CIS2571 – Intro to Java

Chapter 3 → Selections

choose an action with two or more alternatives

Information consolidated from a variety of textbook and online resources

Topic Objectives

- Know how to use the relational operators
- Understand the Boolean data type
- Practice the use of
  - One-way if statements
  - Two-way if-else statements
  - Nested if statements
  - Switch statements
- Understand how the conditional operator is similar to a two-way if-else statement
- Know how to use logical operators
- Know the rules of operator precedence and associativity
- Understand format specifiers
- Know how to use confirmation dialog boxes
- Know the use of some Java static class methods

Comparison or Relational Operators

- binary operators compare operands to yield one of two results: true or false

  < less than

  <= less than or equal to

  = = equal to

  > greater than

  >= greater than or equal to

  != not equal to

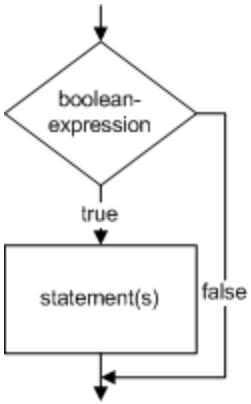
- Boolean variable holds boolean data type

```
boolean lightsOn = true;
```

  - **true** and **false** are literal boolean constants

if Statements

```
if (boolean-expression) {
    statements;
}
```



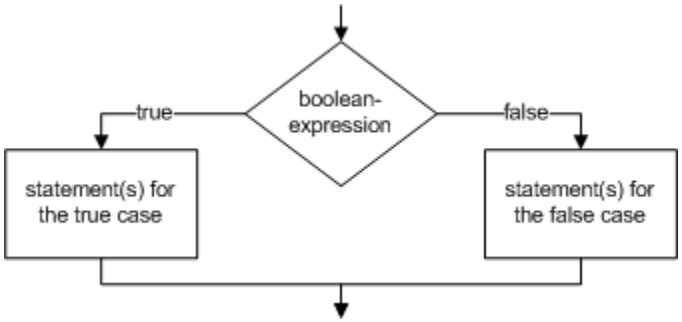
- **one-way if statement** executes an action if the boolean-condition is **true**
- block braces can be omitted if they enclose a single statement

```
if (radius >= 0) {
    System.out.println("The radius is positive.");
}
```

```
if (radius >= 0)
    System.out.println("The radius is positive.");
```

if Statements

```
if (boolean-expression) {
    statement(s)-for-the-true-case;
}
else {
    statements(s)-for-the-false-case;
}
```



- **two-way if-else statement** executes statements
  - true case if boolean-condition evaluates to **true**
  - false case if boolean-condition evaluates to **false**
- block braces can be omitted if they enclose a single statement

```
if (radius >= 0) {
    area = radius * radius * PI;
    System.out.println("The area for the circle of radius " +
        radius + " is " + area);
}
else
    System.out.println("Negative input");
```

Common Errors

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " +
    radius + " is " + area);
}
```

- Forgetting necessary braces
- Wrong semicolon at the **if** line

```
if (radius >= 0)
{
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " +
    radius + " is " + area);
}
```

Common Errors

```
if (even)
  System.out.println("It is even");
```

- Redundant testing of **boolean** values
- Dangling **else** ambiguity
  - Always matches most recent unmatched **if** clause in same block
  - Add braces to group if statement with no else

```
if (i > j)
  if (i > k)
    System.out.println("i > j AND i > k");
else
  System.out.println("i <= j");
```

```
if (i > j) {
  if (i > k)
    System.out.println("i > j AND i > k");
  }
else
  System.out.println("i <= j");
```

Conditional Operator

- Uses ? and :
  - Ternary (three operands) operator
- Conditional expression evaluates to
  - expression1 if boolean-expression is **true**
  - expression2 if boolean-expression is **false**

```
boolean-expression ? expression1 : expression2;
```

```
max = (num1 > num2) ? num1 : num2;
```

```
boolean isPos = (radius >= 0) ? true : false;
```

Logical Operators

&&	true	false
true	true	false
false	false	false

- Also known as Boolean operators
- Operates on Boolean variables to create new Boolean value
- && logical AND (binary operator)
  - evaluates **true** only when both operands are **true**
  - short-circuits to **false** if first operand is **false**

op precedence (App C)
%
== !=
&&
=

```
// leap year is divisible by 4 but not 100
boolean isLeapYear = year % 4 == 0 && year % 100 != 0;
```

year	year % 4	year % 100	isLeapYear
2016	0	16	T && T -> T
2015	3	15	F && T -> F
2014	2	14	F && T -> F
...			
2000	0	0	T && F -> F?

Logical Operators

||	true	false
true	true	true
false	true	false

- || logical OR (binary operator)
  - evaluates **true** when either (or both) operand(s) is **true**
  - short-circuits to **true** if first operand is **true**

```
// leap year is divisible by 4 but not 100, or divisible by 400
boolean isLeapYear = (year % 4 == 0 && year % 100 != 0) ||
  (year % 400 == 0);
```

```
year     year % 4    year % 100    year % 400    isLeapYear
----     --------    ----------    ----------    ----------
2016     0           16            16            T || F
2015     3           15            15            F || F
2014     2           14            14            F || F
...
2000     0           0             0             F || T
```

Logical Operators

^	true	false
true	false	true
false	true	false

- ^ exclusive OR (binary operator)
  - only true if operands have **different boolean** values
  - evaluates **true** if one <u>or</u> the other operand is **true**, but <u>not</u> both

```
boolean inJava, inCPP;
// additional statements
if (inJava ^ inCPP) {
  System.out.println("Enrolled in Java or CPP, but not both!");
}
```

Logical Operators

	!
true	false
false	true

- ! logical NOT (unary operator)
  - forces operand to its opposite state

```
if (!(radius >= 0)){
  System.out.println("Invalid radius!");
}
OR
boolean isPos = (radius > 0) ? true : false;
if (!isPos) {
  System.out.println("Invalid radius!");
}
```

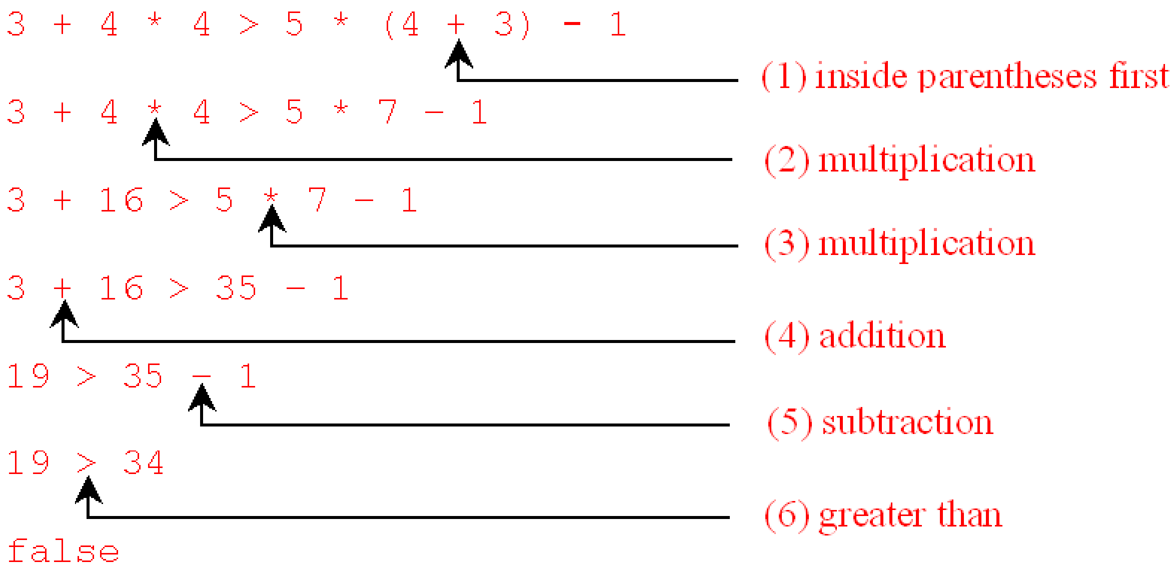
Operator Precedence and Associativity

- If operators with the same precedence are next to each other, their associativity determines the order of evaluation.
- All binary operators except assignment operators are left-associative.
- Operator Precedence Chart *(see Appendix C in textbook)*

```
var++, var--
+, - (Unary plus and minus), ++var,--var
(type) Casting
! (Not)
*, /, % (Multiplication, division, and remainder)
+, - (Binary addition and subtraction)
<, <=, >, >= (Comparison)
==, !=; (Equality)
^ (Exclusive OR)
&& (Conditional AND) Short-circuit AND
|| (Conditional OR) Short-circuit OR
=, +=, -=, *=, /=, %= (Assignment operator)
```

Example

- Applying the operator precedence and associativity rule, the expression 3 + 4 * 4 > 5 * (4 + 3) - 1 is evaluated as follows:



```
3 + 4 * 4 > 5 * (4 + 3) - 1
3 + 4 * 4 > 5 * 7 - 1
3 + 16 > 5 * 7 - 1
3 + 16 > 35 - 1
19 > 35 - 1
19 > 34
false
```

(1) inside parentheses first

(2) multiplication

(3) multiplication

(4) addition

(5) subtraction

(6) greater than

Nested if Statements

- If statements can be nested
  - No limit to the depth of nesting
- Nested if can be used to implement multiple alternatives
  - Equivalent multi-way if-else statements avoid deep indentation

```
if (score >= 90.0)
  grade = 'A';
else
  if (score >= 80.0)
    grade = 'B';
  else
    if (score >= 70.0)
      grade = 'C';
    else
      if (score >= 60.0)
        grade = 'D';
      else
        grade = 'F';
```

equivalent

```
if (score >= 90.0)
  grade = 'A';
else if (score >= 80.0)
  grade = 'B';
else if (score >= 70.0)
  grade = 'C';
else if (score >= 60.0)
  grade = 'D';
else
  grade = 'F';
```

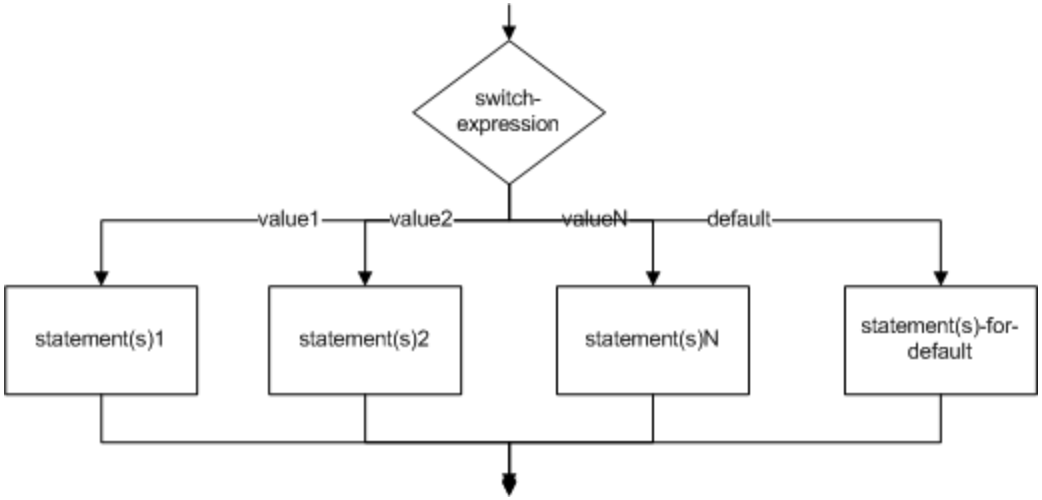
switch Statement

- Can be used to replace nested **if** statements
  - switch-expression must be **char**, **byte**, **short**, **int**, or **String** type enclosed in parentheses
    - According to the Java Language Specification**, "*…The type of the Expression must be char, byte, short, int, Character, Byte, Short, Integer, String, or an enum type (§8.9), or a compile-time error occurs.*"
  - value1, value2, … valueN must be constants that are the same data type as switch-expression
    - checked in sequential order
    - if match found, statements executed until **break** or end of switch occurs
  - **break** and **defaults** are optional

** http://docs.oracle.com/javase/specs/jls/se7/html/jls-14.html#jls-14.11

switch Statement Format

```
switch (switch-expression){
  case value1: statement(s)1;
    break;
  case value2: statement(s)2;
    break;
  ...
  case valueN: statement(s)N;
    break;
  default: statement(s)-for-default;
}
```



switch Statement Example

```
if (grade == 'A')
  System.out.println("Outstanding");
else if (grade == 'B')
  System.out.println("Nice Work");
else if (grade == 'C')
  System.out.println("You Passed");
else if (grade == 'D')
  System.out.println("Try Harder Next Time");
else
  System.out.println("OUCH");
```

```
switch(grade){
  case 'A': System.out.println("Outstanding");
    break;
  case 'B': System.out.println("Nice Work");
    break;
  case 'C': System.out.println("You Passed");
    break;
  case 'D': System.out.println("Try Harder Next Time");
    break;
  default: System.out.println("OUCH");
}
```

Formatting Console Output

- printf statement used to format output

```
System.out.printf(format, item1, item2, … itemk)
```

  - **format** is string that may contain substrings and format specifiers
    - %b → Boolean value
    - %c → character
    - %d → decimal integer
    - %f → floating point number
      - *width.precision* (default 6 digits)
    - %e → scientific notation
    - %s → string
  - Optional *width* specifier between % and conversion code
    - Minimum width; expands for larger values; pads with spaces for smaller values
    - Default right justified in given *width*
      - - to left justify
  - Items must match specifiers in order, number, and type

Examples

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);
```

items

```
display            count is 5 and amount is 45.560000
```

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %5d and amount is %7.2f", count, amount);

12345678901234567890123456789012345678 90
count is     5 and amount is   45.56
```

```
int count = 5;
double amount = 45.56;
System.out.printf("count is %-5d and amount is %-7.2f", count, amount);

12345678901234567890123456789012345678 90
count is 5     and amount is 45.56
```

Static Methods

- **Math.random()**
  - generates a pseudorandom **double** number
    - 0.0 <= Math.random() < 1.0
  - use type conversion, offset, and scaling to generate other random numbers
    - integer >= 0 and < 100
      - `(int)(Math.random() * 100)`
    - integer >= 1 and <= 6
      - `1 + (int)(Math.random() * 6)`

scale *(0.0 to 5.99)*

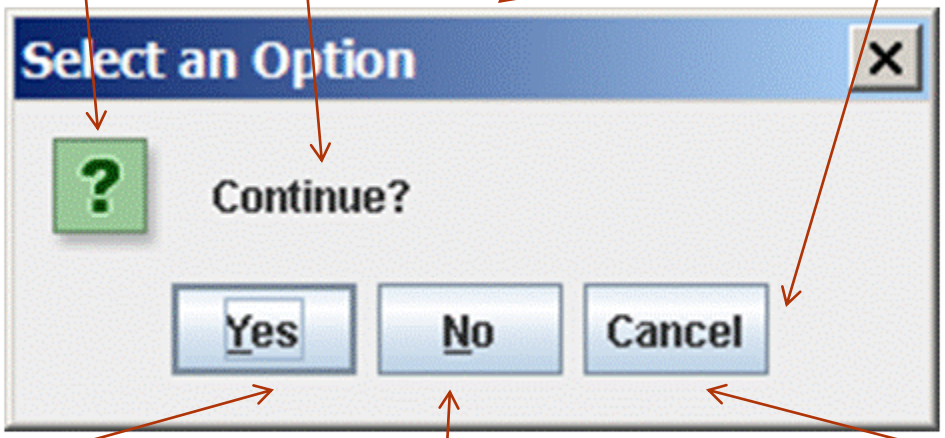
scale and convert *(0 to 5)*

scale, convert, and offset *(1 to 6)*

- **System.exit(n)**
  - terminates program and returns value n
    - n = 0 implies normal termination

GUI Confirmation Dialogs

```
int option = JOptionPane.showConfirmDialog
      (null, "Continue", "Select an Option",
       JOptionPane.YES_NO_CANCEL_OPTION,
       JOptionPane.QUESTION_MESSAGE);
```



JOptionPane.YES_OPTION

JOptionPane.CANCEL_OPTION

JOptionPane.NO_OPTION