Control Structures: If & Switch

CCS1110
Programming
Principles and Algorithms

Dr Ioanna Stamatopoulou

Control Structures - Outline

- Decision making
- if statement
- switch case
- Sequential vs selective execution

The `if` statement

The `if` statement: An example

- The following is an example of an **if statement**:

```
if ( grade > 39 ) {
    System.out.println(grade + "!!! Congratulations!");
}
```

- This means that the command:

```
System.out.println(grade + "!!! Congratulations!");
```

inside the **body** of the if statement is executed **only if the condition** `grade > 39` is **true**

The `if` statement: An example (complete)

```
/**
*   Filename: AnnounceResult.java
*   The program prompts the user to enter a grade and, if the grade is
*   greater than 39, the program prints the grade together with a
*   congratulations message.
*/
public class AnnounceResult
{
   public static void main (String[] args)
   {
      Scanner scanner = new Scanner( System.in );
      System.out.print("Please enter your grade: " );
      int grade = scanner.nextInt();

      if (grade > 39) {
         System.out.println(grade + "!!! Congratulations!");
      }

   } // end of main method
} // end of class
```


Conditions: `if-then-else` An Example

```
int positiveTotal=0, negativeTotal=0;
int currNumber;
// let's assume variable grade gets a value from the user

if (currNumber >= 0)
   positiveTotal += currNumber;
else
   negativeTotal += currNumber;
```

`if` statement: Syntax

Single selection:

```
if (condition) {
    statement_1 ;
    ...
    statement_n ;
}
```

Two alternative (mutual exclusive) cases:

```
if (condition) {
    statement_1 ;
    ...
}
else {
    statement_1 ;
    ...
}
```

Multiple alternatives:

```
if (condition_1) {
    statement_1 ;
    ...
}
else if (condition_2) {
    statement_1 ;
    ...
}
else if (condition_3) {
    statement_1 ;
    ...
}
...
else {
    statement_1 ;
    ...
}
```


Boolean expressions

What is a "condition"?

- A condition is a boolean expression:
  - A statement that can be evaluated toeither being **true** or **false**
- In the following example, depending on the value of the variable grade, the statement grade == 100 is either true or false

```
if ( grade == 100 ) {
    System.out.println("Excellent!");
}
```


Forming boolean expressions: Equality & Relational operators

- Equality operators
  - a is equal to b — `a == b`
  - a is not equal to b — `a != b`
- Relational operators
  - a is greater than b — `a > b`
  - a is greater than or equal to b — `a >= b`
  - a is less than b — `a < b`
  - a is less than or equal to b — `a <= b`

Common Mistake: Don't use the assignment operator to check for equality!

!

```
if ( grade = 100 ) {
    System.out.println("Excellent! ");
}
```

Operator Precedence

- Priority during execution:

  Assignment operator
  > Equality operators
  > Relational operators

Building more complex boolean expressions

- To build more complex boolean expressions we need boolean operators
- Boolean operators require boolean operands

```
int a = 10;
int b = 20;

if ( a >= 100 && b > 10 || a > b ) {
    System.out.println("This message will be printed only if the above complex
                                                boolean expression is true!");
}
```

Boolean Operators

- **AND** operator **&&**
  - BOTH operands are true
- **OR** operator **||**
  - AT LEAST ONE OF THE TWO operands is true
- **NOT** operator **!**
  - Requires only one operand
  - Negates the value of the operand
  - (turns true to false and vice-versa)

The `switch` statement

The `switch` statement

- The switch statement may be used instead of the if statement in situations where **the conditions of all alternative cases depend on the value of one variable**

The `switch` statement: General Syntax

```
switch (variable) {
   case variable-value-1:
      statement;
      break;
   case variable-value-2:
      statement;
      break;
   ...
   default:
      statement;
      break;
}
```

The `switch` statement: An example

```
char grade;
...//grade gets a value
String message;
switch (grade)
{
   case 'A': message = "Very Good";
             break;
   case 'B': message = "Try more";
             break;
   case 'C': message = "Oops";
             break;
   default:  message = "No grade?";
             break;
}
```


Mind the break!

- Execution starts
  - After the case statement that is true
- Execution ends
  - At the next break, not the next case

```
char grade;
...//grade gets a value
int passes = 0, honours = 0, fails = 0;
switch (grade) {
   case 'A': honours++;
   case 'B':
   case 'C': passes++;
             break;
   default:  fails++;
             break;
}
```

Formatting conventions

- Leave an empty line before and after a control structure (if, switch)
- Leave a space before and after an operator (assignment, equality, relational, boolean, etc.)