Chp 5: Operators

- Operators
  - symbol that causes some operations to be performed on one or more variables (or data values: literals or constants).
  - Java Operators:
    - fundamental arithmetic operators
    - assignment operators
    - increment/decrement operators
    - arithmetic assignment operators
    - concatenation operators
    - relational operators
    - logical operators

(1)
- Fundamental Arithmetic Operators
  - unary operators:
    - change the sign, e.g., -10
  - binary operators: + - * / % (modulus)
    - 7 / 5 is 1 (integer division)
    - 7 % 5 is 2

(2)
- Assignment operator
  - syntax: variable name = expression
    e.g., a = b ;
  - evaluate the expression on right hand side and assign the value to the variable on left hand side

(3)
- Increment/decrement Operators on int type variables
  - ++ and --
  - In prefix mode: ++Variable
    e.g., a = ++b ;
    - Variable b is incremented by 1 **first**
    - Then, expression is evaluated
  - In postfix mode: Variable++
    e.g., a = b++ ;
    - Expression is evaluated **first**
    - Variable b is then incremented by 1
  - decrement operator -- for decrement by 1

(4)
- Arithmetic Assignment Operators
  += -= *= /= %=
  Note: a += b ; means a = a + b

- Expressions
  - must produce a value
  - e.g.,
    - A constant or literals
    - A variable
    - A combination of operators and operands (variables, constants, and literals)
    - may involve incrementing or decrementing
    - may also involve assignments inside
      num + (h = 5)
      - this expression has side effect
      - good programming style: avoid this
  - Variable += Expression:
- Multiple assignments can also be done in one statement:
  a = b = c = 1 ;

(5)
- Concatenation Operators +
  - join strings (and other data type) together to produce a new string
  - e.g.,

```
    float j = 6.8f;
    System.out.print("float j = " + j);
```

- Data Type Conversion
  - conversion of one data type into another type
  - needed when more than one type of data objects appear in an expression/assignment
  - Three types of conversions:
    - Explicit conversion
    - Arithmetic conversion
    - Assignment conversion

(1)
- Explicit Conversion
  - uses the type cast operators
    e.g.,
    ```
    int num;
    double result;
    result = (double) num;
    ```
  - converts the operand into the type as indicated by the type cast operator

(2)
- Arithmetic Conversion
  - used in any operation that involves operands of two different types
  - converts the operands to the type of the higher ranking of the two
  - ranking (from high to low):
    double > float > long > int > short > byte
    - floating-point type FIRST then integer type

(3)
- Assignment Conversion
  - convert the type of the result from an expression to the type of the left hand side (variable) if they are different
  - e.g.,
    ```
    int i;
    double x=2.5, y=5.3;
    i = x+y;          // i will have a value of 7
    ```
    (but Java will give you an error message during compilation)
  - If the variable on the left-hand side of the assignment statement has a higher rank or same rank as the expression, then there is no loss of information
  - When loss of information -> syntax error (during compile)
- Conversion between Char and Numeric Types:
  - from integer to character:
    - only lower sixteen bits of data are used
      ```
      char ch = (char)0xAB0041
      ```
      (note: character has 2 byte only)
  - from floating-point to char:
    - integral part of the floating-point value is cast into char
      ```
      char ch = (char)65.78;
      ```
  - from char to a numeric data type:
    - character's Unicode is cast into the specified numeric type
    - ```
      int i = (int) 'A' ;  // i is 65
      int j = '2' + '3' ;  // i is 50 + 51, i.e., 101
      ```
- Precedence of Operators
  - needed when evaluating an expression
  - decreasing order:

()	parentheses
++, --	increment, decrement
+,-	unary
(Type)	type cast
*, /, %	multiplication, division, modulus
+,-,+	binary addition, subtraction, String concatenation
=,+=,-=,*=,/=	assignment

  - from left to right if same level)
      ```

- Programming Style
  - more space!!!
    should use indentation, blank spaces, blank lines

  - use meaningful names
    - class names
      - first letter of each word should be capitalized
        CurrencyExchange
    - Variable, method and package names
      - lowercase letters should be used
      - If the name consists of several words, use capital letter
        for the first letter of subsequent words
        convertDollars()
    - Constants
      - all uppercase letters
      - underscores should be used to separate between words
        MAX_LIMIT
  - Add comments to your program during programming
    - At the beginning of the program, put comments to say what
      the program will do; your name and date
    - put comments to record the modifications
    - If the purpose of a portion of the program is not clear
      put comments to say what the program (or this part) is doing