Lecture 2

Comments, Variables, Expressions, Arithmetic, Standard I/O

Introduction to Computer Programming - CST8110

Algonquin College - David Lareau

Fall 2017

School of Advanced Technology
ALGONQUIN COLLEGE

Comments

```
public class Comments {

  public static void main(String [] args) {
    // this is a single line comment

    /* this is also a single line comment */

    /* This is a multi-line comment.
    Comments are ignored by the compiler. */
  }

}
```

Comments are for you

Comments are mostly for you in this course. Feel free to write most of them in your native language.

```
public class Comments2 {

  public static void main(String [] args) {
    // demandez à   l'utilisateur leur âge
  }

}
```

Warning

- Always use english if the comment is meant for me.

Printing a Line of Text

```
public class HelloWorld {

  public static void main(String [] args) {
    // Print a message to the standard output.
    System.out.println("Hello World");
  }

}
```

Note

- The standard output is normally the command line terminal screen.
- Most statements end with a semi-colon.
- Text starts and ends with straight double quotes (").

Literals

Code can contain constant values known as literals (a.k.a. hard-coded values).

- A whole number: 16
- A real number: 3.1416
- A single symbol: 'c'
- A sequence of symbols: "David Lareau"

Note

- A single letter literal starts and ends with straight single quotes (').
- A text literal starts and ends with straight double quotes (").

Variables

Variables are names corresponding to a section of memory.
Variables have a type, indicating what kind of data is stored in its memory.

- int: an integer type for whole numbers.
- double: a type for real numbers.
- char: a type for a single symbol.
- String: an object type that can store a sequence of char. We will learn more about objects later.

Variables Value

```
public class Variables {
  public static void main(String [] args) {
    // a variable does not have to be initialized
    // with a value until it is first used.
    double average;

    // a variable can change value during
   execution.
    average = 2;

    // Assigning a value to a variable is not
    // algebra. The following line increments
    // the variable's value by 1.
    average = average + 1;
    // The program evaluates the line as follow:
    // 1) take the value in 'average'
    // 2) add 1 to it
    // 3) put the result in 'average'
  }
}
```

Variables and Literals

```
public class VariablesAndLiterals {

  public static void main(String [] args) {
    // integer type variables can hold whole
   numbers.
    int age = 16;

    // double type variables can hold real numbers.
    double average = 3.98;

    // String type variables can hold text.
    String name = "David";

    // char type variables can hold a single
   letter/symbol.
    char symbol = '$';
  }

}
```

Variables and Literals

Demo

- Draw the memory representation of the variables of the previous code examples.
- The String variables are drawn quite different because they are reference variables. We will learn more about those later.

Demo

- Try storing a value in a variable where the type do not match (e.g. a real number in a integer variable).
- Cast from double to integer.

Arithmetic (+, -, *)

You can add, subtract and multiply numeric values.

```
public class Arithmetic {

  public static void main(String [] args) {
    System.out.println(2.1 + 5.3);
    System.out.println(2 - 5);
    System.out.println(2 * 5); // multiply
  }

}
```

Arithmetic (/)

You can divide number values, but if the two values are integers the result is an integer.

```
public class Arithmetic {

  public static void main(String [] args) {
    System.out.println(5 / 2.0); // 2.5 (a double)
    System.out.println(5 / 2); // 2 (an int)
  }

}
```

Integer division is as if the result was casted as an int. The result loses it's decimal point values.

Promotion

Note

- Integers are promoted to double if the other number is a double.

```
public class Arithmetic {

  public static void main(String [] args) {
    int x = 3;
    double y = 2;
    int z = x + y; // does not compile, cannot put
   a double in an integer
    int z = (int)(x + y); // ok, lose the decimal
   point
    double a = x + y;
  }

}
```

Remainder (%)

To get the remainder value of a division you can take advantage of the integer division.

```
public class Arithmetic {

  public static void main(String [] args) {
    int dividend = 5;
    int divisor = 2;
    int division = dividend / divisor; // 2
    int remainder = dividend - division * divisor;
   // 5 - 4 = 1
  }

}
```

Remainder (%)

But there is a **modulo** operator that can calculate the remainder for you:

```
public class Arithmetic {

  public static void main(String [] args) {
    int remainder = 5 % 2; // 1

    System.out.println(125 % 100); // 25
  }

}
```

Cyclic Values (%)

The modulo operator is useful when working with cyclic values like seconds [0, 59] or cents [0, 99].

Demo

- Let's write a program that converts cents to dollar and cents.
- (e.g. 243 cents is 2 dollars 43 cents).

Note

- Java supports modulo on the double type as well even though it makes no sense to have remainder with those.
- This is to ease managing cyclic values on those types.

Operator Precedence

The order of operation is similar to what is the norm in mathematics. You can add parenthesis to modify it, or make it clearer.

```
public class Arithmetic {
  public static void main(String [] args) {
    int x;
    x = 5 * 3 + 1; // 15 + 1  = 16
    x = 1 + 5 * 3; // 1 + 15  = 16
    x = (1 + 5) * 3; // 6 * 3 = 18

    // beware integer arithmetic
    x = 5 * 3 / 2; // 15 / 2 = 7
    x = 3 / 2 * 5; // 1 * 5  = 5
    x = (int)((5 * 3) / 2.0); // 15 / 2.0 = 7.5 = 7
    x = (int)((3 / 2.0) * 5); // 1.5 * 5  = 7.5 = 7
  }
}
```

Operator Precedence

Links

- http://www.cs.bilkent.edu.tr/ guvenir/courses/CS101/op_precedence.html

String Concatenation

Joining two String of text together is called Concatenation.

```
public class Concatenation {

  public static void main(String [] args) {
    String beginning = "Hello, my name is ";
    String name = "David";
    String sentence = beginning + name + ".";
    System.out.println(sentence);
  }

}
```

String Concatenation

Converting an int or double to String is as simple as concatenating them to a String.

```
public class Concatenation {

  public static void main(String [] args) {
    int age = 16;
    String sentence;

    sentence = "I'm " + age + " years old.";
  }

}
```

Standard Input

Note

- The Standard Input is normally the command line terminal, just like the Standard Output.
- System.in works with low-level bytes, so we will use a utility called a Scanner.
- The Scanner class is part of the Java Standard Library in the java.util package.

Standard Input

```
// the import statement lets the compiler know
    where Scanner can be found.
import java.util.Scanner;

public class StandardInput {

  public static void main(String [] args) {
    // create a scanner object, and
    // attach it to the standard input
    Scanner scanner = new Scanner(System.in);

  }

}
```

Standard Input

Note

- Reading input is not preceded by an explicit prompt. You must tell the user what you expect them to do with a print to System.out first.

```
import java.util.*;

public class StandardInput {
  public static void main(String [] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("What is your name?");
    String line = scanner.nextLine();
    System.out.println("Your name is: " + line);
  }
}
```

Useful Comments

```
// Written by David Lareau on May 18th, 2017.
// Examples of good comment practices.
import java.util.*;

public class Comments3 {

  public static void main(String [] args) {
    Scanner scanner = new Scanner(System.in);

    // ask the user for its name
    System.out.print("Enter your name: ");
    String name = scanner.nextLine();

    // print a story involving the user
    System.out.println(name +
      " has a little lamb.");
  }

}
```