Java Language Basics

CCVT-2018

Ravi Tomar

Assistant Professor(Senior Scale)
Department of Analytics
School of Computer Science
University of Petroleum & Energy Studies

January 25, 2018

Variables

You've already learned that objects store their state in fields. However, the Java programming language also uses the term "variable" as well. This section discusses this relationship, plus variable naming rules and conventions, basic data types (primitive types, character strings, and arrays), default values, and literals.

Variables

In the Java programming language, the terms "field" and "variable" are both used; this is a common source of confusion among new developers, since both often seem to refer to the same thing.

Few Questions which arise?

- What are the rules and conventions for naming a field?
- Besides int, what other data types are there?
- Do fields have to be initialized when they are declared?
- Are fields assigned a default value if they are not explicitly initialized?

We will answer all questions in coming slides.

Variables

The Java programming language defines the following kinds of variables:

- **Instance Variables (Non-Static Fields)** Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the static keyword. Non-static fields are also known as instance variables because their values are unique to each instance of a class (to each object, in other words); the currentSpeed of one bicycle is independent from the currentSpeed of another.
- **Class Variables (Static Fields)** A class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated. A field defining the number of gears for a particular kind of bicycle could be marked as static since conceptually the same number of gears will apply to all instances. The code static int numGears = 6; would create such a static field. Additionally, the keyword **final** could be added to indicate that the number of gears will never change.
- **Local Variables** Similar to how an object stores its state in fields, a method will often store its temporary state in local variables. The syntax for declaring a local variable is similar to declaring a field (for example, int count = 0;). There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared — which is between the opening and closing braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.
- **Parameters** You've already seen examples of parameters, both in the Bicycle class and in the main method of the "Hello World!" application. Recall that the signature for the main method is public static void main(String[] args). Here, the args variable is the parameter to this method. The important thing to remember is that parameters are always classified as "variables" not "fields".

Variables

Naming:

- Variable names are case-sensitive.
- A variable's name can be any legal identifier; an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "$", or the underscore character "_".NO WHITESPACE PERMITTED.
- Recommended always begin your variable names with a letter, not "$" or "_".
- Subsequent characters may be letters, digits, dollar signs, or underscore characters
- use **camelCasing** for variable name and function names
- Class name begins with capital letter.
- CONSTANT will be in all capital.

Variables

Primitive data Types:

- Java is statically-typed.
- A variable's data type determines the values it may contain.
- Java programming language supports eight primitive data types.
  - byte- 8-bit signed two's complement integer-128 to 127
  - short- 16-bit signed two's complement integer -32,768 to 32,767
  - int 32 bit
  - long 64-bit
  - float 32 bit
  - double 64 bit
  - boolean needs 1 bit but size is not precisely defined
  - char 16 bit Unicode character

Variables

Literals: A literal is the source code representation of a fixed value; literals are represented directly in your code without requiring computation.

- An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int.
- The prefix 0x indicates hexadecimal and 0b indicates binary

-
  ```
  // The number 26, in decimal
  int decVal = 26;
  // The number 26, in hexadecimal
  int hexVal = 0x1a;
  // The number 26, in binary
  int binVal = 0b11010;
  ```
- A floating-point literal is of type float if it ends with the letter F or f; otherwise its type is double and it can optionally end with the letter D or d.
  ```
  double d2 = 1.234e2;
  float f1 = 123.4f;
  ```

Variables

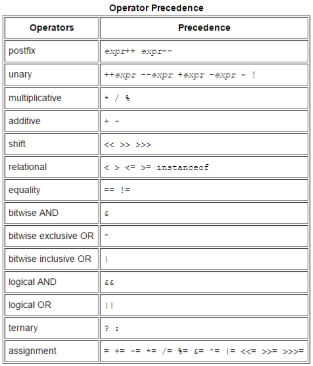
Literals Cont...

- Using Underscore Characters in Numeric Literals:
  - long creditCardNumber = 1234_5678_9012_3456L; long socialSecurityNumber = 999_99_9999L;
  - float pi = 3.14_15F;
  - long hexBytes = 0xFF_EC_DE_5E;
  - long hexWords = 0xCAFE_BABE;
  - long maxLong = 0x7fff_ffff_ffff_ffffL;
  - byte nybbles = 0b0010_0101;
  - long bytes = 0b11010010_01101001_10010100_10010010;
- You can place underscores only between digits; you cannot place underscores in the following places:
  - At the beginning or end of a number
  - Adjacent to a decimal point in a floating point literal
  - Prior to an F or L suffix
  - In positions where a string of digits is expected

- The Java programming language uses both "fields" and "variables" as part of its terminology.
- Instance variables (non-static fields) are unique to each instance of a class.
- Class variables (static fields) are fields declared with the static modifier; there is exactly one copy of a class variable, regardless of how many times the class has been instantiated.
- Local variables store temporary state inside a method.
- Parameters are variables that provide extra information to a method;
- both local variables and parameters are always classified as "variables" (not "fields").
- When naming your fields or variables, there are rules and conventions that you should (or must) follow.
- The eight primitive data types are: byte, short, int, long, float, double, boolean, and char.
- The compiler will assign a reasonable default value for fields of the above types; for local variables, a default value is never assigned.
- A literal is the source code representation of a fixed value.

Operators

This section describes the operators of the Java programming language. It presents the most commonly-used operators first, and the less commonly-used operators last. Each discussion includes code samples that you can compile and run.



Operator Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Expressions, Statements, and Blocks

- Operators may be used in building expressions, which compute values;
- expressions are the core components of statements;
- statements may be grouped into blocks.

Control Flow Statements

control flow statements supported by the Java programming language. It covers the decisions-making, looping, and branching statements that enable your programs to conditionally execute particular blocks of code.

- if-elseif-else
- switch
- while - do-while
- for (initialization; termination; increment) {
  statement(s)
  }
- for (; ;) Infinite Loop
- Enhanced For Loop:
  int[] numbers = 1,2,3,4,5,6,7,8,9,10;
  for (int item : numbers) {
  System.out.println("Count is: " + item); }

Control Flow Statements

break , continue& return

- Break is used to break the block, it can be labeled or unlabeled.
- Unlabeled-break is used to break **innermost** switch for, while, or do-while loop
- Labeled-break is used to break **outermost** statement.
- continue statement skips the current iteration of a for, while , or do-while loop.
- unlabeled continue same as unlabeled break.
- return value or simple return