Variables, Data Types, and Math Operators

Jim Wilson

@hedgehogjim | blog.jwhh.com | jimw@jwhh.com

What to Expect in This Module

Variables

Primitive data types

Arithmetic operators

Type conversion

Variables

Named data storage

Strongly typed

Value can be modified

```
int dataValue;
dataValue = 100;
```

```
int myInfo = 200;
```

Naming Variables

- Variable naming is based on a combination of rules and conventions
  - Rules allow the use of letters, numbers, $ and _
    - By convention only letters and numbers are used
  - Rules require that first character is not a number
    - By convention it is always a letter
  - By convention follow the style often referred to as “Camel Case”
    - First letter is lower case
    - Start of each word after the first is upper case
    - All other letters are lower case

```
int total;
int grade4;
int bankAccountBalance;
int level2Training;
```

Using Variables

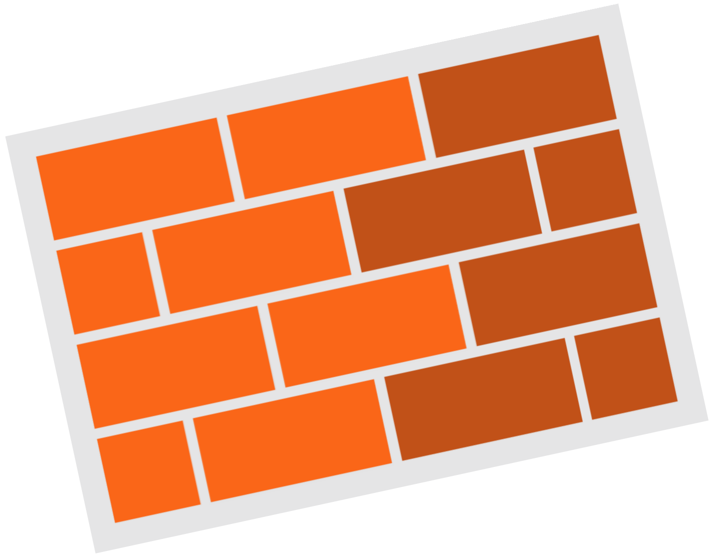
```java
public class Main {
  public static void main(Strings[] args) {
    int myVar;
    System.out.println(myVar);
    myVar = 50;
    System.out.println(myVar);

    int anotherVar = 100;
    System.out.println(anotherVar);

    myVar = anotherVar;
    System.out.println(myVar);

    anotherVar = 200;
    System.out.println(anotherVar);

    System.out.println(myVar);
  }
}
```

```
Error
50
100
100
200
100
```

Primitive Data Types

Built into the language

Foundation of all other types



Four categories of primitive types

- Integer
- Floating point
- Character
- Boolean

Integer Types

Type	Size (bits)	Min Value	Max Value	Literal Format
byte	8	-128	127	0
short	16	-32768	32767	0
int	32	-2147483648	2147483647	0
long	64	-9223372036854775808	9223372036854775807	0L

```
byte numberOfEnglishLetters = 26;
short feetInAMile = 5283;
int milesToSun = 92960000;
long nationalDebt = 18100000000000L;
```

Floating Point Types

- Implementation of IEEE 754 floating point standard
- Stores values containing a fractional portion
- Supports positive, negative, and zero values

Type	Size (bits)	Smallest Positive Value	Largest Positive Value	Literal Format
float	32	1.4×10^{-45}	3.4×10^{38}	0.0f
double	64	4.9×10^{-324}	1.7×10^{308}	0.0 or 0.0d

```
float milesInAMarathon = 26.2f;
double atomWidthInMeters= 0.0000000001d;
```

Character and Boolean Types

- The char type stores a single Unicode character
  - Literal values placed between single quotes
  - For Unicode code points, use \u followed by 4-digit hex value

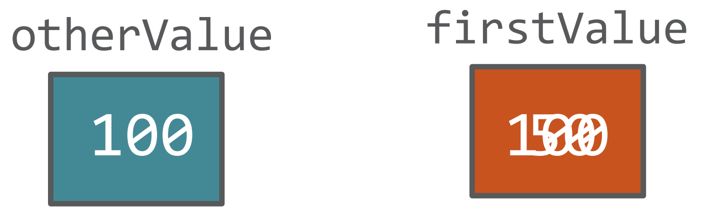
```
char regularU = 'U';
char accentedU = '\u00DA'; // Ú
```

- The boolean type stores true/false values
  - Literal values are true and false

```
boolean iLoveJava = true;
```

Primitive Types Are Stored By-value

```
int firstValue = 100;
int otherValue = firstValue;
firstValue = 50;
```



Arithmetic Operators

Basic operators

+ - * / %

Prefix/postfix operators

++ --

Compound assignment operators

+= -= /= *= %=

Basic Math Operators

	Operator	Floating Point Example		Integer Example	
Add	+	1.0 + 2.0	**3.0**	1 + 2	**3**
Subtract	-	5.0 - 4.0	**1.0**	5 - 4	**1**
Multiply	*	4.0 * 2.0	**8.0**	4 * 2	**8**
Divide	/	13.0 / 5.0	**2.6**	13 / 5	**2**
Modulus	%	13.0 % 5.0	**3.0**	13 % 5	**3**

Prefix / Postfix Operators

++ increments value by 1

-- decrements value by 1

As prefix applies operation before returning value

As postfix applies operation after returning value

```
int myVal = 5;
System.out.println(++myVal);    6
System.out.println(myVal);      6
```

```
int myVal = 5;
System.out.println(myVal++);    5
System.out.println(myVal);      6
```

Compound Assignment Operators

```
int myVal = 50;
myVal -= 5;
System.out.println(myVal);
```

45

```
int result = 100;
int val1 = 5;
int val2 = 10;
result /= val1 * val2;

System.out.println(result);
```

50

100

2

Combines an operation and assignment

Applies result of right side to left side

Stores that result in variable on left side

Available for 5 basic math operators

+=   -=   *=   /=   %=

Operator Precedence

Operators are evaluated in a well-defined order

Operators of equal precedence are evaluated left-to-right

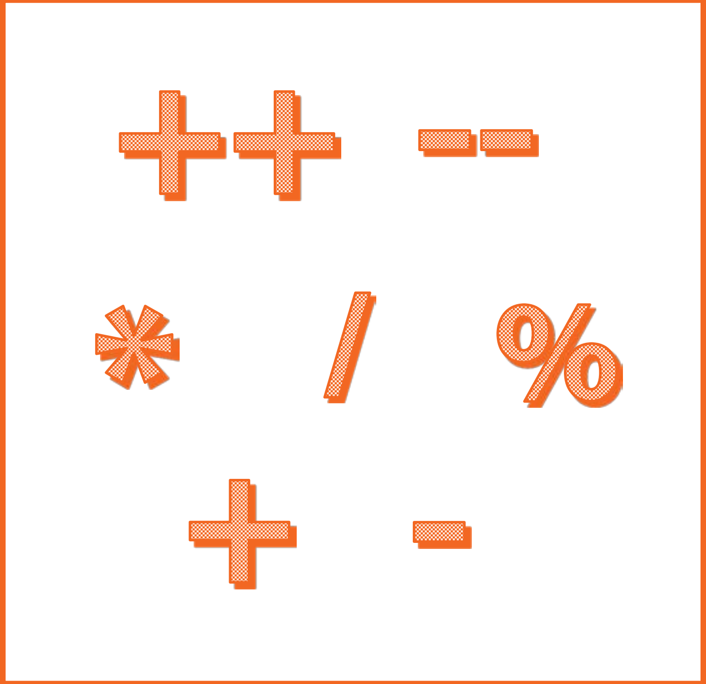
Can override precedence with parenthesis

Nested parenthesis evaluated from the inside out

Postfix	x++ x--
Prefix	++x --x
Multiplicative	* / %
Additive	+ -

Demo

Operator Precedence



Type Conversion

Implicit type conversion

- Conversions performed automatically by the compiler

```
int iVal = 50;
long lVal = iVal;
```

Explicit type conversion

- Conversions performed explicitly in code with cast operator

```
long lVal = 50;
int iVal = (int) lVal;
```

Implicit Type Conversion

Widening conversions are automatic

Mixed integer sizes

- Uses largest integer in equation

Mixed floating point sizes

- Uses double

Mixed integer and floating point

- Uses largest floating point in equation

Explicit Type Conversion

Can performing widening and narrowing

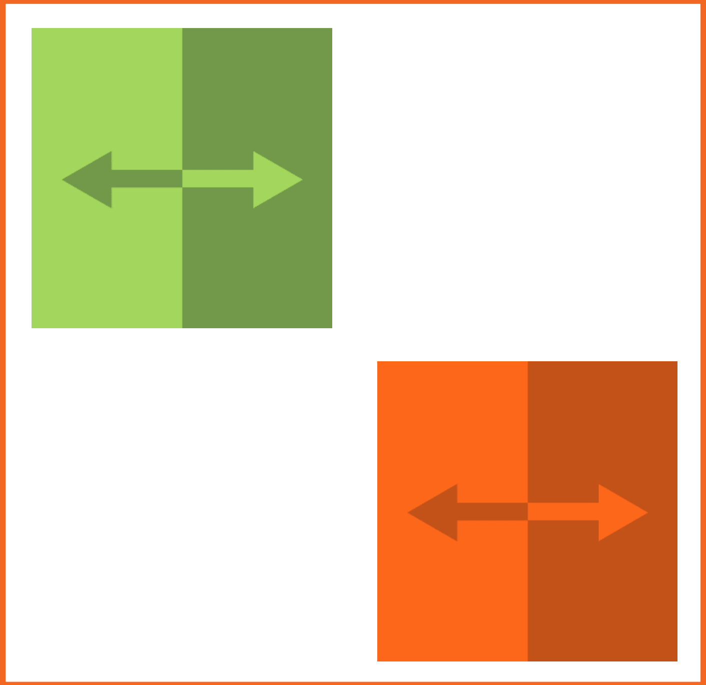
Floating point to integer drops fraction

Use caution with narrowing conversions

Integer to floating point can lose precision

Demo

Type Conversion



Summary

- Variables are strongly typed in Java
- Primitive types
  - Integer types, floating point types, char type, boolean type
- Math operators
  - Basic operators, postfix/prefix operators, compound assignment operators
- Math operators follow a well-defined order of precedence
- Type conversions
  - Compiler can automatically apply widening type conversions
  - Use type casting to explicitly perform type conversions