Computer Programming I - CSC111

Chapter 3 – Flow of Control

Dr. Mejdl Safran

mejdl@ksu.edu.sa

Outline

- Basic `if-else` statement
- Boolean expressions
- Nested `if-else` statements
- Multibranch `if-else` statements
- Case Study – Body Mass Index
- Case study – the `exit` method
- The `switch` statement

Flow of control

- *Flow of control* is the order in which a program performs actions.
  - Up to this point, the order has been sequential.
- A *branching statement* chooses between two or more possible actions.
- A *loop statement* repeats an action until a stopping condition occurs.

The `if-else` statement

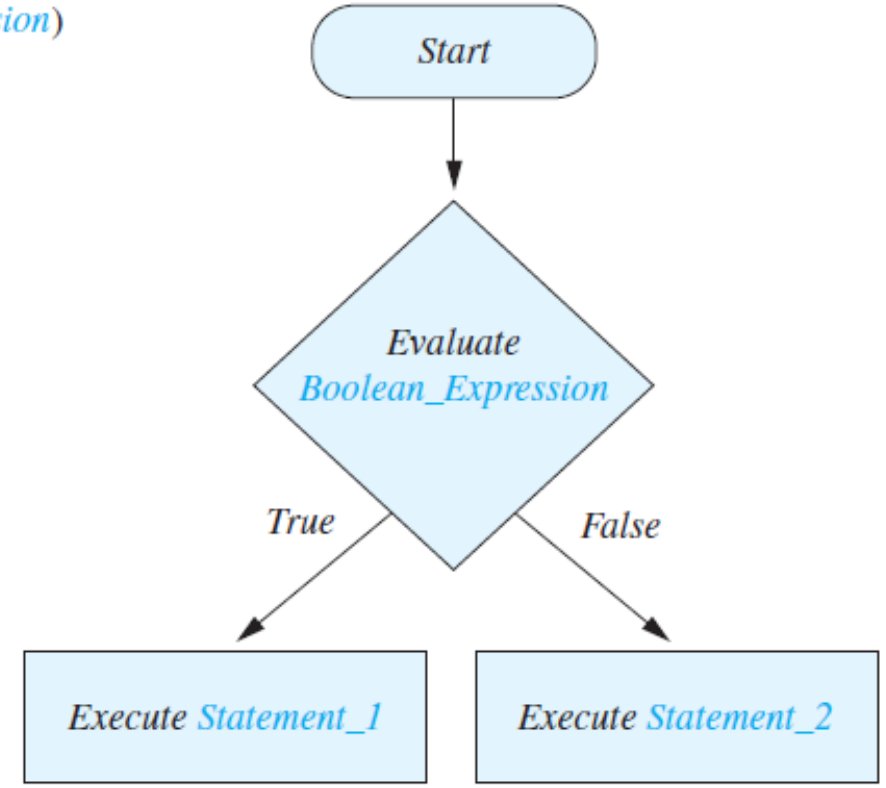
- A *branching statement* that chooses between two possible actions.

- Syntax

```
if (Boolean_Expression)
   Statement_1
else
   Statement_2
```

Semantics of the `if-else` statement

```
if (Boolean_Expression)
    Statement_1
else
    Statement_2
```



The `if-else` statement: Example

```
double score;
Scanner keyboard = new Scanner(System.in);

System.out.println("Please enter your score: ");
score = keyboard.nextDouble();

if(score >= 60)
    System.out.println("You have passed!");
else
    System.out.println("Sorry, you have failed, try the course again");
```

The `if-else` statement: Example

Output

```
Please enter your score:
95
You have passed!


Please enter your score:
58
Sorry, you have failed, try the course
again
```

The `if-else` statement: String example

```
String s1, s2;
System.out.println("Enter two lines of text:");
Scanner keyboard = new Scanner(System.in);
s1 = keyboard.nextLine();
s2 = keyboard.nextLine();

if (s1.equals(s2))
   System.out.println("The two lines are equal.");
else
   System.out.println("The two lines are not equal.");

if (s2.equals(s1))
   System.out.println("The two lines are equal.");
else
   System.out.println("The two lines are not equal.");

if (s1.equalsIgnoreCase(s2))
   System.out.println(
               "But the lines are equal, ignoring case.");
else
   System.out.println(
               "Lines are not equal, even ignoring case.");
```

These two invocations of the method `equals` *are equivalent.*

The `if-else` statement: String example

Sample Screen Output

```
Enter two lines of text:
Java is not coffee.
Java is NOT COFFEE.
The two lines are not equal.
The two lines are not equal.
But the lines are equal, ignoring case.
```

Compound statements

- To include multiple statements in a branch, enclose the statements in braces.
- Syntax

```
{
      Statement_1;
      Statement_2;
    …
}
```

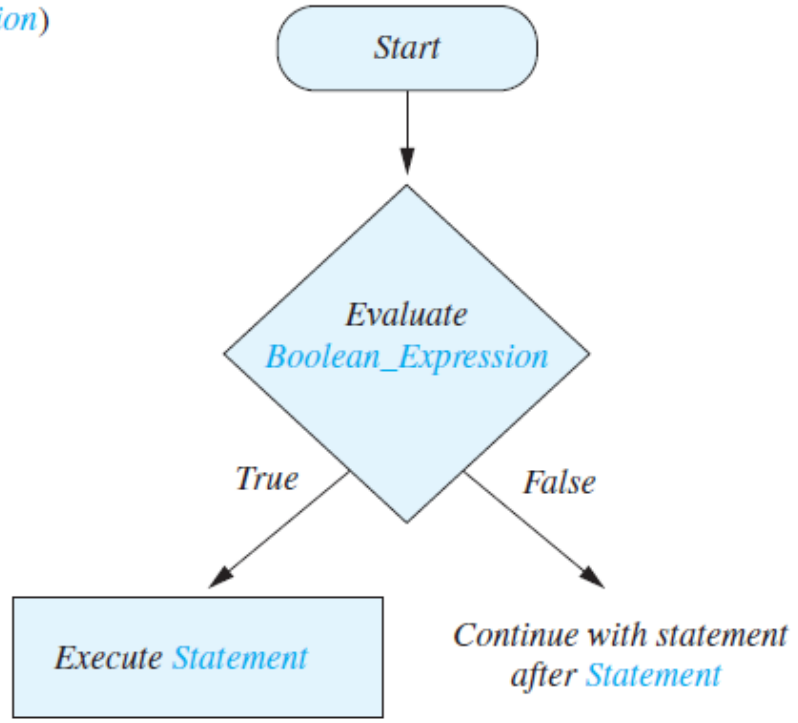
- Example:

```
if (count < 3)
{
    total = 0;
    count = 0;
}
```

Omitting the `else` part

- The semantics of an `if` statement without an `else`

```
if (Boolean_Expression)
    Statement
```



Boolean expressions

- The value of a *boolean expression* is either `true` or `false`.
- Examples

```
time < limit
balance <= 0
```

Java Comparison Operators

Math Notation	Name	Java Notation	Java Examples
=	Equal to	==	`balance == 0` `answer == 'y'`
≠	Not equal to	!=	`income != tax` `answer != 'y'`
>	Greater than	>	`expenses > income`
≥	Greater than or equal to	>=	`points >= 60`
<	Less than	<	`pressure < max`
≤	Less than or equal to	<=	`expenses <= income`

Compound Boolean Expressions (&&)

- Boolean expressions can be combined using the "and" `(&&)` operator.
- Example

```
if ((score > 0) && (score <= 100))
...
```

- Not allowed

```
if (0 < score <= 100)
...
```

Compound Boolean Expressions (&&)

- Syntax

```
(Sub_Expression_1) &&
 (Sub_Expression_2)
```

- Parentheses often are used to enhance readability.
- The larger expression is true only when both of the smaller expressions are true.

Compound Boolean Expressions (||)

- Boolean expressions can be combined using the "or" `(||)` operator.
- Example

```
if ((quantity > 5) || (cost < 10))
...
```

- Syntax

```
(Sub_Expression_1) || (Sub_Expression_2)
```

Compound Boolean Expressions (||)

- The larger expression is true
  - When either of the smaller expressions is true
  - When both of the smaller expressions are true.
- The Java version of "or" is the *inclusive or* which allows either or both to be true.
- The *exclusive or* allows one or the other, but not both to be true.

Negating a Boolean Expression (!)

- A boolean expression can be negated using the "not" (**!**) operator.
- Syntax

  ```
  !(Boolean_Expression)
  ```

- Example

  ```
  (a || b) && !(a && b)
  ```

  which is the *exclusive or*

Negating a Boolean Expression (!)

- Avoiding the negation operator

! (*A Op B*) Is Equivalent to (*A Op B*)	
<	>=
<=	>
>	<=
>=	<
==	!=
!=	==

Negating a Boolean Expression (!)

EXAMPLE

```
if (!(number < 0))
   System.out.println("OK");
else
    System.out.println("Negative!");
```

Java logical operators

Name	Java Notation	Java Examples
Logical *and*	&&	`(sum > min) && (sum < max)`
Logical *or*	||	`(answer == 'y') || (answer == 'Y')`
Logical *not*	!	`!(number < 0)`

Boolean operators

Value of *A*	Value of *B*	Value of *A* && *B*	Value of *A* || *B*	Value of ! (*A*)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Nested `if-else` statements

- An `if-else` statement can contain any sort of statement within it.
- In particular, it can contain another `if-else` statement.
  - An `if-else` may be nested within the "if" part.
  - An `if-else` may be nested within the "else" part.
  - An `if-else` may be nested within both parts.

Nested statements

- Syntax

```
if (Boolean_Expression_1)
    if (Boolean_Expression_2)
        Statement_1;
    else
        Statement_2;
else
    if (Boolean_Expression_3)
        Statement_3;
    else
        Statement_4;
```

Nested statements

- Each `else` is paired with the nearest unmatched `if`.
- **If used properly**, indentation communicates which `if` goes with which `else`.
- Braces can be used like parentheses to group statements.

Nested statements

- Subtly different forms

First Form

```
if (a > b)
{
     if (c > d)
          e = f
}
     else
          g = h;
```

Second Form

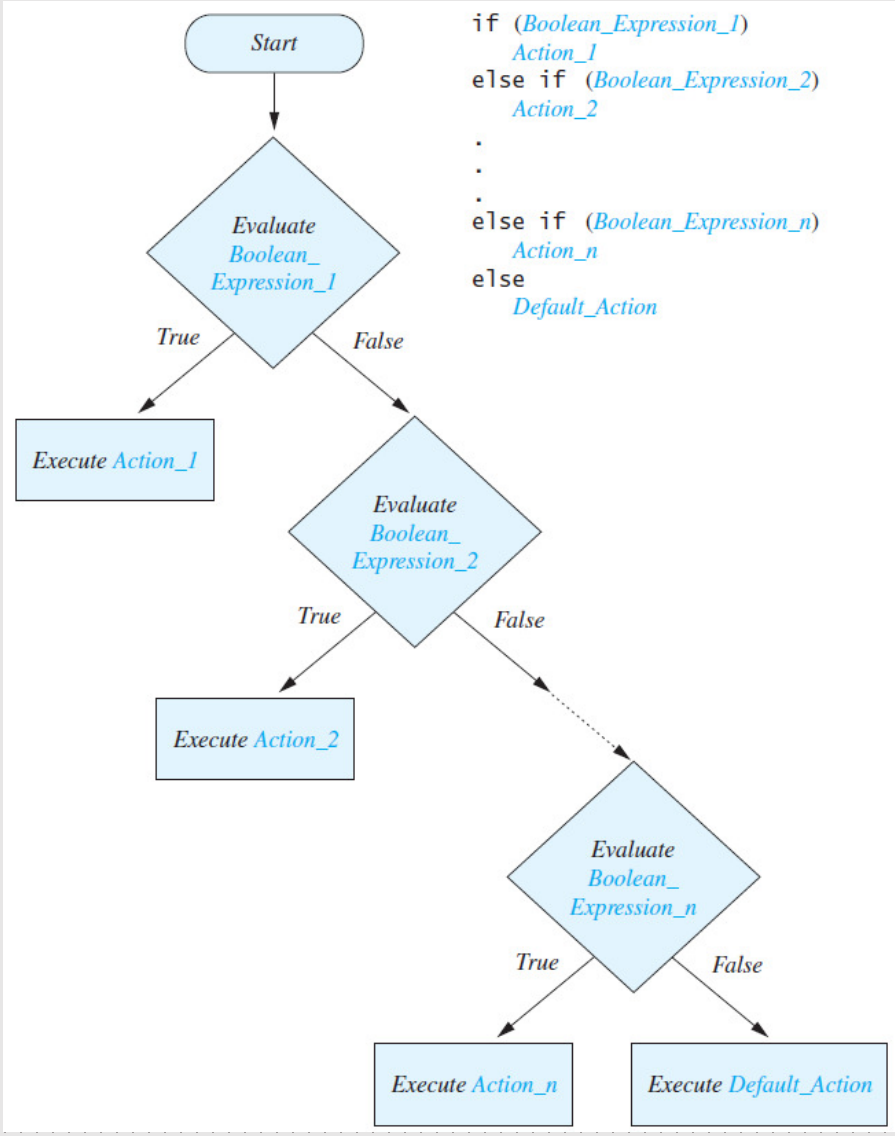
```
if (a > b)
    if (c > d)
         e = f
    else
         g = h;

// oops
```

Multibranch `if-else` statements

- Syntax

```
if (Boolean_Expression_1)
    Statement_1;
else if (Boolean_Expression_2)
    Statement_2;
else if (Boolean_Expression_3)
    Statement_3;
else if …
else
    Default_Statement;
```

Multibranch `if-else` statements



Multibranch `if-else` statements: example

```
if (balance > 0)
   System.out.println("Positive balance");
else if (balance < 0)
   System.out.println("Negative balance");
else if (balance == 0)
   System.out.println("Zero balance");
```

```
if (balance > 0)
   System.out.println("Positive balance");
else if (balance < 0)
   System.out.println("Negative balance");
else
   System.out.println("Zero balance");
```

Assigning letter grader example

```
import java.util.Scanner;
public class Grader
{
    public static void main(String[] args)
    {
        int score;
        char grade;

        System.out.println("Enter your score: ");
        Scanner keyboard = new Scanner(System.in);
        score = keyboard.nextInt();

        if (score >= 90)
                grade = 'A';
        else if ((score >= 80) && (score < 90))
                grade = 'B';
        else if ((score >= 70) && (score < 80))
                grade = 'C';
        else if ((score >= 60) && (score < 70))
                grade = 'D';
        else
                grade = 'F';

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

Assigning letter grader example : equivalent code

```
import java.util.Scanner;
public class Grader
{
    public static void main(String[] args)
    {
        int score;
        char grade;

        System.out.println("Enter your score: ");
        Scanner keyboard = new Scanner(System.in);
        score = keyboard.nextInt();

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

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

Case Study – Body Mass Index

- Body Mass Index (BMI) is used to estimate the risk of weight-related problems
- BMI = mass / height^2
  - Mass in kilograms, height in meters
- Health assessment if:
  - BMI < 18.5 Underweight
  - 18.5 ≤ BMI < 25 Normal weight
  - 25 ≤ BMI < 30 Overweight
  - 30 ≤ BMI Obese

Case Study – Body Mass Index

- Algorithm
  - Input height in feet & inches, weight in pounds
  - Convert to meters and kilograms
    - 1 feet = 12 inches
    - 1 inch = 0.254 meters
      - ((feet * 12) + inches) * 0.254
    - 2.2 lb = 1 kg
      - pound / 2.2
  - Compute BMI
  - Output health risk using if statements

```java
import java.util.Scanner;
public class BMI
{
    public static void main(String[] args)
    {
        Scanner keyboard = new Scanner(System.in);
        int pounds, feet, inches;
        double heightMeters, mass, BMI;
        System.out.println("Enter your weight in pounds.");
        pounds = keyboard.nextInt();
        System.out.println("Enter your height in feet" +
        "followed by a space" +
        "then additional inches.");
        feet = keyboard.nextInt();
        inches = keyboard.nextInt();
        heightMeters = ((feet * 12) + inches) * 0.0254;
        mass = (pounds / 2.2);
        BMI = mass / (heightMeters * heightMeters);
        System.out.println("Your BMI is " + BMI);
        System.out.print("Your risk category is ");
        if (BMI < 18.5)
                System.out.println("Underweight.");
        else if (BMI < 25)
                System.out.println("Normal weight.");
        else if (BMI < 30)
                System.out.println("Overweight.");
        else
                System.out.println("Obese.");
    }
}
```

Sample Screen Output

```
Enter your weight in pounds.
150
Enter your height in feet followed
by a space then additional inches.
5 5
Your BMI is 25.013498117367398
Your risk category is Overweight.
```

Challenge

```
int x = 1;
int y = 2;
int z=y=x;
if(++x < x++)
      x=y;
else if( (x=y) <= ++z)
      x*=2;
x*= y*z/2;
System.out.println("x= " + x);
```

The `exit` Method

- Sometimes a situation arises that makes continuing the program pointless.
- A program can be terminated normally by `System.exit(0)`.

The `exit` Method

- Example

```
if (numberOfWinners == 0)
{
    System.out.println ("Error: Dividing by zero.");
    System.exit(0);
}
else
{
    oneShare = payoff / numberOfWinners;
    System.out.println("Each winner will receive $" +
  oneShare);
}
```

The type `boolean`

- The type `boolean` is a primitive type with only two values: `true` and `false`.
- Boolean variables can make programs more readable.

```
if (systemsAreOK)
```

instead of

```
if((temperature <= 100) && (thrust >=
 12000) && (cabinPressure > 30) && …)
```

Boolean expressions and variables

- Variables, constants, and expressions of type `boolean` all evaluate to either `true` or `false`.
- A boolean variable can be given the value of a boolean expression by using an assignment operator.

```
boolean isPositive = (number > 0);
...
if (isPositive) ...
```

Naming boolean variables

- Choose names such as `isPositive` or `systemsAreOk`.
- Avoid names such as `numberSign` or `systemStatus`.

What output is produced by this code?

```
int number = 7;
boolean isPositive = (number > 0);
if (number > 0);
number = -100;
if (isPositive)
      System.out.println("Positive.");
else
      System.out.println("Not positive.");
      System.out.println(number);
```

Output

```
Positive.
-100
```

Precedence rules

- In what order are the operations performed?

```
score < min/2 - 10 || score > 90
score < (min/2) - 10 || score > 90
score < ((min/2) - 10) || score > 90
(score < ((min/2) - 10)) || score > 90
(score < ((min/2) - 10)) || (score > 90)
```

Short-circuit evaluation

- Sometimes only part of a boolean expression needs to be evaluated to determine the value of the entire expression.
  - If the first operand associated with an `||` is `true`, the expression is `true`.
  - If the first operand associated with an `&&` is `false`, the expression is `false`.
- This is called *short-circuit* or *lazy* evaluation.

Short-circuit evaluation

- Short-circuit evaluation is not only efficient, sometimes it is essential!
- A run-time error can result, for example, from an attempt to divide by zero.

```
if ((number != 0) && (sum/number > 5))
```

- *Complete evaluation* can be achieved by substituting `&` for `&&` or `|` for `||`.

Example

```
int x= 10, y=20;

if(x>15 && y++<50)
      x=0;
System.out.println("x= " + x + " ; y= " + y);

if(x>15 & y++<50)
      x=0;
System.out.println("x= " + x + " ; y= " + y);
```

Output

```
x= 10 ; y= 20
x= 10 ; y= 21
```

Input and output of boolean values

```
boolean booleanVar = false;
System.out.println(booleanVar);
System.out.println("Enter a boolean value:");
Scanner keyboard = new Scanner(System.in);
booleanVar = keyboard.nextBoolean();
System.out.println("You entered " + booleanVar);
```

Output

```
false
Enter a boolean value:
true
You entered true
```

Input validation

- You should check your input to ensure that it is within a valid or reasonable range.
- Example:

```
int timeInHours;
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter the time in hours (24-hour clock): ");
timeInHours = keyboard.nextInt();
if(timeInHours >= 0 && timeInHours < 24)
System.out.println("Time is " + timeInHours);
else
System.out.println("Your input is out of range!");
```

Challenge

```
int timeInHours;
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter the time in hours (24-hour clock): ");
timeInHours = keyboard.nextInt();
if(timeInHours >= 0 && timeInHours < 24)
System.out.println("Time is " + timeInHours);
else
System.out.println("Your input is out of range!");
```

- Add check for minutes
- Try to convert and output the time to the 12-hour clock

The `switch` statement

- The `switch` statement is a mutltiway branch that makes a decision based on an *integral* (integer or character) expression.
  - Java 7 allows String expressions
- The `switch` statement begins with the keyword `switch` followed by an integral expression in parentheses and called the *controlling expression.*

The `switch` statement

- A list of cases follows, enclosed in braces.
- Each case consists of the keyword `case` followed by
  - A constant called the *case label*
  - A colon
  - A list of statements.
- The list is searched for a case label matching the controlling expression.

The `switch` statement

- The action associated with a matching case label is executed.
- If no match is found, the case labeled `default` is executed.
  - The `default` case is optional, but recommended, even if it simply prints a message.
- Repeated case labels are not allowed.

The `switch` statement

- Syntax

```
switch(Controlling_Expression)
{
    case Case_Label:
        Statement(s);
        break;
    case Case_Label:
    …
    default:
    …
}
```

The `switch` statement

- The action for each case typically ends with the word `break`.
- The optional `break` statement prevents the consideration of other cases.
- The controlling expression can be anything that evaluates to an integral type.

```
int day;
Scanner input = new Scanner(System.in);
System.out.println("Enter the number of the day: ");
day = input.nextInt();
switch(day)
{
     case 1:
        System.out.println("Sunday");
        break;
     case 2:
        System.out.println("Monday");
        break;
    case 3:
        System.out.println("Tuesday");
        break;
    case 4:
        System.out.println("Wednesday");
        break;
    case 5:
        System.out.println("Thursday");
        break;
    case 6:
        System.out.println("Friday");
        break;
    case 7:
        System.out.println("Saturday");
        break;
    default:
        System.out.println("Wrong input!! Your input should be between 1-7");
        break;
}
```

Sample Output

```
Enter the number of the day:
1
Sunday
```

Sample Output

```
Enter the number of the day:
9
Wrong input!! Your input should be between 1-7
```

Another example

- Write a program using switch statement that prints the appropriate message according to your entered grade.

Grade to enter	Message to print
A	Excellent grade
B	Very good grade
C	Good grade
D , E or F	Low grade
Other grades	Invalid grade

```java
char grade;
Scanner input = new Scanner(System.in);
System.out.println("Enter your grade (A,B,C,D,E or F): ");
grade = input.nextLine().charAt(0);

switch (grade) {
    case 'A':
        System.out.println("Excellent grade");
        break;
    case 'B':
        System.out.println("Very good grade");
        break;
    case 'C':
        System.out.println("Good grade");
        break;
    case 'D':
    case 'E':
    case 'F':
        System.out.println("Low grade");
        break;
    default:
        System.out.println("Invalid grade");
        break;
}
```