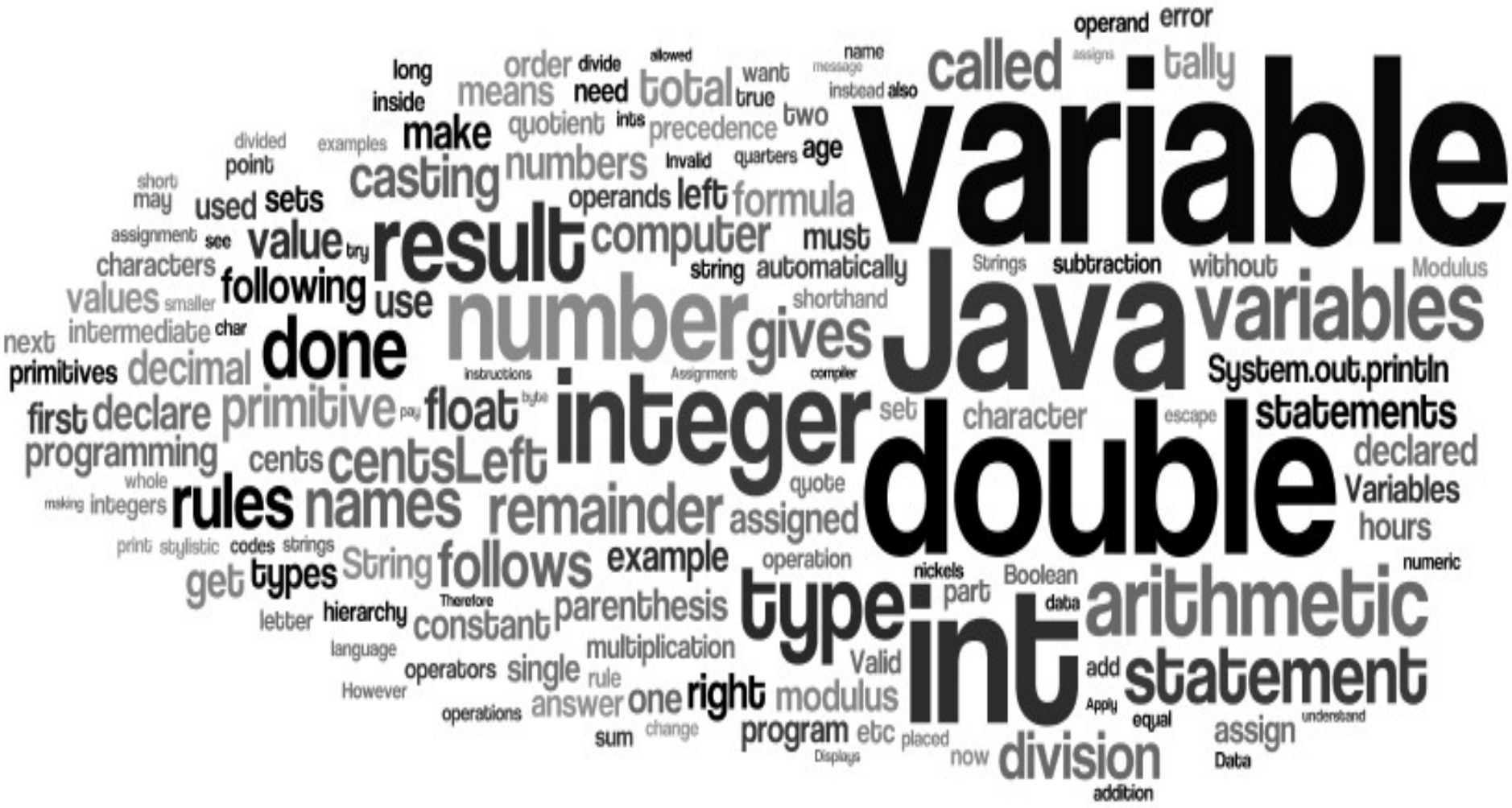
Chapter 2



Variables

Objectives

- Describe the term variable.
- Create valid variable names by following Java's syntactical and stylistic rules.
- Name and describe each of the primitive types used in the Java language.
- Declare and assign values to variables.
- Use comments, strings, and statements properly in coding.
- Describe string concatenation.
- Use escape codes to format output.

Objectives Continued

- List the rules of precedence in order.
- Apply precedence rules to solve arithmetic Java statements.
- Solve arithmetic Java statements that include a mixture of the 5 arithmetic operators.
- Match shorthand assignment operators with their equivalent longhand notations.
- Apply casting rules to coding situations.

Java Variables

- **Constant Data** – cannot be changed after the program compiles
- **Variable Data -** might change after the program compiles
  - **Variables** are locations in memory in which values can be stored.
  - Variables have a type, name, and a value.

Variable Names…

- Consist of letters, digits, underscores (_) or $
- Cannot have the first character as a digit
- Can be any length (extremely short or long names are awkward)
- Cannot be keywords
- Should begin with a lowercase letter (following words should have an initial uppercase letter such as *theButton* or *grossPay*)
- Should be meaningful.

Variable Naming Practice

Variable Name	Valid or Invalid and Why?
grossPay	
Hourly wages	
card-game	
total2	
totalscore	
grand_Total	
java	
dollars$	

Variable Types

Type	Size	Range	Sample
byte (integer)	8 bits	-128 to 127	byte numberOfChildren;
short (integer)	16 bits	-32768 to 32767	short age;
int (integer)	32 bits	-2,147,483,648 to 2,147,483,647	int counter;
long (integer)	64 bits	-9223372036854775808 to 9223372036854775807	long debt;
float (floating point)	32 bits single precision	-3.4028235E+38 to +3.4028235E+38 (7 digits precision)	float rate;
double (floating point)	64 bits double precision	-1.7976931348623157E+308 to +1.7976931348623157E+308 (16 digits precision)	double tax;
char	16 bits unsigned	one character using Unicode system	char answer='Y';
boolean	8 bits	true or false	boolean reply=false;

Declaring Variables

- Must be declared before they can be used.
- Should only be declared once in a program.
- Are usually declared at the beginning of a class.
  - Example: int number;
- You can declare multiple variable names of same type by separating with commas.
  - Example: int x, y, z;

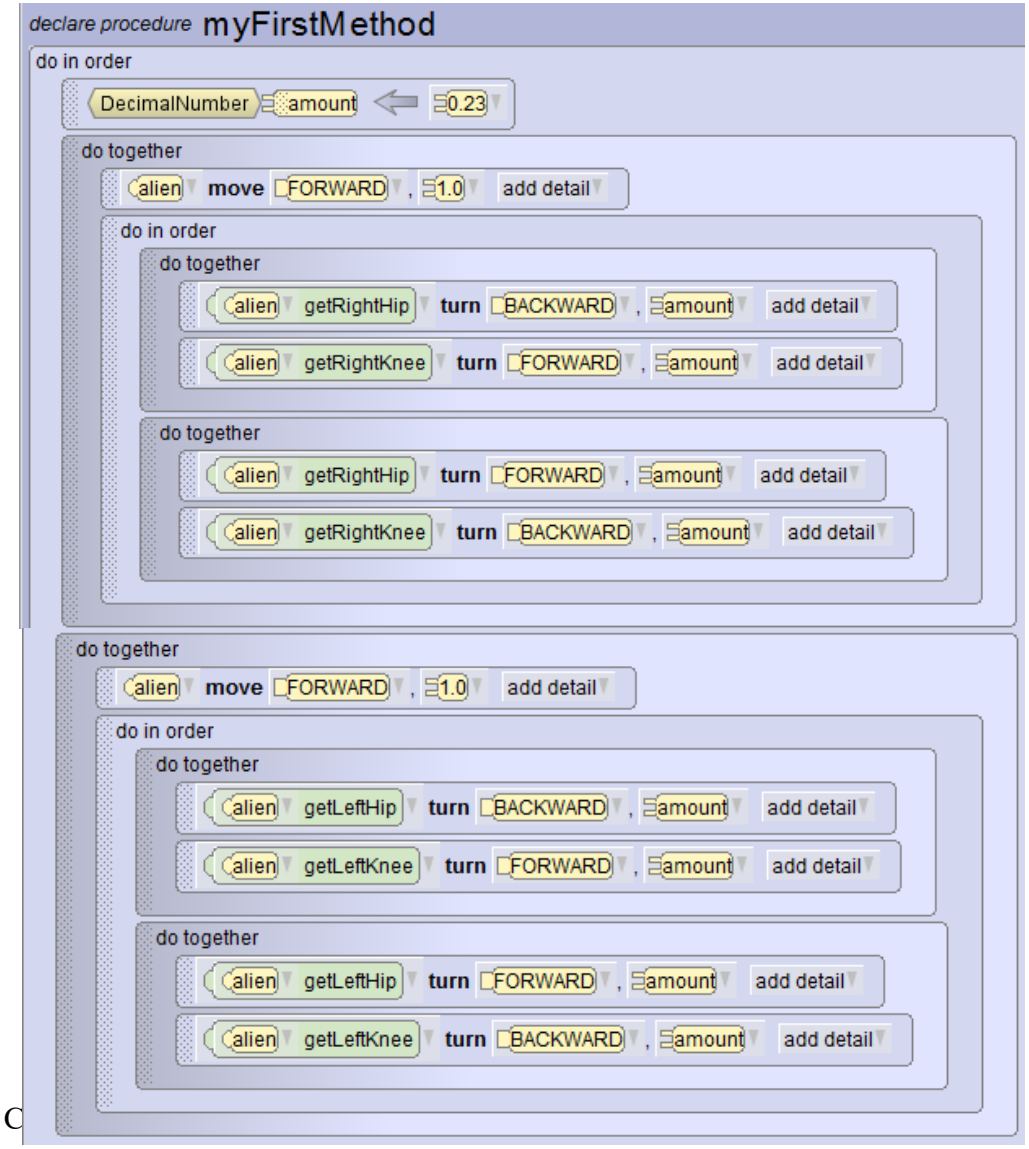
Assigning Values to Variables

- You can declare a variable and initialize it (give it a value) at the same time.
  - Example: int number = 15;
- If you try to assign a number with a decimal point to an integer type variable, you will get an error.
  - Example: int y = 18.5;   // ERROR!!

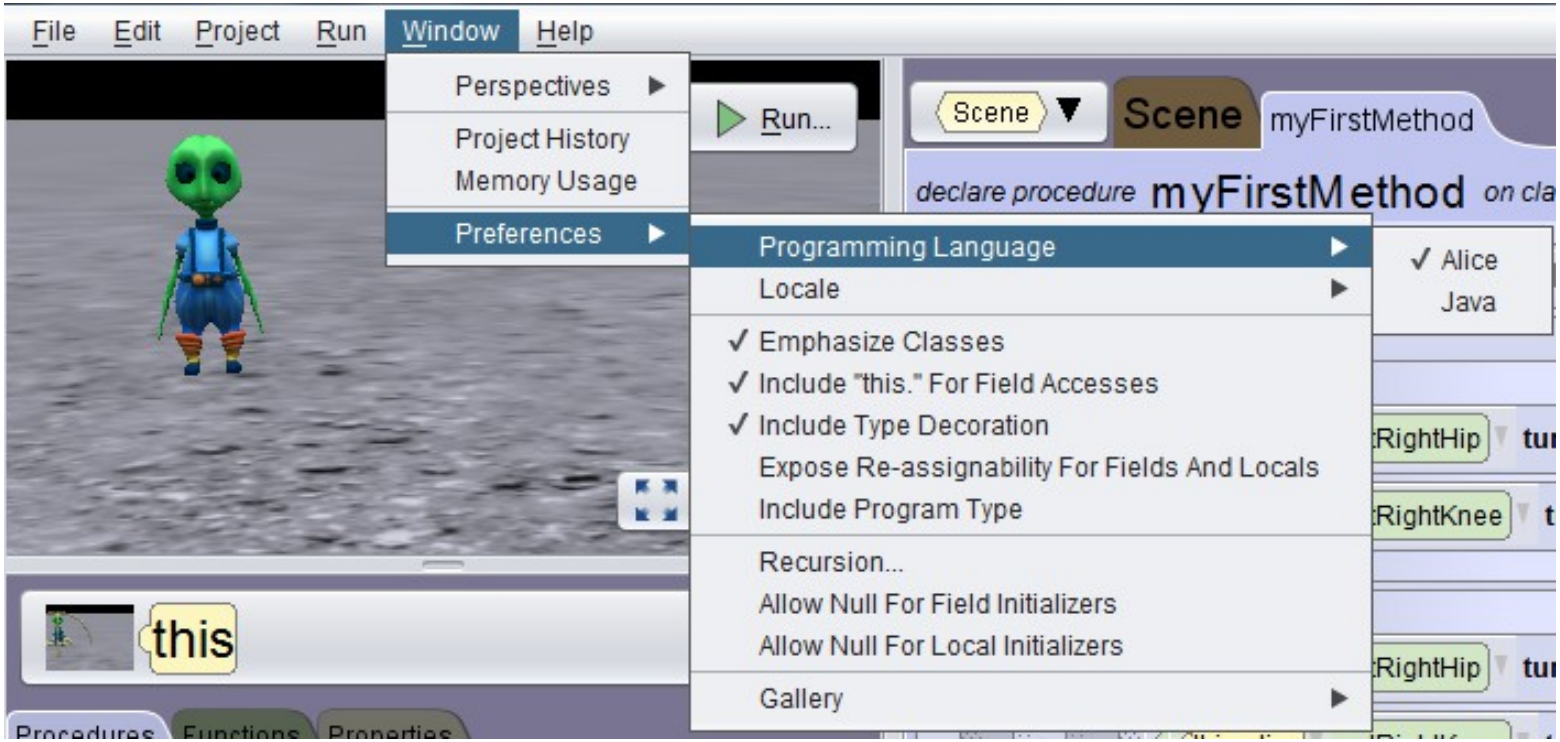
Assigning Values to Variables Continued…

- By default, whole numbers are of type *int* and decimal point numbers are of type *double*.
- If you declare a variable with a type other than int or double, you will need to attach a letter to the numeric constant or you will get an error.
  - Example: If you declare the variable "a" to be a float and want to assign it the value of 100.98 you would need to write it as: float a = 100.98F; OR float a = 100.98f;

Alice Variable Example



Alice Preferences



String Variables

- A **String** is a series of characters that can be used to display messages, input text, etc.
  - May include letters, digits, and/or special characters.
  - A String is an object in Java.
  - String literals are written inside double quotes.
  - Examples :

    "John Doe"

    "219 Grant Street "

Strings Continued…

- String Declaration
  - Examples:

```
String message; // declare a string called message
String message = "Go Cougars!!!!"; // declared and assigned value
```

- String Concatenation
  - The + operator, when used with strings and other objects, creates a single string that contains the concatenation (combining) of all its operands.

```
double total = 50.95;
System.out.println ( "The total is " + total ) ;
Output:  The total is 50.95
```

Precedence Rules

1. Innermost parentheses are done first. If the parentheses are on same level, the formula inside of the leftmost parenthesis is done first.
2. Multiplication (*), division (/), and modulus (%) are done from left to right in order that they are encountered.
3. Addition (+) and subtraction (-) are all on same level and done from left to right in order that they are encountered.
4. Assignment operator (=). Places result in variable on left side of = sign.

Note: Modulus (%) is the remainder of the division.

Practice

Example: answer = 15 /3 + (1 + 2) * 4

3	1 +2 done 1st because of parenthesis (formula is now =15/3+3*4)
5	15/3 done 2nd because of division & leftmost (formula is now =5+3*4)
12	3*4 done 3rd because multiplication is same level (formula is now 5+12)
17	5+12 done 4th because all * / % are done and + is next (formula is 17)
answer =17	last of all 17 is placed in variable answer

Integer Division

- When dividing two integers:
  - the quotient is an integer
  - the remainder is truncated

- Dividing by zero creates a run-time error

Modulus

Equation	Explanation	Answer
13 % 4	13 divided by 4 gives you a quotient of 3 and a remainder of 1	1
6 % 2	6 divided by 2 gives you a quotient of 3 and a remainder of 0	0
int y = 45 % 6 * 5 % 2;	45 divided by 6 gives you a quotient of 7 and remainder of 3. The intermediate result of 45 % 6 is 3 and so now you multiply the 3 by 5 which gives you 15. Now you are left with 15 % 2 which means to divide 15 by 2 giving you a quotient of 7 and remainder of 1.	1

Shorthand Assignment Operators

Shorthand Assignment Operator	Longhand Notation
x + = y;	x = x + y;
x - = y;	x = x - y;
x * = y;	x = x * y;
x / = y;	x = x / y;
x % =y;	x = x % y;
x ++;	x = x + 1;
x --;	x = x – 1;

Casting


```
z  = (int) 31.56;    // puts the integer part which is 31 into z
z  = (int) (x / y);  //  divides x by y and then puts integer into z
```

Walk-Through

```
double y = 25 % 4 + 3 * 5 / 2.0 + (6 / 5);
           25 % 4 + 3 * 5 / 2.0 + 1;
           1 + 3 * 5 / 2.0 + 1;
           1 + 15 / 2.0 + 1;
           1 + 7.5 + 1;
           8.5 + 1;
           9.5;
```