Programming in the Small I: Names and Things (Part I)

188230 Advanced Computer Programming

Asst. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

Agenda

- The Basic Java Application
- Variables and the Primitive Types
- Strings, Objects, Enums, and Subrountines
- Programming Exercise

Programs in the Large

- A computer can perform only very simple operations
- A computer performs complex tasks by stringing together large numbers of such operations
- Suck tasks must be ”scripted” in complete and perfect detail by programs.
- **The design of the overall structure** of a program is called ”**programming in the large**”

Programming in the Small

- **Programming in the small**, which is sometimes called **coding**
  - The details of the design of the programming in the large
- The details are the explicit, step-by-step instructions for performing fairly small-scale tasks
- When you do coding, you are working fairly ”close to the machine”

Programming in the Small

- Small does not mean that it's unimportant
- This material is an essential foundation for all types of programming
- If you don't understand it, you can't write programs
  - No matter how good you get at designining their large-scale structure

Programming Languages

- A program is a sequence of simple instructions that a computer can execute
  - Thus programs have to be written in programming languages
- Programming languages differ from ordinary human languages
  - It's completely unambiguous and very strict about what is and is not allowed in a program
  - The rules that determine what is allowed are called the syntax of the langauge

Syntax of the Language

- Syntax rules specify
  - The basic vocabulary of the language
  - How programs can be constructed using things like loops, branches, and subroutines
- A syntactically correct program is one that can be successfully compiled or interpreted
  - Programs that have syntax errors will be rejected
- You have to develop a detailed knowledge of the syntax of the programming language

Semantics

- You want a program that will run and produce the correct result
  - It's not enough to write a program that will run
- The meaning of the program has to be right
- The meaning of a program is referred to as its semantics
- A semantically correct program is one that does what you want it to

Pragmatics

- A program can be syntactically and semantically correct but still be a pretty bad program
- A good program has ”style”
  - It is written in a way that will make it easy for people to read and to understand
- It has an overall design that will make sense to human readers
- These aspects of programming are sometimes refered to as pragmatics

HelloWorld.java

```
// A program to display the message
// "Hello World!" on standard output
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}    // end of class HelloWorld
```

- What's the command that actually displays the message ”Hello World”?

A built-in subrountine

- A built-in subroutine consists of the instructions for performing some task, chunked together and given a name
- That name can be used to “call” the subroutine whenever that task needs to be performed
- A built-in subroutine is one that is already defined as part of the language and therefore automatically available for use in any program.
- What's the example of a built-in subrountine in ”HelloWorld.java” program?

Comments

- Comments in a program are entirely ignored by the computer
- Comments are for human readers only
- Programs are meant to be read by people as well as by computers
- Without comments, a program can be very difficult to understand

Comments in Java

- Java has two types of comments
- The first type, used in the above program, begins with // and extends to the end of a line.
  - The computer ignores the // and everything that follows it on the same line
- Java has another style of comment that can extend over many lines
  - The comment begins with /* and ends with */

A Java class

- All programming in Java is done inside “classes.”
- The first line in the above program (not counting the comments) says that this is a class named HelloWorld.
- “HelloWorld,” the name of the class, also serves as the name of the program
- Not every class is a program

Java class vs. Java program

- In order to define a program, a class must include a subroutine named main, with a definition that takes the form:

```
public static void main(String[] args) {
        statements
}
```

- When you tell the Java interpreter to run the program, the interpreter calls the main() subroutine, and the statements that it contains are executed

”public” keyword

- The word “public” in the first line of main() means that this routine can be called from outside the program
- This is essential because the main() routine is called by the Java interpreter, which is something external to the program itself

Program Template

```
public class (program-name)   {
 (optional-variable-declarations-and-subroutines)
    public static void main(String[] args) {
        (statements)
    }
    (optional-variable-declarations-and-
subroutines)
}
```

Class name and File name

- The name on the first line is the name of the program, as well as the name of the class
- If the name of the class is HelloWorld, then the class should be saved in a file called HelloWorld.java
- When this file is compiled, another file named HelloWorld.class will be produced
- This class file, HelloWorld.class, contains the Java bytecode that is executed by a Java interpreter

Agenda

- The Basic Java Application
- Variables and the Primitive Types
- Strings, Objects, Enums, and Subrountines
- Programming Exercises

Names

- Names are fundamental to programming
- In programs, names are used to refer to many different sorts of things
- A programmer must understand the rules for giving names to things and the rules for using the names to work with those things

Syntax Rules of Names in Java

- A name is a sequence of one or more characters
- It must begin with a letter or underscore and must consist entirely of letters, digits, and underscores
- Some legal names:
  - N n rate x15 quite a long name HelloWorld
- No spaces are allowed in identifiers
  - HelloWorld (legal)
  - Hello World (illegal)

Syntax Rules of Names in Java

- Upper case and lower case letters are considered to be different
  - HelloWorld, helloworld, HELLOWORLD, and hElloWorLD are all distinct names
- Certain names are reserved for special uses in Java, and cannot be used by the programmer for other purposes
  - These reserved words include: class, public, static, if, else, while, and several dozen other words.

Pragmatics of Naming

- Names of classes should begin with upper case letters
- Names of variables and of subroutines should begin with lower case letters
- When a name is made up of several words, capitalize each word,
  - Examples: HelloWorld, interestRate

Compound Names

- Compound names consist of several ordinary names separated by periods
- Things in Java can contain other things
- A compound name is a kind of path to an item through one or more levels of containment.
- Example: System.out.println(...)
  - Class called "System" contains attribute called "out" which in turn contains function called "println"
- Non-compound names are called simple identifiers

Identifiers

- An identifer is a name for things
- An identifier includes both compound names and non-compound names
  - Examples: Compound names => System.out.println
  - Non-compound names => HelloWorld
- Is a reserved word such as ”if” an identifer?

Variables

- In machine language, data can only be referred to by giving the numerical address of the location in memory where it is stored
- In Java, names are used instead of numbers to refer to data
- Variable is a name to refer to a location in memory that can hold data
- Think of a variable as a container or box where you can store data that you will need to use later

Variable and Expression

- In Java, the only way to get data into a variable -- that is, into the box that the variable names -- is with an assignment statement
  - variable = expression;
  - where expression represents anything that refers to or computes a data value
- Examples
  - rate = 0.07;
  - interest = rate * principal;

Types

- A variable in Java is designed to hold only one particular type of data
  - It can legally hold that type of data and no other
  - Java is a strongly typed language
- There are eight so-called primitive types built into Java
  - byte, short, int, long, float, double, char, and boolean

Integer Literals

- A literal of type long having ”L” as suffix
  - 728476874368L
- A numeric literal that begins with a 0 is interpreted as an octal number
  - 045 represents the number 37, not the number 45
- Hexadecimal literal begins with 0x or 0X, as in 0x45 or 0xFF7A
  - Example: 0x45 or 0xFF7A

Non-Integer Literals

- A unicode literal consists of \u followed by four hexadecimal digits
  - The character literal '\u00E9' represent the Unicode character that is an ”e” with an acute accents
- Boolean literals include ”true” and ”false”
- A string is a sequence of characters
  - To present **the string value** I said, *”Are you listening!” with a line feed at the end, you need to type* ***the string literal***
  - *"I said, \"Are you listening!\"\n"*

Variables in Programs

- A variable can be used in a program only if it has first been declared
- A variable declaration statement is used to declare one or more variables and to give them names
- When the computer executes a variable declaration
  - It sets aside memory for the variable and associates the variable's name with that memory

Variable Declaration

- A simple variable declaration takes the form
  - Type-name variable-name-or-names
- The variable-name-or-names can be a single varaible name or a list of variable names separated by commas
- Good programming style is to declare only one variable in a declaration statement, unless the variables are closely related in some way

Sample Variable Decclarations

- Examples of variable name and variable names declaration:
  - boolean isFinished;
  - char firstInitial, middleInitial, lastInitial;
- It is also good style to include a comment with each variable declaration to explain its purpose in the program
  - double principal; // Amount of money invested.
  - double interestRate; // Rate as a decimal, not percentage.

Variable Declarations

- Variable declarations can occur anywhere inside the subroutine, as long as each variable is declared before it is used in any expression
- Variables declared inside a subroutine are called local variables for that subroutine
  - They exist only inside the subroutine, while it is running, and are completely inaccessible from outside
- Declare important variables at the beginning of the subroutine, and use a comment to explain the purpose of each variable

Sample Java Class Interest

```
/**
 * This class implements a simple program that
 * will compute the amount of interest that is
 * earned on $17,000 invested at an interest
 * rate of 0.07 for one year.  The interest and
 * the value of the investment after one year are
 * printed to standard output.
 */
```

Sample Java Class Interest

```
public class Interest {
   public static void main(String[] args) {
      /* Declare the variables. */
      double principal;     // The value of the investment.
      double rate;          // The annual interest rate.
      double interest;      // Interest earned in one year.
```

Sample Java Class Interest

```
/* Do the computations. */
 principal = 17000;
rate = 0.07;
interest = principal * rate;   // Compute the interest.


principal = principal + interest;
      // Compute value of investment after one year,
with interest.
      // (Note: The new value replaces the old value of
principal.)
```

Sample Java Class Interest

```
    /* Output the results. */
        System.out.print("The interest earned is $");
        System.out.println(interest);
        System.out.print("The value of the investment
after one year is $");
        System.out.println(principal);
    } // end of main()
} // end of class Interest
```

Subroutine Call Statements

- Two different subroutines are used: System.out.print and System.out.println
- The difference between these is that System.out.println adds a linefeed after the end of the information that it displays, while System.out.print does not
- Note that the value to be displayed by System.out.print or System.out.println is provided in parentheses after the subroutine name
  - This value is called **a parameter** to the subroutine.

Agenda

- The Basic Java Application
- Variables and the Primitive Types
- Strings, Objects, Enums, and Subrountines
- Programming Exercises

Subroutine

- A subroutine is a set of program instructions that have been chunked together and given a name
- In Java, every subroutine is contained in a class or in an object
- A subroutine is a "black box" which can be used without knowing what goes on inside.

Classes in Java

- Classes in Java have two very different functions

  1) A class can group together variables and subroutines that are contained in that class

     - These variables and subroutines are called **static** members of the class

  2) They are used to describe objects

     - The class is a type

Static subroutine

- In a class that defines a program, the main() routine is a static member of the class

```
public static void main(String[] args) {
        ...
}
```

- The parts of a class definition that define static members are marked with the reserved word "static", just like the main() routine of a program

Class as a Type

- String is actually the name of a class that is included as a standard part of the Java language
- String is also a type, and literal strings such as "Hello World" represent values of type String

```
String helloWorld = ”Hello World”;
System.out.println(helloWorld);
```

Subroutines in Programs

- Every subroutine is contained either in a class or in an object
- Classes contain subroutines called static member subroutines
  - public class HelloWorld { public static void **main(String args[])** {...}}
- Classes also describe objects and the subroutines that are contained in those objects
  - System.out.**println(...)**

Subroutines in Programs

- Every subroutine is contained either in a class or in an object
- Classes contain subroutines called static member subroutines
  - public class HelloWorld { public static void **main(String args[])** {...}}
- Classes also describe objects and the subroutines that are contained in those objects
  - System.out.**println(...)**

Subroutine System.out.print(...)

- System is one of Java's standard classes
- One of the static member variables in this class is named out
- Since this variable is contained in the class System, its full name is System.out
- The variable System.out refers to an object, and that object in turn contains a subroutine named print
- The compound identifier System.out.print refers to the subroutine print in the object out in the class System

Static Subroutines in Class Math

- Math.abs(x)
- Math.sin(x), Math.cos(x), and Math.tan(x)
- Math.asin(x), Math.acos(x), and Math.atan(x)
- Math.exp(x)
- Math.log(x)
- Math.pow(x,y)
- Math.floor(x)
- Math.round(x)
- Math.random()

Time Measurement Subroutine

- System.currentTimeMillis(), from the System class
- When this function is executed, it retrieves the current time, expressed as the number of milliseconds that have passed since a standardized base time
- The return value of System.currentTimeMillis() is of type long
- Just record the time at which the task is begun and the time at which it is finished and take the difference.

Sample Class TimedComputation

```
/**
 * This program performs some mathematical
 * computations and displays the results.
 * It then reports the number of seconds that the
 * computer spent on this task.
 */
```

Sample Class TimedComputation

```
public class TimedComputation {
   public static void main(String[] args) {
      long startTime; // Starting time of program, in milliseconds.
      long endTime;   // Time when computations are done, in milliseconds.
      double time;    // Time difference, in seconds.
      startTime = System.currentTimeMillis();
```

Sample Class TimedComputation

```
 double width, height, hypotenuse;  // sides of a triangle

      width = 3.0;

      height = 4.0;

      hypotenuse = Math.sqrt( width*width + height*height );

      System.out.print("A triangle with sides 3 and 4 has hypotenuse ");

      System.out.println(hypotenuse);
```

Sample Class TimedComputation

```
        System.out.print("\nHere is a random number:  ");
        System.out.println( Math.random() );
        endTime = System.currentTimeMillis();
        time = (endTime - startTime) / 1000.0;
        System.out.print("\nRun time in seconds was:  ");
        System.out.println(time);
    } // end main()
} // end class TimedComputation
```

Operations on Strings

- bool s1.equals(s2), s1.equalsIgnoreCase(s2)
- int s1.length()
- char s1.charAt(N) where N is an integer starting from 0
- String s1.substring(N,M) where N and M are integers
- int s1.indexOf(s2)
- int s1.compareTo(s2)
- String s1.toUpperCase(), s2.toLowerCase()
- String s1.trim()

Concatenation of Strings

- You can use the plus operator, +, to concatenate two strings
- The concatenation of two strings is a new string consisting of all the characters of the first string followed by all the characters of the second string
- Examples:
  - "Hello" + "World" evaluates to "HelloWorld"
  - ”Hello ” + ”World” evaluates to ”Hello World”

Enum

- An enum is a type that has a fixed list of possible values, which is specified when the enum is created
- In some ways, an enum is similar to the boolean data type, which has true and false as its only possible values
- However, boolean is a primitive type, while an enum is not.

Definition of an Enum Type

- The definition of an enum types has the (simplified) form:
  - enum enum-type-name { list-of-enum-values }
- This definition cannot be inside a subroutine. You can place it outside the main() routine of the program
- The enum-type-name can be any simple identifier
- Each value in the list-of-enum-values must be a simple identifier, and the identifiers in the list are separated by commas

Sample Enum Type

- enum Season { SPRING, SUMMER, FALL, WINTER }
- By convention, enum values are given names that are made up of upper case letters, but that is a style guideline and not a syntax rule
- Enum values are not variables
- Each value is a constant that always has the same value
- In fact, the possible values of an enum type are usually referred to as enum constants.

A Variable in Enum Type

- you can declare a variable named vacation of type Season with the statement:
  - Season vacation;
- After declaring the variable, you can assign a value to it using an assignment statement.
- The value on the right-hand side of the assignment can be one of the enum constants of type Season
- Remember to use the full name of the constant, including "Season"! For example:
  - vacation = Season.SUMMER;

Subroutine ordinal()

- One of the subroutines in every enum value is named ordinal()
- When used with an enum value, it returns the ordinal number of the value in the list of values of the enum
- The ordinal number simply tells the position of the value in the list
- That is, Season.SPRING.ordinal() is the int value 0, Season.SUMMER.ordinal() is 1, Season.FALL.ordinal() is 2, and Season.WINTER.ordinal() is 3

Sample Class EnumDemo


```
public class EnumDemo {

    // Define two enum types -- remember that the definitions

    // go OUTSIDE The main() routine!

  enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY }

  enum Month { JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC }
```

Sample Class EnumDemo

```
public static void main(String[] args) {
        Day tgif;      // Declare a variable of type Day.
        Month libra;  // Declare a variable of type
Month.


        tgif = Day.FRIDAY;    // Assign a value of
type Day to tgif.
        libra = Month.OCT;    // Assign a value of
type Month to libra.
```

Sample Class EnumDemo

```
        System.out.print("My sign is libra, since I
was born in ");
        System.out.println(libra);   // Output value
will be:  OCT
        System.out.print("That's the ");
        System.out.print( libra.ordinal() );
        System.out.println("-th month of the year.");
        System.out.println("   (Counting from 0, of
course!)");
```

Sample Class EnumDemo

```
        System.out.print("Isn't it nice to get to ");
        System.out.println(tgif);   // Output value will
be:  FRIDAY
        System.out.println( tgif + " is the " +
tgif.ordinal()  + "-th day of the week.");
                // You can concatenate enum values onto
Strings!
    }
 }
```

Programming Exercises

- Write a program called Greetings to output the message in the format

Hello, my name is <Your Name>. Now I study at <University Name>.

Sample output:

Hello, my name is Chanapat. Now I study at Grandma University.

- Write a program called CircleComputation to compute the circumference and the area of a circle.

Use Math.PI and Math.pow()

Sample output:

Circle with radius <radius> has the area as <area> and the circumference as <circumference>

References

- David J. Eck, ”Introduction to Programming Using Java”, Version 5.0, December 2006 http://math.hws.edu/javanotes/