Getting Started With Java – Variables

In order to create useful programs, we need the ability to store information and retrieve that information as needed. Information is stored in the memory of the computer in specially reserved areas known as *variables*.

Identifiers

In order to keep track of these variables, we use *identifier names* (or simply *identifiers*) for each *variable*, *method*, or *class* in our program.

The rules for creating any type of identifier are:

1. Any character used in an identifier must be a letter or the alphabet, a digit (0, 1, ..., 9), or an underscore character (_).

2. The first character cannot be a digit.

3. By convention, *classes* are given identifiers using PascalCase, where each word begins with a capital letter.

4. By convention, *variables* and *methods* are given identifiers using camelCase, where the first letter is lowercase, and any other words in the identifier are uppercase.

5. You cannot use any of the following *reserved words*, each of which has a predefined meaning in the Java language.

abstract	**assert**	**boolean**	**break**	**byte**	**case**
catch	**char**	**class**	**const**	**continue**	**default**
double	**do**	**else**	**enum**	**extends**	**false**
final	**finally**	**float**	**for**	**goto**	**if**
implements	**import**	**instanceof**	**int**	**interface**	**long**
native	**new**	**null**	**package**	**private**	**protected**
public	**return**	**short**	**static**	**strictfp**	**super**
switch	**synchronized**	**this**	**throw**	**throws**	**transient**
true	**try**	**void**	**volatile**	**while**	

Declaring Variables

To reserve space in memory for variables, we need to write a *declaration statement*, where we specify the type of a variable and its identifier (i.e., name). A simple declaration might look like

```
<type> <identifier>;
```

For example:

```
int count;
float area;
boolean finished;
```

Note the a declaration statement, like any other statement in Java is *terminated* (i.e., ended) by the semicolon character. It is possible to declare multiple variables of the same type using a single declaration statement of the form

```
<type> <identifier1>, <identifier2>, ..., <identifierk>;
```

For example:

```
float radius, circumference, area;
```

Assigning Values to Variables

Declaring a variables reserves space in the computer's memory for information of a particular data type (e.g., int, or float, or char). This variable does not become useful until we assign a value to it. There are several ways to assign a value to a variable.

1. When declaring the variable

   ```
   int total = 15;
   ```

2. Immediately after declaring the variable

   ```
   int total;
   total = 15;
   ```

3. After declaring the variable, but later in the program

   ```
   int total;
   .
   .
   .
   total = 15;
   ```

4. Using the value from another variable

   ```
   int a = 5;
   int b;

   b = a;
   ```

Converting Between (Numeric) Data Types

The assigning a value to a variable, it is important that the data type match the value. For numeric data types, it is possible to convert from a smaller data type to a larger data type, but **not** the reverse.

legal conversions: `byte → short → char → int → long → float → double`

It is possible to force Java to accept a conversion, even if it breaks Java's own rules. This is called a *cast*, and it is essentially a promise to the compiler that you know what you are doing. It has the form:

<variable> = (<data type>) <questionable data>;

For example,

```
short a;                    // range of -30000 to +30000
a = (short) 75000;          // 75000 doesn't fit in short - override!
System.out.println(a);      // output will be 9464 since 75000
                            //   doesn't fit!
```

Output of Variables

The print and println methods can be used to output variables or combinations of variables. Keep in mind that the variable must have been given a value before you try to output it, or you will get an error.

```
class PrintValue
{
   public static void main(String[] args)
   {
      int length = 5;
      int width = 10;
      System.out.print("The length is " + length);
      System.out.println(" and the width is " + width);
   }
}
```

```
The length is 5 and the width is 10
```

Constants

Java allows us to associate an identifier with a constant value through the use of the final modifier in the declaration of a variable. By convention, constants are given identifiers which are all upper case, which helps easily identify them when reading code.

```
final int CLASS_SIZE = 30;
final char TERMINATOR = '*';
final float PI = 3.1415;
```

Once an identifier has been declared to be final, its value can never be changed.

Exercises

1. What is the difference between identifiers used for classes and those used for variables and methods? Create some examples of each to illustrate your answer.

2. Identify, with reasons, any identifiers that should not be used for Java variables.
   (a) digitSum
   (b) retail price
   (c) switch
   (d) heightPlusDepth
   (e) this&That
   (f) priceIn$
   (g) number-of-wins
   (h) ageDuGarcon
   (i) average_age
   (j) This

3. A code fragment is a small section of code taken from inside a program. If you wish to test a code fragment, you need to make sure it is part of a complete program.

   What would be printed by each code fragment? Try to do this in your head or on paper first, and then check your answer with a classmate and/or as part of a Java program.

   (a)
   ```
   int i = -47;
   int j = 45;
   System.out.println("The value of i is " + i
                   + "\nwhile the value of j is " + j + ".");
   ```

   (b)
   ```
   boolean done = false;
   System.out.println("We are not " + done + " yet.");
   ```

   (c)
   ```
   double x = 0.012;
   double y = 2.7374e1;
   System.out.println("x -> " + x + "\ny -> " + y);
   ```

4. What would be printed by the following **code fragment** (you will need to put the program framework around this code fragment to run it)?

   ```
   final double PI = 3.1415;
   double diameter = 10;
   double radius, area;

   radius = diameter / 2;
   area = PI * radius * radius;

   System.out.print("A circle of radius " + radius);
   System.out.println(" has an area of " + area);
   ```