Primitive Data Types

August 27, 2007

String Concatenation

- The string concatenation operator (+) is used to append one string to the end of another
  "Peanut butter " + " and jelly"
- It can also be used to append a number to a string
- A string literal cannot be broken across two lines in a program
- See Facts.java (page 65)

Escape Sequences

- What if we wanted to print a the quote character?
- The following line would confuse the compiler because it would interpret the second quote as the end of the string
  System.out.println ("I said "Hello" to you.");
- An escape sequence is a series of characters that represents a special character
  An escape sequence begins with a backslash character (\)
  System.out.println ("I said \"Hello\" to you.");
- See Roses.java (page 68)
Variables

- A variable is a name for a location in memory
- A variable must be declared by specifying the variable's name and the type of information that it will hold

```java
int total;
int count, temp, result;
```

Multiple variables can be created in one declaration

Assignment

- An assignment statement changes the value of a variable
- The assignment operator is the `=` sign

```java
total = 55;
```

- The expression on the right is evaluated and the result is stored in the variable on the left
- The value that was in `total` is overwritten
- You can only assign a value to a variable that is consistent with the variable's declared type
- See Geometry.java (page 71)

Constants

- A constant is an identifier that is similar to a variable except that it holds the same value during its entire existence
- As the name implies, it is constant, not variable
- The compiler will issue an error if you try to change the value of a constant
- In Java, we use the `final` modifier to declare a constant

```java
final int MIN_HEIGHT = 69;
```

Constants

- Constants are useful for three important reasons
- First, they give meaning to otherwise unclear literal values
  - For example, MAX_LOAD means more than the literal 250
- Second, they facilitate program maintenance
  - If a constant is used in multiple places, its value need only be updated in one place
- Third, they formally establish that a value should not change, avoiding inadvertent errors by other programmers

Primitive Data

- There are eight primitive data types in Java
- Four of them represent integers:
  - byte, short, int, long
- Two of them represent floating point numbers:
  - float, double
- One of them represents characters:
  - char
- And one of them represents boolean values:
  - boolean
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 10^15</td>
<td>&gt; 9 x 10^15</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>

Computer Memory

- Main memory is divided into many memory locations (or cells)
- Each memory cell has a numeric address, which uniquely identifies it

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 byte

- byte (8 bits = 1 byte)

Storing a short

- short (16 bits = 2 bytes)

Storing an int

- int (32 bits = 4 bytes)
Storing a long

\[ \begin{align*}
9278 & \quad \text{(long)} \\
9279 & \\
9280 & \\
9281 & \\
9282 & \\
9283 & \\
9284 & \\
9285 & \\
9286 & \\
\end{align*} \]

(64 bits = 8 bytes)

Storing a float

\[ \begin{align*}
9278 & \quad \text{(float)} \\
9279 & \\
9280 & \\
9281 & \\
9282 & \\
9283 & \\
9284 & \\
9285 & \\
9286 & \\
\end{align*} \]

(32 bits = 4 bytes)

Storing a double

\[ \begin{align*}
9278 & \quad \text{(double)} \\
9279 & \\
9280 & \\
9281 & \\
9282 & \\
9283 & \\
9284 & \\
9285 & \\
9286 & \\
\end{align*} \]

(64 bits = 8 bytes)

Characters

- A char variable stores a single character
- Character literals are delimited by single quotes: 
  - 'a' 'X' '7' '$' ',' '
'
- Example declarations:
  ```java
  char topGrade = 'A';
  char terminator = ';', separator = ' ';  
  ```
- Note the distinction between a primitive character variable, which holds only one character, and a String object, which can hold multiple characters

Character Sets

- A character set is an ordered list of characters, with each character corresponding to a unique number
- A char variable in Java can store any character from the Unicode character set
- The Unicode character set uses sixteen bits per character, allowing for 65,536 unique characters
- It is an international character set, containing symbols and characters from many world languages

Storing a char

\[ \begin{align*}
9278 & \quad \text{(char)} \\
9279 & \\
9280 & \\
9281 & \\
9282 & \\
9283 & \\
9284 & \\
9285 & \\
9286 & \\
\end{align*} \]

(16 bits = 2 bytes)
Characters

- The ASCII character set is older and smaller than Unicode, but is still quite popular
- The ASCII characters are a subset of the Unicode character set, including:
  - uppercase letters A, B, C, ...
  - lowercase letters a, b, c, ...
  - punctuation period, semicolon, ...
  - digits 0, 1, 2, ...
  - special symbols &, |, 
  - control characters carriage return, tab, ...

Extended ASCII Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>$</td>
<td>144</td>
<td>£</td>
</tr>
<tr>
<td>129</td>
<td>‹</td>
<td>145</td>
<td>»</td>
</tr>
<tr>
<td>130</td>
<td>¬</td>
<td>146</td>
<td>°</td>
</tr>
<tr>
<td>131</td>
<td>'</td>
<td>147</td>
<td>′</td>
</tr>
<tr>
<td>132</td>
<td>(</td>
<td>148</td>
<td>)</td>
</tr>
<tr>
<td>133</td>
<td>[</td>
<td>149</td>
<td>]</td>
</tr>
<tr>
<td>134</td>
<td>{</td>
<td>150</td>
<td>}</td>
</tr>
<tr>
<td>135</td>
<td>^</td>
<td>151</td>
<td>_</td>
</tr>
<tr>
<td>136</td>
<td>`</td>
<td>152</td>
<td>~</td>
</tr>
<tr>
<td>137</td>
<td>u</td>
<td>153</td>
<td>v</td>
</tr>
<tr>
<td>138</td>
<td>w</td>
<td>154</td>
<td>x</td>
</tr>
<tr>
<td>139</td>
<td>y</td>
<td>155</td>
<td>z</td>
</tr>
<tr>
<td>140</td>
<td>{</td>
<td>156</td>
<td>}</td>
</tr>
<tr>
<td>141</td>
<td>[</td>
<td>157</td>
<td>]</td>
</tr>
</tbody>
</table>

The Unicode Character Code

- [http://www.unicode.org/charts/](http://www.unicode.org/charts/)

Boolean

- A boolean value represents a true or false condition
- The reserved words `true` and `false` are the only valid values for a boolean type
  ```java
do = false;
```
- A boolean variable can also be used to represent any two states, such as a light bulb being on or off

Run alphabet examples
Expressions

- An expression is a combination of one or more operators and operands
- Arithmetic expressions compute numeric results and make use of the arithmetic operators:
  - Addition (+)
  - Subtraction (-)
  - Multiplication (*)
  - Division (/)
  - Remainder (%)
- If either or both operands used by an arithmetic operator are floating point, then the result is a floating point

Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)
  - 14 / 3 equals 4
  - 8 / 12 equals 0
- The remainder operator (%) returns the remainder after dividing the second operand into the first
  - 14 % 3 equals 2
  - 8 % 12 equals 8

Operator Precedence

- Operators can be combined into complex expressions
  - result = total + count / max - offset;
- Operators have a well-defined precedence which determines the order in which they are evaluated
  - Multiplication, division, and remainder are evaluated prior to addition, subtraction, and string concatenation
- Arithmetic operators with the same precedence are evaluated from left to right, but parentheses can be used to force the evaluation order

Expression Trees

- The evaluation of a particular expression can be shown using an expression tree
- The operators lower in the tree have higher precedence for that expression
  - a + (b - c) / d

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;