Data Conversion & Scanner Class
August 30, 2006

Quick review of last lecture

Numeric Primitive Data
• The difference between the various numeric primitive types is their size, and therefore the values they can store:

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 x 10^18</td>
<td>&gt; 9 x 10^18</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^38 with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 10^308 with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>

Storing Information
• Each memory cell stores a set number of bits (usually 8 bits, or one byte)
• Large values are stored in consecutive memory locations

Storing a short
• short (16 bits = 2 bytes)

Storing a double
• double (64 bits = 8 bytes)
### Operator Precedence

<table>
<thead>
<tr>
<th>Precedence Level</th>
<th>Operator</th>
<th>Operation</th>
<th>Associates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>unary plus</td>
<td>R to L</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>unary minus</td>
<td>R to L</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>multiplication</td>
<td>L to R</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>division</td>
<td>L to R</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>remainder</td>
<td>L to R</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
<td>addition</td>
<td>R to L</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>subtraction</td>
<td>R to L</td>
</tr>
<tr>
<td></td>
<td></td>
<td>string concatenation</td>
<td>R to L</td>
</tr>
<tr>
<td>4</td>
<td>=</td>
<td>assignment</td>
<td>R to L</td>
</tr>
</tbody>
</table>

### Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators.
- First the expression on the right hand side of the `=` operator is evaluated.
- Then the result is stored in the variable on the left hand side.

```
answer = sum / 4 + MAX * lowest;
```

### Other material from Sec 2.4

- The right and left hand sides of an assignment statement can contain the same variable.
- First, one is added to the original value of `count`.
- Then the result is stored back into `count` (overwriting the original value).

```
count = count + 1;
```

### Increment and Decrement

- The increment and decrement operators use only one operand.
- The increment operator (`++`) adds one to its operand.
- The decrement operator (`--`) subtracts one from its operand.
- The statement `count++;` is functionally equivalent to `count = count + 1;`.
- The increment and decrement operators can be applied in postfix form:
  ```
  count++
  ```
  or prefix form:
  ```
  ++count
  ```
- When used as part of a larger expression, the two forms can have different effects.
- Because of their subtleties, the increment and decrement operators should be used with care.
Assignment Operators

• Often we perform an operation on a variable, and then store the result back into that variable.
• Java provides assignment operators to simplify that process.
• For example, the statement
  \[ \text{num } += \text{ count}; \]
  is equivalent to
  \[ \text{num } = \text{num } + \text{ count}; \]

Assignment Operators

• The right hand side of an assignment operator can be a complex expression.
• The entire right-hand expression is evaluated first, then the result is combined with the original variable.
• Therefore
  \[ \text{result } /= (\text{total-MIN} ) \% \text{ num}; \]
  is equivalent to
  \[ \text{result } = \text{result } / ((\text{total-MIN} ) \% \text{ num}); \]
Conversion Techniques

• 1) Assignment conversion
  • Value of one type is assigned to a variable of another
type during which the value is converted to the new
type.

• 2) Promotion
  • Occurs automatically when certain operators need to
modify their operands.

• 3) Casting (a.k.a. type casting)
  • Specified explicitly by the programmer

Assignment conversion

float money;
int dollars;
dollars=5;
money = dollars; // OK, money is now equal to 5.0
dollars= money; // Compile error

(automatic) promotion

float sum, result;
int count;
sum= 12.0;
count=5;
result = sum/count; // count promoted to float
// before the division

(automatic) promotion

// the number ‘5’ is first promoted to a string and then
// the two strings are concatenated
System.out.println("Five is equal to "+ 5);
Type Casting

- The programmer explicitly asks the compiler to change the type of a variable or a temporary result before the next operation will take place.
- Without the cast Java typically will refuse to compile the program.

```
float money;
int dollars;
dollars=5;
money = dollars; // OK, money is now equal to 5.0
dollars= (int) money; //Compile error
```

Type Casting + Promotion

```
float result;
int total, count;
total= 12;
count=5;
result = (float) total / count;  // result = 2.4
// 1. total is cast to float
// 2. count is promoted to float
// 3. the division is performed
```

Interactive Programs

- Programs generally need input on which to operate
- The Scanner class provides convenient methods for reading input values of various types
- A Scanner object can be set up to read input from various sources, including the user typing values on the keyboard
- Keyboard input is represented by the System.in object

Casting Example

Reading Input

- The following line creates a Scanner object that reads from the keyboard:
  
  ```java
  Scanner scan = new Scanner (System.in);
  ```

- The `new` operator creates the `Scanner` object
- Once created, the `Scanner` object can be used to invoke various input methods, such as:
  
  ```java
  answer = scan.nextLine();
  ```

- In order to use the `Scanner` object you must put this line at the top of your Java program
  
  ```java
  import java.util.Scanner;
  ```

Input Tokens

- Unless specified otherwise, **white space** is used to separate the elements (called **tokens**) of the input
- White space includes space characters, tabs, newline characters
- The `next` method of the `Scanner` class reads the next input token and returns it as a string
- Methods such as `nextInt` and `nextDouble` read data of particular types

**See GasMileage.java** (page 92)

Scanner Class

- The `Scanner` class is part of the `java.util` class library, and must be imported into a program to be used
- See `Echo.java` (page 91)
- The `nextLine` method reads all of the input until the end of the line is found
- The details of object creation and class libraries are discussed further in Chapter 3

More Scanner Examples

THE END