Outline

° Problem: How do I define and run a Java program?
  • What does the program do with my text?
  • What makes Java different?

° Representing computation with objects

° Representing data
  • Characters
  • Integers
  • Floating point

° Making assignments to variables
Program Development

° The mechanics of developing a program include several activities
  • Writing the program in a specific programming language (such as Java)
  • Translating the program into a form that the computer can execute
  • Investigating and fixing various types of errors that can occur
° Software tools can be used to help with all parts of this process

Translation of Source Code

° The translation of source code into machine language
° Subsequent execution on a particular type of CPU can occur in a variety of ways.
  • By using a compiler.
  • By using an interpreter.
  • By using both a compiler and interpreter
    - This is how Java does it.
A Compiler

- A compiler is a program that translates code from one language to an equivalent code in another language.
- The original code is called source-code.
  - The language into which it is translated is called the target language.
  - For many traditional compilers, the source code is translated directly into a particular machine language.
  - The translation process occurs once and the resulting executable program can be run whenever needed.

Java Programming Language

- Each type of CPU executes only a particular machine language
- A program must be translated into machine language before it can be executed
- A compiler is a software tool which translates source code into a specific target language
- Often, that target language is the machine language for a particular CPU type
An Interpreter

- An interpreter is similar to a compiler but has important differences...
  - An interpreter performs the translation and execution activities.
    - A small part of the source code, such as one statement, is translated and executed.
    - Then another part is translated and executed, and so on.
  - The program runs more slowly because the translation process occurs during each execution.
    - Each statement is translated, then executed immediately.

Java Translation

You write this

- You write Java code.
- Java compiler translates Java code into bytecode.
- Java bytecode is the intermediate representation.

DrJava does this

- DrJava translates bytecode into machine code.
- Your computer reads the bytecode and "executes" the program.
Java Translation

- The Java compiler translates Java source code into a special representation called bytecode.
- Java bytecode is not the machine language for any traditional CPU.
- Another software tool, called an interpreter, translates bytecode into machine language and executes it.
- Therefore the Java compiler is not tied to any particular machine.
- Java is considered to be neutral.

Java Program Structure

- A program is made up of one or more classes.
- A class contains one or more methods.
- A method contains program statements.
- A Java application always executes the main method.
Classes

- An object is defined by a class
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class

Objects and Classes

A class (the concept)  An object (the realization)

Bank Account

John’s Bank Account  Balance: $5,257
Bill’s Bank Account  Balance: $1,245,069
Mary’s Bank Account  Balance: $16,833

Multiple objects from the same class
Character Strings

- A string of characters can be represented as a string literal by putting double quotes around the text:

- Examples:
  - "This is a string literal."
  - "123 Main Street"
  - "X"

- Every character string is an object in Java, defined by the String class

- Every string literal represents a String object

  There are many String objects but only one String class

Character Strings

- Every character string is an object in Java, defined by the String class

- Every string literal, delimited by double quotation marks, represents a String object

- The string concatenation operator (+) is used to append one string to the end of another

- It can also be used to append a number to a string

- A string literal cannot be broken across two lines in a program
Using Classes

° We invoke the `println` method to print a character string
° The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println("Hello, World");
```

What about the “print” method?

---

String Concatenation

° The `+` operator is also used for arithmetic addition
° The function that it performs depends on the type of the information on which it operates
° If both operands are strings, or if one is a string and one is a number, it performs string concatenation
° If both operands are numeric, it adds them
° The `+` operator is evaluated left to right, but parentheses can be used to force the order

```
System.out.println("24 and 45 concatenated: " + 24 + 45);
System.out.println("24 and 45 added: " + (24 + 45));
```
Example

```java
//********************************************************************
// Facts.java  Author: Lewis/Loftus
// Demonstrates the use of the string concatenation operator and the
// automatic conversion of an integer to a string.
//********************************************************************

public class Facts
{
     //-----------------------------------------------------------------
     // Prints various facts.
     //-----------------------------------------------------------------
     public static void main (String[] args)
     {
          // Strings can be concatenated into one long string
          System.out.println ("We present the following facts for your "
                          + "extracurricular edification:");
          System.out.println ();
          // A string can contain numeric digits
          System.out.println ("Letters in the Hawaiian alphabet: 12");
          // A numeric value can be concatenated to a string
          System.out.println ("Dialing code for Antarctica: " + 672);
          System.out.println ("Year in which Leonardo da Vinci invented "
                          + "the parachute: " + 1515);
          System.out.println ("Speed of ketchup: " + 40 + " km per year");
     }
}
```

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
  ```java
  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 (\)
  ```java
  System.out.println ("I said \"Hello\" to you.");
  ```
Escape Sequences

° Some Java escape sequences:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>backspace</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>\r</td>
<td>carriage return</td>
</tr>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>

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
**Variables**

- A variable can be given an initial value in the declaration

  ```
  int sum = 0;
  int base = 32, max = 149; // note syntax...
  ```

- When a variable is referenced in a program, its current value is used

**Assignment**

- An *assignment statement* changes the value of a variable

  ```
  total = 55;
  ```

- The *assignment operator* is the `=` sign

- 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
Constants

- A constant is an identifier that holds one value while the program is active.
- The compiler will issue an error if you try to change the value of a constant during execution.
- In Java, we use the final modifier to declare a constant.
  
  ```java
  final int MIN_HEIGHT = 69;
  ```

  Note: Constants are written in caps to distinguish themselves from other 'variables' whose values can change.
  - give names to otherwise unclear literal values
  - facilitates updates of values used throughout a program
  - prevent inadvertent attempts to change a value
  - (Discuss: final float RATE = 0.15; only change value...)

Primitive Data

- There are exactly eight primitive data types in Java.
- Four represent integers:
  - byte, short, int, long (no fractions)
- Two represent floating point numbers:
  - float, double (contain decimals)
- One represents characters: char
- One represents boolean values: boolean
- All have different 'sizes' and 'ranges'...
Numeric Primitive Data

° Sizes and Ranges of storable values below.
° Use size as ‘appropriate’ but if in doubt, be generous.

<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>

Numeric Primitive Data

° Default: int is 32 bits; but 45L or 45l => long

° Default: for decimal data:
  • assumes all literals are type double.
  • To make ‘float’ ➔ 45.6F or 45.6f
  • Can say, if desired, 45.6D or 45.6d, but unnecessary.
Characters

° A char variable stores a single character from the Unicode character set
° A character set is an ordered list of characters, and each character corresponds to a unique number
° 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
° Character literals are delimited by single quotes:

' a ' ' X ' ' 7 ' ' $ ' ',' ' \n '

⇒ ‘7’ is not equivalent to 7 is not equivalent to “7”

Characters

° The ASCII character set is older and smaller than Unicode, but is still quite popular
  • Has evolved to eight-bits per byte.
  • (char is a ‘primitive data type’; String is a class)
  • Because String is a class, it has many methods (operations) that can be performed on String objects!
° The ASCII characters are a subset of the Unicode character set, including:

  uppercase letters    A, B, C, ...
  lowercase letters    a, b, c, ...
  punctuation          period, semi-colon, ...
  digits               0, 1, 2, ...
  special symbols      & , |, \, ...
  control characters   carriage return, tab, ...
**Boolean**

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

```java
boolean done = false;
```
Logical Errors

- When your program has a **logical error**, it will compile and execute, but produces incorrect results. (These are runtime errors too!)
  - A **logical error** occurs when a value is calculated incorrectly.
  - A programmer must **test the program thoroughly**, comparing the expected results to those that actually occur.
    - The process of finding and correcting defects in a program is called **debugging**.

Basic Program Development

Edit and save program → Compile program → Execute program and evaluate results

Errors → Edit and save program
Good Programming Practice

- Write a comment before each class, documenting the purpose of the class.
- Write end of line (single line) comment, documenting the purpose of the statement.
- Use DrJava to check your syntax.
- Declare each variable in each line, allowing end of line comment. E.g.
  
  ```java
  int age; // The age of my dog
  ```
- Choose meaningful variable names helps a program to be self-documenting. Easy to understand.

Summary

- Java code is primarily interpreted so it can be used on any computer
- Java is based on objects which contain information (state) and actions (methods)
- Strings form an important part of data representations
- Assignments and basic data types allow for data storage and manipulation
- Reading: L+L: 1.5-1.6, 2.1-2.3