Java Programming

Program Organization and Comments

Dan McElroy
August 2018

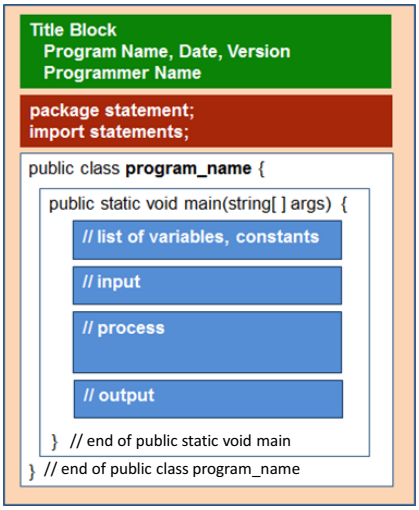
Programming Organization

The layout of a program should be fairly straight forward and simple. Although it may just look like a bunch of funny letters and symbols, once you recognize the pattern of how things are organized it makes it much easier to understand what is happening in the program.

Program Organization

The title block at the top of the program is made up of comments that briefly describe the program, date, version, programmer, etc.

The **package** and **import** statements come next.

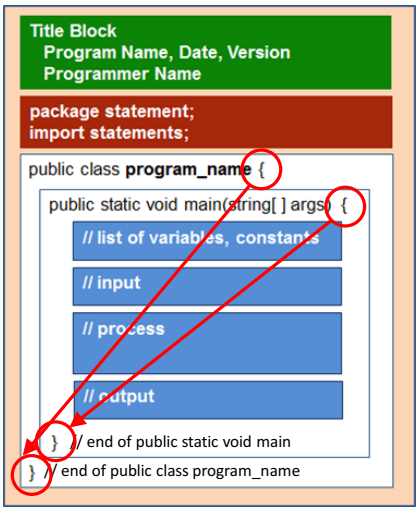


Program Organization

Everything in Java is based on objects. Objects are defined by a **class** statement. This is next. Curly braces { and } surround the block of code that belongs to the class definition. This defines a block of code. All of the code here must be stored in a file with same name as the **class** statement, followed by **.java**

Another set of curly braces { and } surround the code that belongs to **public static void main (…**

The body of the program is organized into the list of variables, input, processing and output.



Comments

Comments are written in a human language to help you or the next person better understand what the computer language is doing. Put good comments in your code even if you think that no one else will ever see them. There have been times that I was deeply involved in a project and knew every part of the project and did not think that I needed to comment my code because I knew it so well. I then started a new project and later returned to my first project and had to spend an extra amount of time and effort to figure out what I had done in the old project because I did not have good comments.

Comment Styles

1. Comments that start with // end when the end of the line is reached. Example:

```
tax = subtotal * TAX_RATE;    // compute the tax
```

2. A block of comments can be enclosed using /* and */ Example:

```
/* Java soda machine
   Lets the user select one or more sodas
   Computes the subtotal, tax and total to be paid
   Displays the subtotal, tax and total
*/
```

The TITLE Comment Block

Use a comment block at the top of each file to indicate the program's name, the programmer's name, the date and version of the program, and a brief description of the program and any input or output parameters. The helps identify what the file or program is used for without needing to read the program code and try and figure out what it does.

```
/* AvgTemp.java : Compute the average temperature
   Version: 1.0
   Date 9/4/2018
   Programmer: Dan McElroy
   Class: CIS084 Java Programming

   Inputs: 10 temperatures
   Output: Average temperature
*/
```

Comment Each Paragraph

English essays either use a blank line between paragraphs or indent each paragraph with a few spaces at the beginning of the paragraph. When you use several lines of code that are related to each other and do a specific task, put a blank line and a comment to briefly describe the code.

```
// Compute RegHours and OvertimeHours
if (Hours <= 40)
{
        RegHours = Hours;           // only regular hours
        OvertimeHours = 0.0;        // no overtime
}
else
{
        RegHours = 40;              // regular pay first 40 hours
        OvertimeHours = Hours-40;   // OT for anything over 40
}

// Compute the paycheck
RegPay = RegHours * PayRate;
OvertimePay = OvertimeHours * PayRate * 1.5;
Paycheck = RegPay + OvertimePay;
```

Comment Tricky Code

Put a comment on individual lines of code if you did anything that may take some extra work figuring out what the code is doing.

```
// Compute RegHours and OvertimeHours
if (Hours <= 40)
{
        RegHours = Hours;            // only regular hours
        OvertimeHours = 0.0;         // no overtime
}
else
{
        RegHours = 40;               // regular pay first 40 hours
        OvertimeHours = Hours-40;    // OT for anything over 40
}

// Compute the paycheck
RegPay = RegHours * PayRate;
OvertimePay = OvertimeHours * PayRate * 1.5;
Paycheck = RegPay + OvertimePay;
```

The **package** Statement

Every **class** in Java is contained in something called a ***package***. The **package** statement is automatically created when you use an IDE to create your program.

This is not a problem unless you use the online compile or the command line interface (CLI) compiler, in which case you need to remove the line that starts with **package.**

The **import** Statement

The **import** statement is used to merge code from another file into your program as the program is being compiled.

The **import** statement allows us to write Java programs and use code that has already been written and is already stored on the disk.

The **public class …** statement

Everything in Java is object oriented. Objects are defined with the **class** statement. Look closely at the **class** statement below. It has the name of the program followed by an open curly-brace { There should be a close curly-brace at the end of the file that matches up with the open curly-brace. Everything within the curly-braces is the block of code that belongs to the **class.**

```
public class paycheck {
```

public static void main (…

The executable code for our program is placed within the open and close curly-braces. ***String[ ] args*** provides a way to receive information if the program is launched from the command line or some other program. Additional subroutines may be called (activated) by main and they will have their own open and close curly-braces enclosing their code.

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

Input, Process, Output

It does not really matter which language you use, most programs are broken up into pieces that each input some data, process it and then output it. In a console application, the **main** usually inputs from the keyboard and outputs to the display. Smaller pieces of code can be placed in subroutines that in turn input, process and output data that is usually supplied by another part of the program rather than the keyboard

List of Variables and Constants

Variables in Java are declared as follows and can be initialized at the same time they are declared:

data_type variable_name semicolon

Examples:

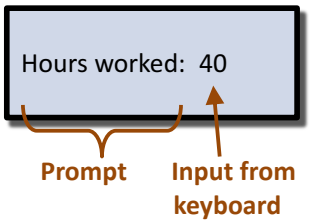
```
int  length;
int  count = 20;
double price;
```

Input Data to be Processed

Data must be input into the program before it can be processed. Data can be input from the keyboard or another device such as a disk file.

When data is input from the keyboard, it is necessary to provide a 'prompt' message on the screen that asks for the data, otherwise the user may just see a flashing cursor _ and not know that they need to do anything, or what type of data should be input.

The Prompt Message



The word “prompt” comes from the theater. A person might hold up a card to help the actor remember what line he or she needs to say next. TV shows use a machine called a teleprompter.

```
System.out. print ("Hours worked: "); /* prompt */
hours = stdin.nextDouble(); /* input the hours */
```

Provide a colon : and a space at the end of the prompt to keep the user’s input from sitting right next to the prompt message.

Process the Data

The processing of the data will be dependent on the requirements of the program.

You may need to create variables in addition to the variables needed for the input and output data.

Output the Result

Outputting a text string of text to the display in Java is fairly easy. There are three common ways to do it:

```
System.out.print ("You passed the test");  // stays on same line
System.out.println ("You passed the test");  // moves to next line
System.out.printf ("You passed the test\n"); // with formatting
```

The printf statement provides a way to format multiple pieces of information on a single line that is being output to the display.