Variables, Data Types, and Math Operators

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim   jwhh.com

Overview

Variables

Primitive data types

Primitive data type storage

Arithmetic operators

Data type conversions

```
int dataValue;

dataValue = 100;


int myInfo = 200;
```

Variables

Named data storage

Strongly typed

```
int total;

int grade4;

int ~~2much~~;
```

Variable Naming

Use only letters and numbers

First character cannot be a number

```
int sum;

int studentCount;

int bankAccountBalance;

int level2Training;
```

Style Names Using Camel Case

Start each word after the first with upper case

All other letters are lower case

Using Variables

Main.java

```java
int myVar;
myVar = 50;
System.out.println(myVar);
int anotherVar = 100;
System.out.println(anotherVar);
myVar = anotherVar;
System.out.println(myVar);
```

```
50
100
100
```

```java
final int maxStudents = 25;

final int someVariable;

int someOtherVariable = 100;

someVariable = someOtherVariable;
```

Variables Can Be Declared Final

Use final modifier

Value cannot be changed once set

Helps avoid errors caused by inadvertent variable changes

Primitive Data Types

Built into the language

Foundation of all other types

Four categories

- Integer
- Floating point
- Character
- Boolean

```java
byte numberOfEnglishLetters = 26;

short feetInAMile = 5280;

int milesToSun = 92960000;

long milesInALightYear = 5879000000000L;
```

Integer Types

Type	Bits	Min Value	Max Value	Literal Form
byte	8	-128	127	0

```
float kilometersInAMarathon = 42.195f;

float absoluteZeroInCelsius = -273.15f;

double atomWidthInMeters = 0.0000000001d;
```

Floating Point Types

Store values containing a fractional portion

Type	Bits	Smallest Positive Value	Largest Positive Value	Literal Form
float	32	1.4×10^{-45}	3.4×10^{38}	0.0f

```
char regularU = 'U';

char accentedU = '\u00DA'; // Ú
```

Character Type

Stores a single Unicode character

Literal values placed between single quotes

For Unicode code points, use \u followed by 4-digit hex value

```java
boolean iLoveJava = true;
```

Boolean Type

Stores true/false values

Literal values are true and false

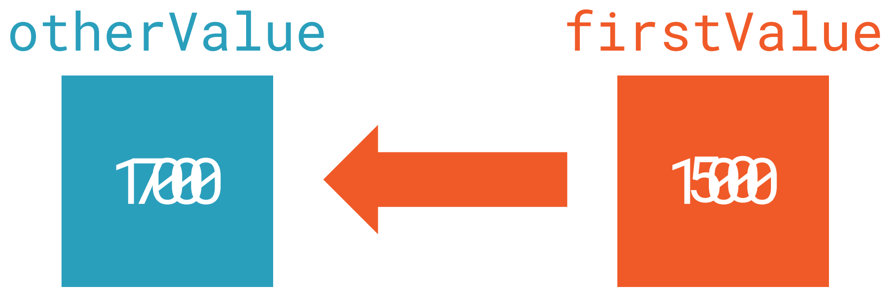
```
int firstValue = 100;

int otherValue = firstValue;

firstValue = 50;

otherValue = 70;
```

Primitive Types Are Stored by Value



Arithmetic Operators

Basic

Produce a result

No impact on values used in the operation

Prefix/postfix

Increase or decrease a value

Replace original value

Compound assignment

Operate on a value

Replace original value

Basic Operators

		Floating Point Example		Integer Example	
Operation	Operator	Equation	Result	Equation	Result
Add	+				
Subtract	-				
Multiply	*				
Divide	/				
Modulus	%				

Prefix and Postfix Operators

++

Increment value by 1

--

Decrement value by 1

Order matters

Prefix applies operation before returning value

Postfix applies operation after returning value

Prefix and Postfix Operators

Order matters

Main.java

```java
int someValue = 5;
System.out.println(++someValue);
System.out.println(someValue);


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

```
6
6


5
6
```

Compound Assignment Operators

Combine an operation and assignment

- Apply right side value to left side
- Store result in variable on left side

Available for 5 basic math operations

- += -= *= /= %=

Compound Assignment Operators

Main.java

```java
int myValue = 50;

myValue -= 5;

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

```
45
```

Compound Assignment Operators

Main.java

```java
int myOtherValue = 100;

int val1 = 5;
                                   50
int val2 = 10
                             ┌─────┴─────┐
myOtherValue /= val1 * val2;

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

```
2
```

Operator Precedence

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

Operator Precedence

Operators of equal precedence evaluated left-to-right

Can override precedence with parenthesis

Nested parenthesis evaluated from inside out

```
int intValueOne = 50;

long longValueOne = intValueOne;


long longValueTwo = 50;

int intValueTwo = (int) longValueTwo;
```

Type Conversion

Implicit type conversion

- Conversion automatically performed by the compiler

Explicit type conversion

- Conversion performed explicitly in code with cast operator

Implicit Type Conversion

Widening conversions are performed automatically

Mixed integer sizes

Uses largest integer in equation

Mixed floating point sizes

Uses double

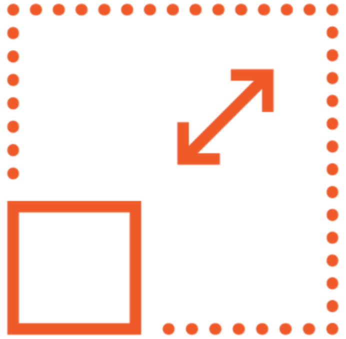
Mixed integer and floating point

Uses largest floating point in equation

Explicit Type Conversion

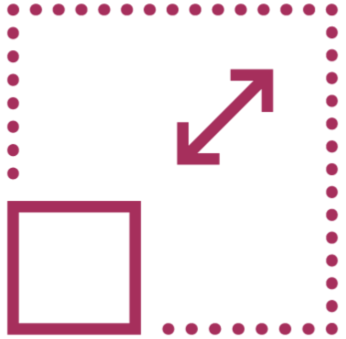
Can perform widening or narrowing conversions

Be aware of potential side-effects



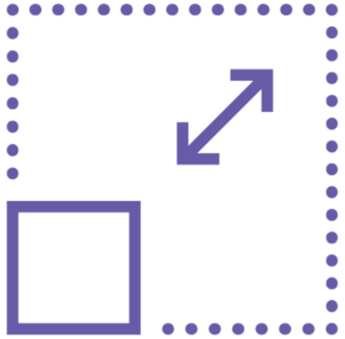
Narrowing conversions

Significant bits may be discarded



Floating point to integer

Fractional portion is discarded



Integer to floating point

Precision may be lost

Summary

Variables

- Strongly typed
- By default variables can be modified
- Mark as final to prevent modification

Primitive types

- Integer types
- Floating point types
- Character type
- Boolean type

Summary

Math operators

- Basic operators
- Postfix/prefix operators
- Compound assignment operators

Math operator precedence

- Well-defined order of precedence
- Evaluated left-to-right when tied
- Can override with parenthesis

Summary

Implicit type conversion

- Widening conversions are performed automatically

Explicit type conversion

- Use cast operator
- Can be widening or narrowing
- Be aware of potential side-effects