Variables, primitive types, and mathematical expressions

Chris La Pilla
CompSci 6
Summer 2007

Literals
• Previously, we saw the statement:
  System.out.println("Hello, World!");
• "Hello, World!" is a literal
• A literal is a piece of information (i.e. data) that is explicitly given in the code
• All literals have a type that describes the kind of data they hold
• The data itself is the literal’s value

Literals (cont.)
• "Hello, world!" has type String
• Its value is Hello, world!
• A String is a sequence of characters (letters, digits, symbols, etc.)
• println method works with other types as well, e.g.
  System.out.println(10);
• Here, 10 is a literal of type int, an integer

Variables
• Literals are not flexible; their value is fixed unless you change them and recompile
• We want programs to behave more dynamically
• Can do so by using variables
• A variable is a storage location in memory that can hold one of many values

Declarations
• Variables must be declared before they are used, e.g.
  int age = 45;
• int is the variable’s type
• age is the variable’s identifier
• = 45 specifies the initial value for the variable (optional)

Assignment
• Once a value is declared, we can change its value with the assignment operator:
  int age = 45; //declares a new variable age = 46; //assign a new value for age
• The value on the right side is stored in the variable on the left side of the = operator
• Important: the = operator denotes an action, not a mathematical relation; i.e., it means "set the value of age to 46", not "age is equal to 46"
Value of a Variable

- Variables may be used any place a value is expected, e.g.
  ```java
  int age1 = 245;
  int age2 = age1;
  System.out.println(age2);
  ```
- The value currently held by age1 is assigned to age2
- The value currently held by age2 is used for the println method

Variable Types

- Java is a strongly-typed language: every variable must have a known type at compile-time
- Types tell the virtual machine how much memory is needed to store a particular value
- Also allow the compiler to enforce that values aren’t assigned to variables of the wrong type:
  ```java
  String name = "Jim";
  int age = name; //type mismatch error
  ```

Mathematical Expressions

- For now, we’ll consider only ints and doubles
- Consider the statement: `age = 30 + 15;`
- In above statement, `30 + 15` is an expression
- Like literals and variables, expressions have both a type and a value
- Here, the type is int, and the value is the result of the + operator, which returns the sum of the two operands (30 and 15)
Mathematical Expressions (cont.)

• Some mathematical operators for ints and doubles are +, -, *, / for addition, subtraction, multiplication, division
• - can be used as a unary (negation) or binary (subtraction) operator:
  oppositeScore = -score;
  change = paid - total;

Expression types

• The type of an expression depends on the types of the operands
• If both operands are ints, the result is an int
• If one or both operands are doubles, the result is a double
• You can assign an int value to a variable of type double, but not the other way around:
  double age = 10 + 35; //OK
  int age = 10.0 + 35; //type mismatch error

Division

• / operator is used for division
• There is a difference between integer division and floating-point division
• If both operands are integers, / returns only the whole-number part of the quotient:
  System.out.println(129 / 10); //prints 12
  System.out.println(129.0 / 10); //prints 12.9:
• Forgetting this is a common bug!

Modulo Operator (%)

• The modulo operator % performs integer division and returns the remainder:
  System.out.println(129 / 10) //prints 12
  System.out.print(129 % 10) //prints 9

Shortcuts

• Increment/decrement operators:
  a++; //same as a = a + 1;
  b--; //same as b = b - 1;
• Can also use shorthand to combine assignment and arithmetic:
  a += 5; //same as a = a + 5;
  Works for other operators (-, *, /, %) as well

Order of operations

• Multiple operators can be chained together:
  System.out.println(5 + 4 * 3); //prints 17
• Like algebra, operators are applied in a certain order:
  Unary -
  *, /, % (from left to right)
  +, - (from left to right)
• Can use parentheses to force precedence:
  System.out.println((5 + 4) * 3); //prints 27
Summary

• Literals, variables, and expressions all have a value and a type
• They can be used wherever a value is expected, e.g.
  – right hand side of an assignment statement
  – argument of a method call
• The type and result of mathematical operators depend on the types of the operands