. Decision-making statements

Second grade equation revisited

```
double a, b, c;

Scanner sc = new Scanner(System.in);
System.out.print("a? ");
a = sc.nextDouble();
System.out.print("b? ");
b = sc.nextDouble();
System.out.print("c? ");
c = sc.nextDouble();

double discriminant;
discriminant = b*b - 4*a*c;

System.out.println("\nEquation: ");
System.out.println(a + "x^2 + " + b + "x + " + c + " = 0");
System.out.println();
```



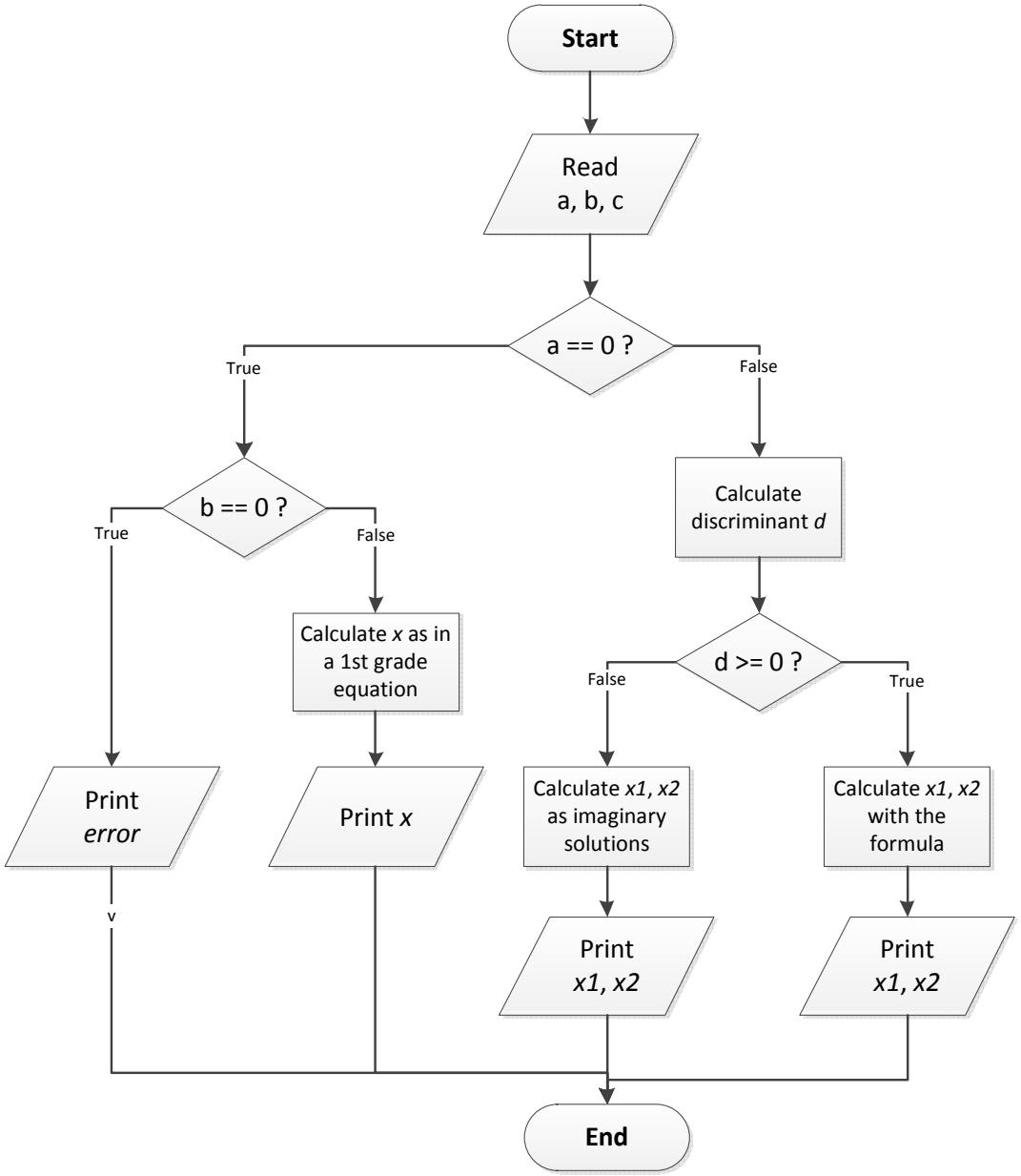

```
double x1, x2;
x1 = (-b + Math.sqrt(discriminant)) / (2 * a);
x2 = (-b - Math.sqrt(discriminant)) / (2 * a);

System.out.println("Solutions (real roots): ");
System.out.println("x1: " + x1);
System.out.println("x2: " + x2);
```

```
System.out.println("The equation has no real solutions");
```

```
System.out.println("\nEnd of example");
```

jgromero@inf.uc3m.es

1. Decision-making statements



1. Decision-making statements

Second grade equation revisited

EquationBest.java

```
public class EquationBest {

    public static void main(String[] args) {

        /* Read values from the keyboard */
        double a = 0,
               b = 0,
               c = 0;
        try {
            Scanner sc = new Scanner(System.in);
            System.out.print("a? ");
            a = sc.nextDouble();
            System.out.print("b? ");
            b = sc.nextDouble();
            System.out.print("c? ");
            c = sc.nextDouble();
        } catch (InputMismatchException e) {
            System.out.println("[ERROR] Wrong number format.");
            System.exit(-1);
        }
```

Exception management

`try-catch` is used to detect if there was an `InputMismatchException` error when reading from the keyboard

If this error happens, the `catch` block is executed; otherwise, the `catch` block is not executed

1. Decision-making statements

Second grade equation revisited

```
/* Solve equation */
// (if a is 0, this is not a second grade polynomial)
if(a == 0) {
    System.out.println("This is not a quadratic equation.");
    System.out.println("a value is 0.");

    // (if b is 0, this is not a polynomial)
    if(b == 0) {
        System.out.println("[ERROR] This is not an equation.");
        System.out.println("b value is 0.");
    } else {
        double x1;
        x1 = -c / b;
        System.out.println("x --> " + x1);
    }

} else {
```

Nested conditions

Nested if-else allow the programmer to define multiple execution branches depending on variable values

1. Decision-making statements

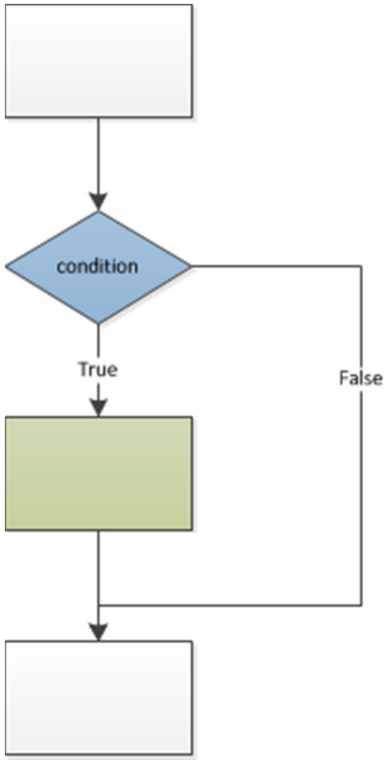
Second grade equation revisited

```
        } else {

            // (if discriminant is positive, real roots)
            // (otherwise, roots are complex)
            double discriminant;
            discriminant = b*b - 4*a*c;

            double x1, x2;
            System.out.println("\nEquation: ");
            System.out.println(a + "x^2 + " + b + "x + " + c + " = 0");
            System.out.println();

            if(discriminant > 0) {
                x1 = (-b + Math.sqrt(discriminant)) / (2 * a);
                x2 = (-b - Math.sqrt(discriminant)) / (2 * a);
                System.out.println("Real roots");
                System.out.println("x1 --> " + x1);
                System.out.println("x2 --> " + x2);
            } else {
                double r, complex;
                r = -b / (2 * a);
                complex = Math.sqrt(4*a*c - b*b) / (2 * a);
                System.out.println("Complex roots");
                System.out.println("x1 --> " + r + " + " + complex + "i");
                System.out.println("x2 --> " + r + " - " + complex + "i");
            }
        }
    }
}
```

1. Decision-making statements

if-else

The **else** part is optional




```
if (<boolean expression>) {
  <statement(s)>
}
```

If there is only one instruction inside the block, the **braces** can be removed

```
if (<boolean expression>)
  <statement>
```

```
if (<boolean expression>)
  <statement>
else
  <statement>
```

Do not forget the **;**

if else instructions can be **nested**

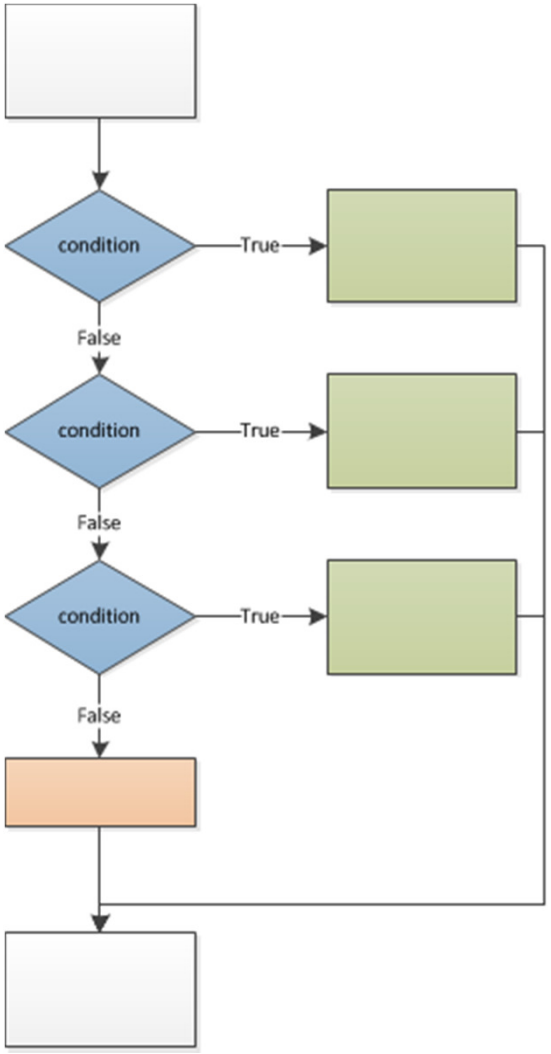
1. Decision-making statements

if-else-if

Nested **`if-else`** instructions can be arranged to implement mutually exclusive execution branches

- Only one of the blocks is executed
- If none of the conditions is true, the final **`else`** block is executed

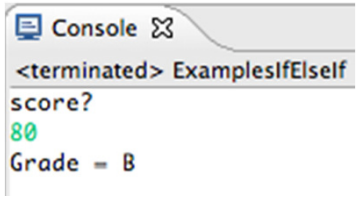
```
if (<boolean expression 1>) {
  <statement(s)>
} else if (<boolean expression 2>) {
  <statement(s)>
} else if (<boolean expression 3>) {
…
} else {
  <statement(s)>
}
```



1. Decision-making statements

if-else-if

```
Scanner sc = new Scanner(System.in);
System.out.println("score? ");
int testscore = sc.nextInt();

char grade;

if (testscore >= 90) {
    grade = 'A';
} else if (testscore >= 80) {
    grade = 'B';
} else if (testscore >= 70) {
    grade = 'C';
} else if (testscore >= 60) {
    grade = 'D';
} else {
    grade = 'F';
}

System.out.println("Grade = " + grade);
```



ExamplesIfElseIf.java

Outline

1. **Decision-making statements**
   - if-else
   - **switch**
2. Looping statements
   - while
   - do-while
   - for
3. Branching statements
   - break
   - continue
   - System.exit

Allows for multiple execution paths, depending on the value of the *switch variable*

The *switch variable* must be integer, character, string or enumerated value

If the *switch variable* is equal to the value of a **`case`**, the sentences following the `case` are executed **until a `break` is found**

If a `case` block does not have a `break`, the execution continues in the next `case`, even if it is false!

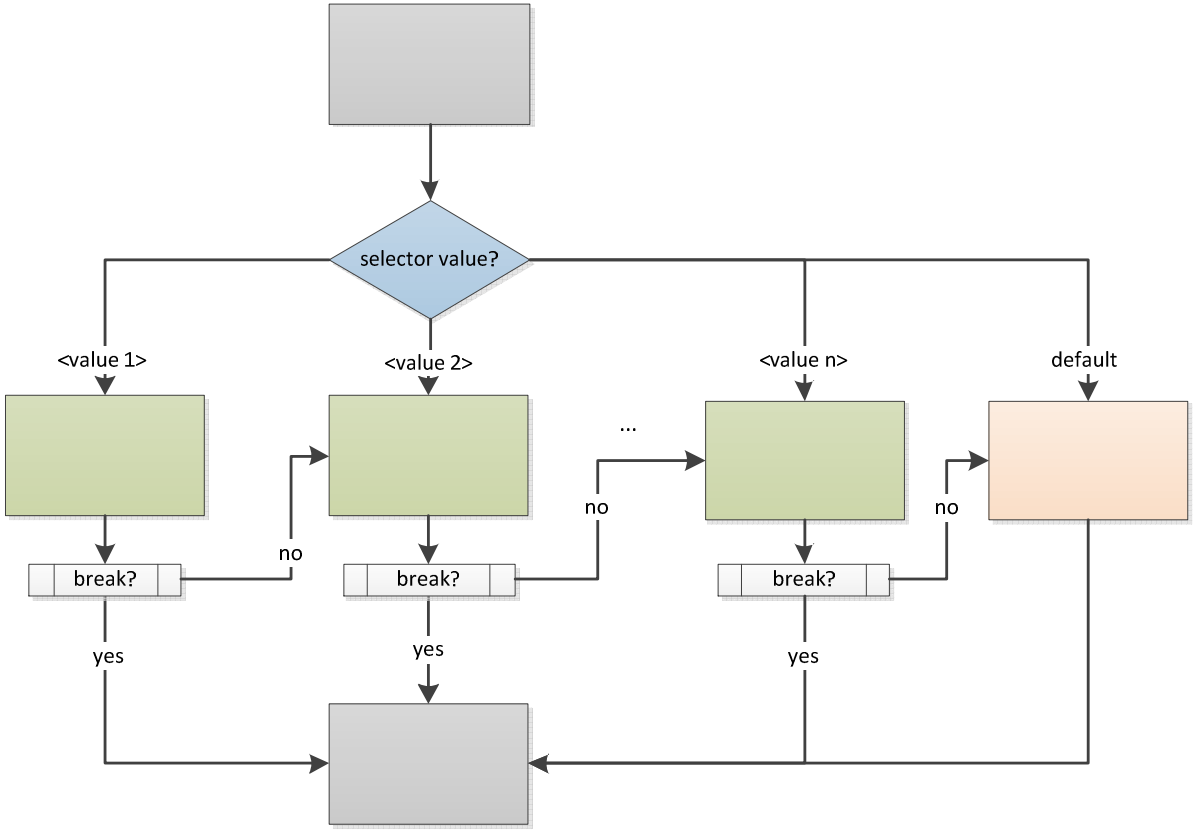
Braces are not required to delimit each `case`

If no `case` is true, the **`default`** block (if defined –since it is optional) is executed

Deciding between `switch` and `if-else-if` is based on the type of the *switch variable* and code readability

1. Decision-making statements

switch

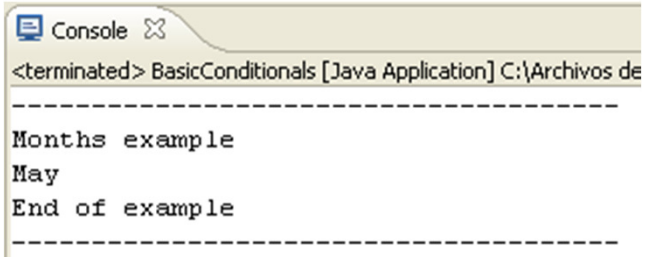



```
switch (<variable>) {
  case <value 1>:
    <sentence(s)>
    [break;]
  case <value 2>:
    <sentence(s)>
    [break;]
  default:
    <sentence(s)>
}
```

1. Decision-making statements

switch

```
/* switch */
// Months
System.out.println("-------------------------------------");
System.out.println("Months example");

int month = 5;
switch(month) {
    case 1:
        System.out.println("January");
        break;
    case 2:
        System.out.println("February");
        break;
    case 3:
        System.out.println("March");
        break;
    case 4:
        System.out.println("April");
        break;
    case 5:
        System.out.println("May");
        break;
    case 6:
        System.out.println("June");
        break;
    case 7:
        System.out.println("July");
        break;
    case 8:
        System.out.println("August");
        break;
    case 9:
        System.out.println("September");
        break;
    case 10:
        System.out.println("October");
        break;
    case 11:
        System.out.println("November");
        break;
    case 12:
        System.out.println("December");
        break;
    default:
        System.out.println("Invalid month.");
        break;
}
```



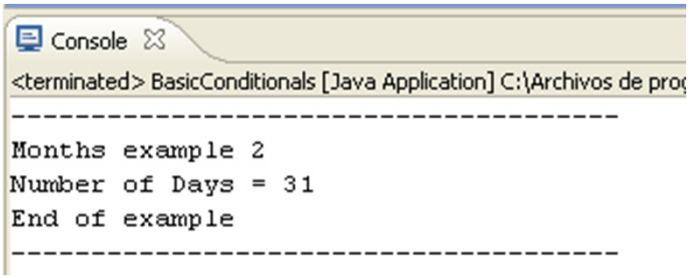
BasicConditionals.java

jgromero@inf.uc3m.es

1. Decision-making statements

switch + if-else

```
System.out.println("---------------------------------------");
System.out.println("Months example 2");

int month_2 = 5;
int year = 2009;
int numDays = -1;

switch(month_2) {
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
    numDays = 31;
    break;
case 4:
case 6:
case 9:
case 11:
    numDays = 30;
    break;
case 2:
    if ( ((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0) )
        numDays = 29;
    else
        numDays = 28;
    break;
default:
    System.out.println("Invalid month.");
    break;
}
System.out.println("Number of Days = " + numDays);
System.out.println("End of example");
System.out.println("---------------------------------------");
```



BasicConditionals.java

Outline

1. Decision-making statements
   - if-else
   - switch
2. **Looping statements**
   - **while**
   - do-while
   - for
3. Branching statements
   - break
   - continue
   - System.exit

Loops repeat sequentially the instructions in a block of code while a condition holds

- When the block of code associated to a loop instruction is finished, the condition is tested
- If the condition holds, the block is executed again
- If the condition does not hold, the execution continues with the instructions below the block

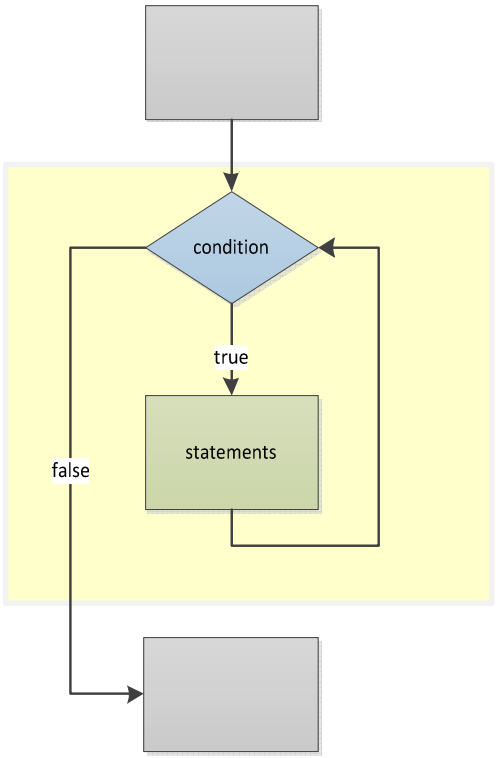
2. Looping statements
while

Continually executes a block of statements while a particular condition is true

When the program reaches the **`while`** statement for the first time,

- If the condition is true, the block of code associated to the **`while`** is executed
- If the condition is false, the program continues by the sentence below the block

After finishing the **`while`** block, the condition is tested again




```
while(<boolean expression>) {
  <sentence(s)>
}
```

2. Looping statements

while

```
/* while */
// Print even numbers between 1 and 100

System.out.println("---------------------------------------");
System.out.println("Even numbers example");
int n = 1;

while(n <= 100) {

    if(n % 2 == 0) {     // n is even
        System.out.println(n + " is even.");
    }

    n = n + 1;           // increment n
}

System.out.println("End of example");
System.out.println("---------------------------------------");
```

```
Console
<terminated> ExamplesWhile [Java App
---------------------------------
Even numbers example
2 is even.
4 is even.
6 is even.
8 is even.
10 is even.
12 is even.
14 is even.
16 is even.
18 is even.
20 is even.
22 is even.
24 is even.
26 is even.
28 is even.
30 is even.
32 is even.
34 is even.
36 is even.
38 is even.
40 is even.
42 is even.
44 is even.
46 is even.
48 is even.
50 is even.
```

ExamplesWhile.java

2. Looping statements

while

```
// Add numbers read from the keyboard
System.out.println("---------------------------------------");
System.out.println("Read values while the user wants to continue");

String input = "";
int addition = 0;
Scanner sc = new Scanner(System.in);

while( !input.equals("n") ) {
    System.out.print("value: ");
    int x = sc.nextInt();
    addition += x;

    System.out.print("continue (y/n)? ");
    input = sc.next();
}

System.out.println("The sum of the values is: " + addition);

System.out.println("End of example");
System.out.println("---------------------------------------");
```

```
Console
<terminated> ExamplesWhile [Java Application] /Sys
---------------------------------------
Read values while the user wants to cont
value: 2
continue (y/n)? y
value: 3
continue (y/n)? y
value: 5
continue (y/n)? y
value: 10
continue (y/n)? y
value: 1
continue (y/n)? n
The sum of the values is: 21
End of example
---------------------------------------
```

ExamplesWhile.java

Outline

1. Decision-making statements
   - if-else
   - switch
2. **Looping statements**
   - while
   - **do-while**
   - for
3. Branching statements
   - break
   - continue
   - System.exit

2. Looping statements
do-while

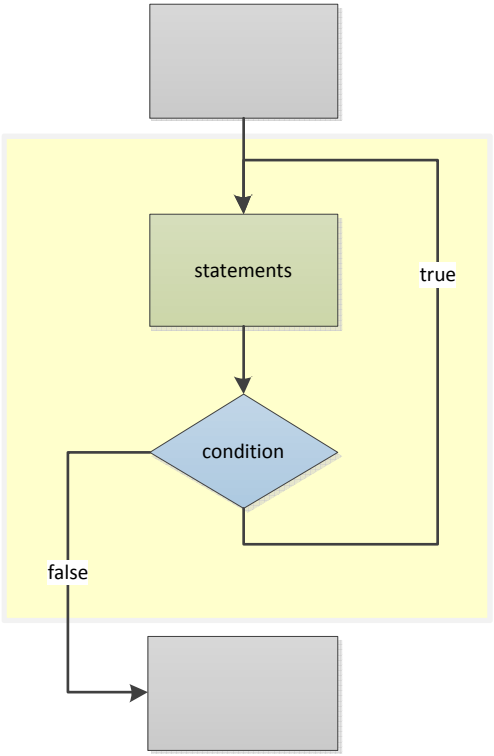
Executes a block of statements; repeat the execution if the condition is true

First, the block of code is executed

If the condition is true, the block of code associated to the **do while** is executed again

After finishing the **do while** block, the condition is tested again

If the condition is false, the execution continues by the first instruction below the block




```
do{
  <sentence(s)>
} while(<boolean expression>);
```

2. Looping statements

do-while

```
/* do while */
// Print even numbers between 1 and 100

System.out.println("-------------------------------------");
System.out.println("Even numbers example");
int n = 1;

do {

    if(n % 2 == 0) {    // n is even
        System.out.println(n + " is even.");
    }

    n = n + 1;          // increment n
} while(n <= 100);

System.out.println("End of example");
System.out.println("-------------------------------------");
```

```
Console
ExamplesDoWhile [Java Application]
----------------------------
Even numbers example
2 is even.
4 is even.
6 is even.
8 is even.
10 is even.
12 is even.
14 is even.
16 is even.
18 is even.
20 is even.
22 is even.
24 is even.
26 is even.
28 is even.
30 is even.
32 is even.
34 is even.
36 is even.
38 is even.
40 is even.
```

ExamplesDoWhile.java

2. Looping statements

do-while

```
// Add numbers read from the keyboard
System.out.println("---------------------------------------");
System.out.println("Read values while the user wants to continue");

String input = "";
int addition = 0;
Scanner sc = new Scanner(System.in);

do {
    System.out.print("value: ");
    int x = sc.nextInt();
    addition += x;

    System.out.print("continue (y/n)? ");
    input = sc.next();

} while(!input.equals("n"));

System.out.println("The sum of the values is: " + addition);

System.out.println("End of example");
System.out.println("---------------------------------------");
```

```
Console
<terminated> ExamplesDoWhile [Java Application] /System/
---------------------------------------
Read values while the user wants to continue
value: 2
continue (y/n)? y
value: 5
continue (y/n)? y
value: 3
continue (y/n)? n
The sum of the values is: 10
End of example
---------------------------------------
```

ExamplesDoWhile.java

2. Looping statements
while and do-while

```
/* do while and while */
// Count down
System.out.println("---------------------------------------");
int c;

// with do while
System.out.println("Count down with 'do while'");
c = 10;
do {
    System.out.println(c);
    c = c - 1;
} while (c < 10 && c > 0);

// with while
System.out.println("Count down with 'while'");
c = 10;
while (c < 10 && c > 0) {
    System.out.println(c);
    c = c - 1;
}

System.out.println("End of example");
System.out.println("---------------------------------------");
```

```
Console
<terminated> ControlFlow [Java Application] C:\Program Files\Java'
---------------------------------------
Count down with 'do while'
10
9
8
7
6
5
4
3
2
1
Count down with 'while'
End of example
---------------------------------------
```

ExamplesWhileDoWhile.java

Outline

1. Decision-making statements
   - if-else
   - switch
2. **Looping statements**
   - while
   - do-while
   - **for**
3. Branching statements
   - break
   - continue
   - System.exit

2. Looping statements
for

Executes a block of statements; repeat the execution if the condition is true (similar to **`while`**)

Additionally, performs more operations

- Pre-block statement (optional)
  - Usually, a variable declaration
- After-block statement (optional)
  - Usually, a variable increment/decrement

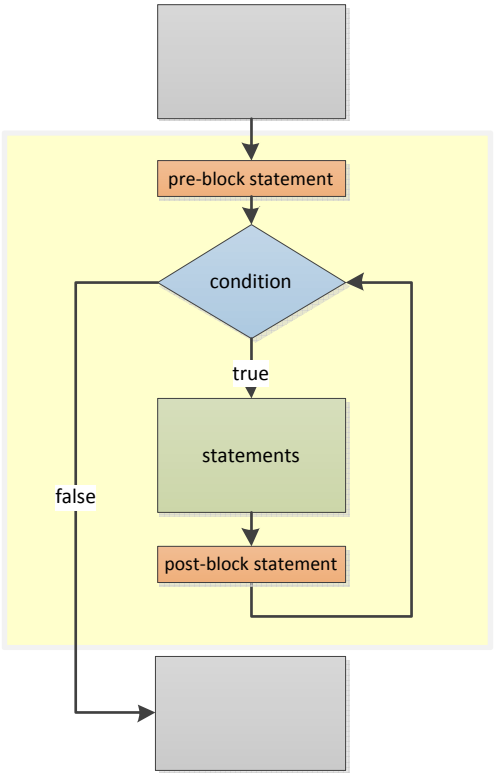
The first time, the *pre-block* statement is executed

If the condition is true, the associated block of code

After finishing the block, the *after-block* statement is executed

If the condition is true, the block is executed again

If the condition is false, the execution continues by the next instruction below the **`for`**




```
for ([pre-block]; <expression>; [post-block]){
  <statement(s)>
}
```

2. Looping statements
for

```
/* for */
// Print multiplication table
int t = 7;

System.out.println("Multiplication table: " + t);
for(int j = 0; j <= 10; j++) {
    System.out.println(t + " x " + j + " = " + t * j);
}
```

```
Console
<terminated> ControlFlow [Java Applica
Multiplication table: 7
7 x 0 = 0
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

pre-block and post-block statements

The *pre-block* statement is usually a variable definition, whereas the post-block statement is usually a modification of the *pre-block* variable

```
System.out.println("Multiplication table (while): " + t);
int k = 0;
while( k<=10 ) {
    System.out.println(t + " x " + k + " = " + t * k);
    k++;
}
```

`for` / `while` equivalence

`for` loops can be implemented with `while` loops, and vice versa

ExamplesFor.java

Loop instructions can be nested –with the following observations:

- The inner loop must be included **inside** the outer loop
- For each value of the counter of the outer instruction, the counter of the inner instruction takes all its values
  - Outer loop: i = {0, …, n-1}
    - Inner loop: j = {0, …, m-1}
      - **n x m pairs (i, j) inside the loop**

2. Looping statements
for

```
// Store and print multiplication tables
System.out.println("-------------------------------------");
System.out.println("All the multiplication tables");

int [][] tables = new int[11][11];

for(int i=0; i<=10; i++) {
    System.out.println("\nMultiplication table: " + i);

    for(int j=0; j<=10; j++) {
        tables[i][j] = i * j;
        System.out.println(i + " x " + j + " = " + tables[i][j]);
    }
}
System.out.println("End of example");
System.out.println("-------------------------------------");
```

```
Console
<terminated> ControlFlow [Java Application] C:\Program Files\J
-------------------------------------
All the multiplication tables

Multiplication table: 0
0 x 0 = 0
0 x 1 = 0
0 x 2 = 0
0 x 3 = 0
0 x 4 = 0
0 x 5 = 0
0 x 6 = 0
0 x 7 = 0
0 x 8 = 0
0 x 9 = 0
0 x 10 = 0

Multiplication table: 1
1 x 0 = 0
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication table: 2
```

`for` and arrays

`for` loops are frequently used to traverse all the elements of an array: initialization, operations with elements, etc.

ExamplesFor.java

Outline

1. Decision-making statements
   - if-else
   - switch
2. Looping statements
   - while
   - do-while
   - for
3. **Branching statements**
   - **break**
   - **continue**
   - **System.exit**

`break` terminates the execution of the loop

After the **`break`**, the execution continues in the statement just below the loop

```
break;
```

`continue` jumps to the next iteration of the loop

After the **`continue`**, the execution continues just before the evaluation of the condition of the loop

```
continue;
```

`System.exit(-1)` terminates the execution of the program

`System.exit` is used to finish the program when a wrong condition due to the parameters or the input values is detected

```
System.exit(-1);
```

3. Branching statements

break, continue, System.exit

```
/* break and continue */
System.out.println("---------------------------------------");
System.out.println("Break");

for(int i=0; i<=10; i++) {
    if(i== 5)
        break;      // Breaking the for loop
    System.out.println(i);
}

System.out.println("\nContinue");
for(int i=0; i<=10; i++) {
    if(i==5)
        continue;   // Continuing the for loop
    System.out.println(i);
}
System.out.println("End of example");
System.out.println("---------------------------------------");
```

```
Console
<terminated> ControlFlow [Java Application] C:\Program Files\Java\j
---------------------------------------
Break
0
1
2
3
4

Continue
0
1
2
3
4
6
7
8
9
10
End of example
---------------------------------------
```

ExamplesBranching.java

Outline

1. Decision-making statements
   - if-else
   - switch
2. Looping statements
   - while
   - do-while
   - for
3. Branching statements
   - break
   - continue
   - System.exit

Summary
Control flow statements

Conditional instructions

`if-else`
- A block of code is executed depending on a condition

`switch`
- A block of code is executed depending on the value of a single variable
- Cases have a special behavior

Loop instructions

`while`
- A block of code is repeated depending on a condition

`do-while`
- A block of code is repeated depending on a condition
- The block is executed at least once

`for`
- A block of code is repeated depending on a condition
- Additional statements are executed the first time the `for` is reached (pre-statement) and each time the `for` block is finished (post-statement)

Branching instructions

`break`
- The loop is finished; execution continues below the block

`continue`
- The loop is restarted; the condition is evaluated again

`System.exit`
- The program is finished

Recommended lectures

The Java™ Tutorials. Oracle, **Control flow statements** [link]

H. M. Deitel, P. J. Deitel. *Java: How to Program. Prentice Hall, 2007 (7th Edition)*, **Chapters 4** [link]**, 5** [link]**,**

K. Sierra, B. Bates. *Head First Java.* O'Reilly Media, 2005 (2nd Edition), **Chapter 5** [link]

B. Eckel. *Thinking in Java*. Prentice Hall, 2002 (3rd Edition), **Chapter 3** [link]

I. Horton. *Beginning Java 2, JDK 5 Edition*. Wrox, 2004 (5th Edition), **Chapter 3** [link]

Programming – Grado en Ingeniería Informática	
Authors	
Of this version: Juan Gómez Romero *Based on the work by:* Ángel García Olaya Manuel Pereira González Silvia de Castro García Gustavo Fernández-Baillo Cañas	 Universidad Carlos III de Madrid