Numbers

January 12, 2004

Outline

- Variables
- Variable Declaration
- Primitive Variables
- Constants
- Arithmetic Operations
- Real Number Arithmetic
- Arithmetic Expressions
- Assignment (of values to a data type)
- Casting and Rounding

© 2004 Sarah E. Smith
Variables

• Holds a value
• Should have a default value, but this is optional
• Primitive variables only hold one value
• Primitive variables are predefined in the language
• Object variables can hold multiple values and have actions

Variable Declaration

• Syntax:<data type> <variable> [= <default value>];
• Type - how much memory to reserve for use
• Identifier - name of the variable, programmer chosen
  – Starts with a lowercase letter
  – Use uppercase letters to separate words
  – Ex: bodyMassIndex
• Initial Value - optional
• Ex: int bodyMassIndex = 0;
### Primitive Variables - Numeric

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Content</th>
<th>Default Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Integer Number</td>
<td>0</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>Integer Number</td>
<td>0</td>
<td>-32768</td>
<td>32767</td>
</tr>
<tr>
<td>int</td>
<td>Integer Number</td>
<td>0</td>
<td>-2147483648</td>
<td>2147483647</td>
</tr>
<tr>
<td>long</td>
<td>Integer Number</td>
<td>0</td>
<td>-9223372036854775808</td>
<td>9223372036854775807</td>
</tr>
<tr>
<td>float</td>
<td>Real Number</td>
<td>0.0</td>
<td>-3.40282347E+38</td>
<td>3.40282347E+38</td>
</tr>
<tr>
<td>double</td>
<td>Real Number</td>
<td>0.0</td>
<td>-1.7976931348623157E+308</td>
<td>1.7976931348623157E+308</td>
</tr>
</tbody>
</table>

© 2004 Sarah E. Smith

---

### Primitive Values – Non-numeric

- boolean – either true or false
- Ex: boolean is21 = true;
Constants

- Values cannot be changed after they have been assigned
- Keyword is `final`
- Named or symbolic constants – declared like a variable with the keyword `final`
- Literal constants – use the actual value
  - Literal constants default to `int` and `double` data types
  - Use L or l (lowercase L) and F or f at the end of a constant to make long (integer) or float (real)
  - May also use D or d to make a real constant a double

---

Arithmetic Operations

| Operation                  | Java Operator | Example | Value  
|----------------------------|---------------|---------|--------
| Addition                   | +             | x + y   | 17     
| Subtraction                | -             | x – y   | 3      
| Multiplication             | *             | x * y   | 70     
| Division (Integer)         | /             | x / y   | 1      
| Division (Real)            | /             | x / z   | 4.0    
| Modulo Division (remainder on integer division) | %             | x % y   | 3      

© 2004 Sarah E. Smith
Arithmetic Operations (2)

- Exponential numbers written with E notation
  - \textit{number} \times 10^{\text{exponent}} = \langle\text{number}\rangle E\langle\text{exponent}\rangle
- May also use \textit{e}
- Sign on exponent optional for positive numbers
- Always a double or float (even if no decimal in the number) Ex: 22E33
- May specify with D, d, F, or f at the end of the number

Arithmetic Operations (3)

- Other math operations can be found in the \textit{java.lang.Math} library
- Square root – double \texttt{Math.sqrt(double a)}
- Power – double \texttt{Math.pow(double a, double b)}
- Round float – int \texttt{Math.round(float a)}
- Round double – long \texttt{Math.round(double a)}
Real Number Arithmetic

• Real or floating point numbers are not precise

Example:

\[.1 + .1 + .1 + .1 + .1 + .1 + .1 + .1 + .1 + .1 
eq 1.0\]

Arithmetic Expressions

• Use parentheses to declare order of operations
  • \[3 + 4 \times 5 = ?\]
  • \[(3 + 4) \times 5 = 35\]
  • \[3 + (4 \times 5) = 23\]
• Default is to follow precedence rules
• Left to right associativity
Assignment

- Syntax: `<variable> = <value>`
- Identifiers on right and left side of `=`
- Identifier on right specifies a value
- Identifier on left specifies location to store result
- Ex: `int result = 5;`
- Ex: `result = 7;`
- Ex: `int newResult = 5 * result;`

Casting

- “Converts the value of one data type to another data type” [Wu]
- Implicit - numeric promotion
- Explicit - use a type cast operator to convert
Numeric Promotion

- If operations are performed on the same type of data then the result will be the type involved in the operation
  Ex: int result = 2 + 3;
- If you perform an operation on two different data types the result is promoted to the data type with the higher precision (more space)
  Ex: double result = 2 + 3.0;
- More explicit rules are given in Table 3.4 (p96) in the book.

Assignment Conversion

- Narrowing
  int dollars;
  dollars = 20.50; //No! (lose precision)
- Widening
  double area;
  area = 20; //Yes! (gain precision)
- “An assignment conversion only occurs when the data type of the variable has a higher precision than the data type of the expression’s value.” [Wu]
Explicit Casting

- Syntax: ( <data type> ) <expression>
- The data type in the ()s is the type cast operator - this is the data type that you want to change the expression into
- Type cast operator is a unary operator
- Ex:
  ```java
  int x = 10;
double y = (double)x;  // y = 10.0
  ```

Explicit Casting (2)

- When casting a value from a type of higher precision to a type of lower precision, the decimal is lost
- Ex:
  ```java
  double x = 10.6;
  int y = (int)x;  // x = 10
  ```
- Casting must occur in instances where there is a loss of precision

© 2004 Sarah E. Smith
Rounding

• Rounding must be done if you want to have the decimal portion of the number affect the new value.
• Ex:
  double x = 10.6;
  long y = Math.round(x);    // y = 11

References

• java.lang.Math library in Java API:
  http://java.sun.com/j2se/1.4.2/docs/api/