The Java Language

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 688P, Fall Semester 2020

September 28, 2020

Overview

Part 1

1. Quick Review
2. Basic Stuff
   - Primitive Data Types, IEEE 754 Floating Point Standard
   - Three types of Java Variable
   - Arithmetic Operations
   - Control Statements
3. Packages and Import Statements
4. Methods
   - Polymorphism and Class Methods
5. Working with Arrays
   - One- and Multi-dimensional Arrays; Ragged Arrays

Quick Review

Popular Computer Languages

Tend to be designed for a specific set of purposes:

- FORTRAN (1950s – today). Stands for formula translation.
- C (early 1970s – today). New operating systems.
- C++ (early 1970s – today). Object-oriented version of C.
- MATLAB (mid 1980s – today). Stands for matrix laboratory.
- Python (early 1990s – today). A great scripting language.
- HTML (1990s – today). Layout of web-page content.
- Java (1994 – today). Object-Oriented language for network-based computing.
- XML (late 1990s – today). Description of data on the Web.

Basic Stuff

Primitive Data Types (Boolean, char, Integers)

Type	Contains	Default	Size	Range and Precision
boolean	True or false	false	1 bit	
char	Unicode	\u0000	16 bits	\u0000 / \uFFFF
byte	Signed integer	0	8 bits	-128/127
short	Signed integer	0	16 bits	-32768/32767
int	Signed integer	0	32 bits	-2147483648/2147483647
long	Signed integer	0	64 bits	-9223372036854775808 / 9223372036854775807

Note. A 32 bit integer has $2^{32} \approx 4.3$ billion permutatons $\rightarrow$ a working range $[-2.147, 2.147]$ billion.

Primitive Data Types (Floating-Point)

Definition. Floating point variables and constants represent values outside of the integer range (e.g., `3.4`, `-45.33` and `2.714`) and can be very large or small in magnitude, (e.g., `3.0e-25`, `4.5e+05`, and `2.34567890098e+19`).

IEEE 754 Floating-Point Standard. Specifies that a floating point number take the form:

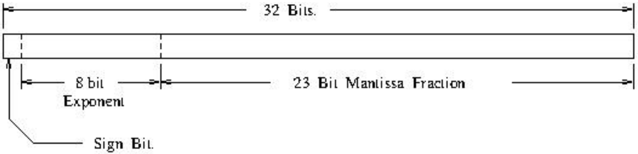
$$X = \sigma \cdot m \cdot 2^E. \tag{1}$$

Here:

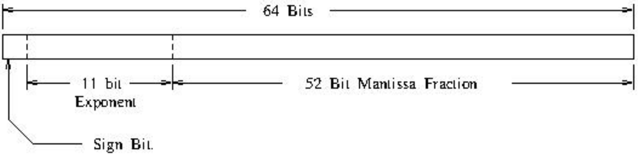
- σ represents the sign of the number.
- m is the mantissa (interpreted as a fraction $0 < m < 1$).
- E is the exponent.

IEEE 754 Floating-Point Standard

Ensures floating point implementions and arithmetic are consistent across various types of computers (e.g., PC and Mac).



IEEE FLOATING POINT ARITHMETIC STANDARD FOR 32 BIT WORDS.



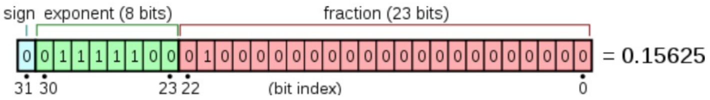
IEEE FLOATING POINT ARITHMETIC STANDARD FOR DOUBLE PRECISION FLOATS.

Largest and Smallest Floating-Point Numbers

```
====================================================================
                  Default
Type   Contains     Value   Size      Range and Precision
====================================================================

float  IEEE 754       0.0   32 bits   +- 13.40282347E+38 /
       floating point                 +- 11.40239846E-45

       Floating point numbers are represented to approximately
       6 to 7 decimal places of accuracy.

double IEEE 754       0.0   64 bits   +- 11.7976931348623157OE+308 /
       floating point                 +- 14.94065645841246544E-324

       Double precision numbers are represented to approximately
       15 to 16 decimal places of accuracy.
====================================================================
```

Working with Double Precision Numbers

Simple Example. Here is the floating point representation for 0.15625



Note. Keep in mind that floating-point numbers are stored in a binary format – this can lead to surprises.

For example, when the decimal fraction 1/10 (0.10 in base 10) is converted to binary, the result is an expansion of infinte length.

Bottom line: You cannot store 0.10 precisely in a computer.

IEEE 754 Floating Point Standard

Support for Run-Time Errors

This standard includes:

- Positive and negative sign-magnitude numbers,
- Positive and negative zeros,
- Positive and negative infinites, and
- Special Not-a-Number (usually abbreviated NaN).

NaN value is used to represent the result of certain operations such as dividing zero by zero.

Java Variables

Definition

A variable is simply a block of memory whose value can be accessed with a name or identifier. It contains either the contents of a primitive data type or a reference to an object. The object may be an instance of a class, an interface, or an array.

Four Attributes of a Variable:

- A type (e.g., int, double, float),
- A storage address (or location) in computer memory,
- A name, and
- A value.

All four parts must be known before a variable may be used in a program.

Java Variables

Variable Declarations

Variables must be declared before they can be used, e.g.,

```
int      iA =  10;
float    fA = 0.0;
double 8dA = 0.0;  <--- illegal! Cannot begin a
                        variable name with a digit.
```

What happens at compile and run time?

When a compiler encounters a variable declaration, ..

- It will enter the variable name and type into a symbol table (so it knows how to use the variable throughout the program).
- It generate the necessary code for the storage of the variable at run-time.

Three Types of Java Variable

Local Variables

- These are variables whose scope is limited to a block of code.
- Local variables are defined within the current block of code and have meaning for the time that the code block is active.

An Example

```
Source code                                          Output
============================================         ================
for ( int i = 0; i <= 2; i = i + 1)                  Loop 1: i = 0
   System.out.println( "Loop 1: i = " + i );         Loop 1: i = 1
                                                     Loop 1: i = 2
for ( int i = 0; i <= 2; i = i + 1)                  Loop 2: i = 0
   System.out.println( "Loop 2: i = " + i );         Loop 2: i = 1
                                                     Loop 2: i = 2
```

Three Types of Java Variable

Instance Variables

- These variables hold data for an instance of a class.
- Instance variables have meaning from the time they are created until there are no more references to that instance.

An Example

```
Definition of a class                Using the class
================================     ===========================
public class Complex {               Complex cA = new Complex();
   double dReal, dImaginary;         cA.dReal = 1.0;
   ....
}                                    Complex cB = new Complex();
                                     cB.dReal = 1.0;
================================     ===========================
```

cA.dReal and cB.dReal occupy different blocks of memory.

Three Types of Java Variable

Class Variables

- These variables hold data that can be shared among all instances of a class.
- Class variables have meaning from the time that the class is loaded until there are no more references to the class.

An Example

```
Definition of a class                Accessing the variable
===================================  ==========================
public class Matrix {                int i = Matrix.iNoColumns;
   public static int iNoColumns = 6.
}
===================================  ==========================
```

The variable is static – no need to create an object first.

Java Variable Modifiers

Variable Modifiers

```
===============================================================
Modifier       Interpretation in Java
===============================================================

public         The variable can be accessed by any class.

private        The variable can be accessed only by methods
               within the same class.

protected      The variable can also be accessed by subclasses
               of the class.

static         The variable is a class variable.
===============================================================
```

Constants

Setting up Constants

In Java constants are defined with variable modifier final indicating the value of the variable will not change.

An Example

```
Definition of a class                       Access
======================================      =====================
public class Math {                         double dPi = Math.PI;
   public static final double PI = 3.14..;
   .....
}
======================================      =====================
```

The variable PI is both static and final. This makes PI a class variable whose assigned value cannot be changed.

Arithmetic Operations

Standard Arithmetic Operations on Integers and Floats

```
+      -      *      /
```

Modulo Operator

The modulo operator % applies only to integers, and returns the remainder after integer division. More precisely, if `a` and `b` are integers then `a % b = k*b + r`.

Integer Division

Truncates what we think of as the fractional components of all intermediate and final arithmetic expressions, e.g.,

```
iValue = 5 + 18/4;  ===> 5 + 4  <=== Step 1 of evaluation
                    ===> 9      <=== Step 2 of evaluation
```

Probably not what we want!

Evaluation of Arithmetic Expressions

Hierarchy of Operators

Operator	Precedence	Order of Evaluation
() [] -> .	1	left to right
! ++ -- + -	2	right to left
* / %	3	left to right
+ -	4	left to right
<< >>	5	left to right
$<$ $\leq$ $>$ $\geq$	6	left to right
== !=	7	left to right
&	8	left to right
$\wedge$	9	left to right
$\vert$	10	left to right

Evaluation of Arithmetic Expressions

Hierarchy of Operators

Operator	Precedence	Order of Evaluation
&&	11	left to right
||	12	left to right
? :	13	right to left
= += *= /= &=	14	right to left
^= |= <<= >>=		
,	15	left to right

Dealing with Run-Time Errors

Dealing with Run-Time Errors

```
Source code
===================================================================
double dA = 0.0;
System.out.printf("Divide by zero:  ( 1/0.0) = %8.3f\n",  1.0/dA );
System.out.printf("Divide by zero:  (-1/0.0) = %8.3f\n", -1.0/dA );
System.out.printf("  Not a number: (0.0/0.0) = %8.3f\n",   dA/dA );

Output
===================================================================
Divide by zero:  ( 1/0.0) = Infinity
Divide by zero:  (-1/0.0) = -Infinity
  Not a number: (0.0/0.0) =      NaN
===================================================================
```

Dealing with Run-Time Errors

Print Variables containing Error Conditions

```
double dB = 1.0/dA;
System.out.printf("dB = 1.0/dA = %8.3f\n",  dB );
double dC = dA/dA;
System.out.printf("dC =  dA/dA = %8.3f\n",  dC );
```

Output

```
=========================================================
dB = 1.0/dA = Infinity
dC =  dA/dA =      NaN
=========================================================
```

Dealing with Run-Time Errors

Evaluate a Function over a Range of Values

```
System.out.println("Evaluate y(x) for range of x values");
System.out.println("===================================");

for ( double dX = 1.0; dX <= 5.0; dX = dX + 0.5 ) {
      double dY = 1.0 + 1.0/(dX - 2.0) - 1.0/(dX - 3.0) + (dX-4.0)/(dX-4.0);
      System.out.printf(" dX = %4.1f   y(dX) = %8.3f\n", dX, dY );
}
```

Output

```
Evaluate y(x) for range of x values
===================================
 dX =  1.0   y(dX) =    1.500
 dX =  1.5   y(dX) =    0.667
 dX =  2.0   y(dX) = Infinity
 dX =  2.5   y(dX) =    6.000
 dX =  3.0   y(dX) = -Infinity
 dX =  3.5   y(dX) =    0.667
 dX =  4.0   y(dX) =      NaN
 dX =  4.5   y(dX) =    1.733
 dX =  5.0   y(dX) =    1.833
```

Dealing with Run-Time Errors

Test for Error Conditions

```
Source code
==============================================================
if( dB == Double.POSITIVE_INFINITY )
    System.out.println("***  dB is equal to +Infinity" );

if( dB == Double.NEGATIVE_INFINITY )
    System.out.println("***  dB is equal to -Infinity" );

if( dB == Double.NaN )
    System.out.println("***  dB is Not a Number" );

Output
==============================================================
***  dB is equal to +Infinity
***  dB is equal to -Infinity
***  dB is Not a Number
==============================================================
```

Control Statements

Control Structures

Allow a computer program to take a course of action that depends on the data, logic and calculations currently being considered.

Machinery:

- Relational and logical operands;
- Selection constructs (e.g., if statements, switch statements).
- Looping contructs (e.g., for loops, while loops).

Common Error. Writing ...

```
if ( fA = 0.0 ) .....
```

instead of

```
if ( fA == 0.0 ) .....
```