Introduction to Programming

with Java, for Beginners

Fundamentals – Part I:
• Comments & Literals
• Operators & Expressions
• Primitive Types & Variables
• Declaration & Assignment Statement
• Strings and Printing

Recap

• Computing
  • Use computer to solve a task
  • Why? – Inherently faster than humans

• Programming Language
  • Language that humans can write to instruct the computer
    ➢ Syntax – grammar of language
    ➢ Semantics – meaning of the language

• Compiler
  • Checks for syntax errors
  • Further translates the language into what computer can understand (more on this later)

Comments

• Comments are used to make code more understandable to humans
• Java Compiler ignores comments

// this is a single line comment

/* this is
    * a multi-line
    * comment
*/

Literals

• Literals are the values we write in a conventional form whose value is obvious

3  // An integer has no decimal point

10.5 // a floating point (double)

‘a’ // a character has single quotes

true // The boolean literals are of two types: true, false

“hello world” // A string literal
Arithmetic Operators
- + to indicate addition
- - to indicate subtraction
- * to indicate multiplication
- / to indicate division
- % to indicate remainder of a division (integers only)
- parentheses ( ) to indicate the order in which to do things

Relational Operators
- == equal to
- != not equal to
- < less than
- > greater than
- <= less than equal to
- >= greater than equal to

Note: Arithmetic comparisons result in a Boolean value of true or false

Boolean or Logical Operators
- Like in English - conditional statements formed using “and”, “or”, and “not”

In Java
- || -> OR operator
  - true if either operand* is true
- && -> AND operator
  - true only if both operands are true
- ! -> NOT operator
  - Is a unary operator – applied to only one operand
  - Reverses the truth value of its operand

* Operand: a quantity upon which an operation is performed

Expression
- An expression is a combination of literals and operators
- An expression has a value
- Given an expression, DrJava prints its value

Welcome to DrJava
> 3
3
> 3 + 5
8
>'a' == 'A' // == Equality operator
false
> true & false // using the logical AND
false
> true || false // using the logical OR
true

Later we’ll see that an expression may contain other things
Such as variables, method calls …
Value & Type

- **Value**: Piece of data
  23, 10.5, true, ‘a’

- **Type**: Kind of data
  integer, floating point, boolean (true/false), character

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>23</td>
<td>integer</td>
</tr>
<tr>
<td>10.5 + 2.0</td>
<td>12.5</td>
<td>fractional</td>
</tr>
<tr>
<td>3 + 5 * 6</td>
<td>33</td>
<td>integer</td>
</tr>
<tr>
<td>(3 * 4)/15</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>true &amp;&amp; false</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

*Note: integer division truncates*

Types: a very important concept!

- All data values in Java have a **type**

- The type of a value determines:
  - How the value is stored in computer’s memory
  - Max/min value that data can be
  - What operations make sense for the value
  - How the value can be converted (cast) to related values

*Note: Types are very helpful in catching programming errors*

Primitive types

- Values that Java knows how to operate on directly
- We will work with 4 of Java’s 8 primitive types
  - Integer (**int**)
    - -1 42
  - Fractional or floating point number (**double**)
    - .1 3.14159 2.99792458E8
  - Character (**char**)
    - ‘J’ ‘ ula’
  - Truth value (**boolean**)
    - true false
- Java’s other types are: byte, short, long, float

Storage Space for Numeric Type

- Numeric types in Java are characterized by their size:
  - how much you can store? – computers have finite memory

- Integer and Character types

<table>
<thead>
<tr>
<th>Type</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>0 : 65535</td>
</tr>
<tr>
<td>int</td>
<td>-2147483648 : 2147483647</td>
</tr>
</tbody>
</table>

*Note: Each char is assigned a unique numeric value & numeric value is stored*

- Floating point types

<table>
<thead>
<tr>
<th>Type</th>
<th>largest</th>
<th>Smallest &gt; 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>3.4E38</td>
<td>1.4E-45</td>
</tr>
<tr>
<td>double</td>
<td>1.7E308</td>
<td>4.9E-324</td>
</tr>
</tbody>
</table>
Variables

- A **variable** is a name together with an associated value
- Value is stored in computer's memory
- Instead of knowing the location, we access the value by the name it is associated with

- Variable must always be associated with **type**
  - It tells the computer how much space to reserve for the variable
  - The value stored can vary over time

Identifiers

- Identifiers are names that you as a coder make up
  - Variable names
  - Also class and method names – more later!

**Java Rule** for Variable names

- Must start with a letter
- May consist of alphanumeric characters and the underscore (_)
- Cannot use keywords such as int, double etc.
- If not followed compiler will complain (syntax error)

Identifiers (contd..)

**Style Rule** for Variable names

- Should be a noun that starts with an lowercase letter
  - E.g. `sum`, `average`
- If the name has multiple words, capitalize the start of every word except the first (style rule)
  - E.g. `firstName`, `lastName`

**Note:** Style rule are for consistency and readability of programs

- Compiler will not complain if the rule is not followed
- If you do not follow the rule you get **penalized** in grading!

Declaring variables

- All variables must be **declared** before being used
  - Done with a declaration **statement**

**Declaration statement**

- Specifies the **type** of the variable, followed by descriptive **variable name**, followed by **semicolon (;)**

**Examples:**

```
int seats;
double averageHeight;
boolean isFriday;
char initial;
```
Storing value into Variables

- To store values into variable we use the assignment operator i.e. "=
  - Variable = Expression; -> assignment statement
  - Right hand side value is assigned to left hand side
- Important
  - Assignment statement must end with a semicolon;
  - When a variable is assigned a value, the old value is discarded and totally forgotten
- Examples
  - seats = 150;
  - averageHeight = (2.1 + 1.74 + 1.58)/3;
  - isFriday = true;

Variable value and type

- The value of a variable may be changed
  - x = 57; (assuming x is an integer)
- However its type may not
  - x = true;  // this causes an syntax error,
    // i.e.compiler will complain
- Sometimes conversions be forced (but with some precision loss)
  - E.g. int x = (int) 3.5 // x becomes 3
    ➢ This is known as casting
  - Note: Any int can be converted to a double i.e.
    ➢ double z = 3;

Initializing Variables

- It’s good idea to declare and initialize a variable in one statement
  - double milesPerHour = 60.5;
  - boolean isTall = true;
  - int age = 17;

Note:
- If a variable is not initialized before using it, you may or may not get a compiler error
  ➢ This will depend where in the program your variable is declared
  ➢ More on this later

Constants

- Variables that don’t change
  - Initialize a value and never change it
  - Program’s computation might be affected if a variable is not consistent throughout
- Rules
  - Java Rule: Must have the keyword final before the type
  - Style Rule: Should have all caps for variable name
    ➢ If multiple words use underscore between words
  - final double PI = 3.14;
  - final int MILES_PER_GALLON = 32;
Putting it all together

- Example:
  final double PI = 3.14;
  double radius = 3;
  double area = radius * radius * PI;
  double circum = 2 * PI * radius;

- In Dr Java, to view the value of area after the statements are executed do the following shortcut:
  > area
  28.26

Another Important Type: String

- A String is an Object, not a primitive type
  - Java also has objects - cover objects later
- String is composed of zero or more chars
- A String is a sequence of characters enclosed by double quotes
  "Java"  "3 Stooges"  "富士山"
- + means concatenation for strings
  "3" + " " + "Stooges" ⇒ "3 Stooges"
- Automatic conversion of numbers to strings
  3 + " " + "Stooges" ⇒ "3 Stooges"

Examples of String creation

In Dr Java:

> String s2 = "hello";
> s2 + " you!"
"hello you!"
> s2 = "The result is " + 100;
> s2
"The result is 100"

System.out.println(String)

- Command that prints string to the output screen
- Can also print literals, and expression values
  - The answer is automatically converted to string
- Prints every time on a new line
- Useful in finding semantic errors in a program

System.out.println("hello world");
System.out.println(5)
System.out.println("x = " + x);

- To not print on new newline use:
  - System.out.print(String)