Java Applications for C/C++ Programmers

Paul I-Hai Lin

Purdue University Fort Wayne Campus

January 15, 2002

Ch 1. Java Programming Language Basics

1.1 Programming Languages

1.2 Java Programming Language Basics

- Java Features and Architecture
- Java Components
- Java Applications
- Java Class Packages

1.3 Java Programming Environment and Tools

- Java Development Tools and Web Sites

1.4 Java versus C++

1.5 Stand-alone Java programs

1.6 Java Programs (Applets and Servlets) for Internet-based Applications

1.7 Getting Started with Java

1.8 Java Programming Examples

1.1 Programming Languages

Programming languages are selected based on a set of commonly accepted criteria:

- Portability
- Reliability
- Maintainability
- Productivity
- Programming methodology
- Performance issues (compiler, interpreter)
- New technology support
- Standardization concern

Examples of high-level general purpose programming languages may include:

- C - developed and implemented by Dennis Ritchie's group at Bell Laboratories in 1973 for writing UNIX operating system, software, and compilers. C is a hardware-independent language for developing various applications with great portability for various target machines ranged from microcontrollers, digital signal processors, personal computer, mainframe computers, etc.
- C++ - an object-oriented programming (OOP) language developed by Bjarne Stroustrup in the early 1980s at Bell laboratories as an extension to the C programming language
- Java - a programming language developed by Sun since 1995 for creating Web pages with dynamic content, large-scale enterprise applications, and enhancing Web server capabilities
- Microsoft VB.NET - a new programming model and environment for developing distributed and Web-based applications, released in 2000.
- C# (C-sharp) and .NET (dot-net) - a programming model and Internet environment developed in 2000 by Microsoft for creating Web-based application. It is an object-oriented language containing best features of C, C++ and Java, and class library to enable rapid application development.

Some programming languages as listed below are commonly used along with C/C++, Java, and/or Visual Basic in many modern distributed applications:

- HTML (Hypertext MarkUp Language) - for creating Web page
- XML (eXtensible Markup Language) - for creating user defined tagged documents
- JavaScript/JScript - client side scripting on Web pages
- VBScript - client side/server side scripting of Web applications
- Perl - Practical Extraction Report Generation Language (server-side)
- ASP (Active Server Page) - for server-side scripting under Microsoft Internet Information server
- JSP (Java Server Page) - for server-side scripting under Java environment

1.2 Java Programming Language Basics

The first paper about Java is entitled “The Java Language Environment: A White Paper,” by James Gosling and Henry McGilton, http://java.sun.com/docs/white/langenv/, May 1996. It highlights the following design goals:

- Simple, Object Oriented, and Familiar
- Distributed, Robust and Secure
- Architecture Neutral and Portable
- High Performance
- Interpreted, Threaded, and Dynamic

Java Features and Architecture

- Has a similar syntax to that of C++
- Object-oriented with automatic garbage collection
- Secure - virus-free, tamper-free
- Robust - not memory overwrite
- Portable - write once run everywhere
- Built-in support classes for window application, GUI objects, networking, multithread and distributed applications, etc

Java Components

- The Java compiler, **javac**, translates Java source code to byte code instructions, "class files", which cannot be directly executed by a processor. Byte codes are a highly portable intermediate instruction (not machine language) for an imaginary Java computer
- The Java Virtual Machine, **java**, is a program that translates/executes Java code instructions. Just-In-Time (JIT) compilation - some JVM can translate byte code into machine language to speed up the program execution.
- The **appletviewer** is a Sun Microsystems tool for viewing HTML documents with applets
- Java Application Programming Interface (API) is a collection of packages of standard Java classes for building Java applications

Java Class Packages (Libraries)

Java classes are prepackaged and available for application programs through the *import* statement. For example, to use mathematical routines, you may import Math class in the java.math package as follows:

import java.math;

However, all classes under java.lang package are automatically available to any Java program so that you may not need to explicitly import this Java.lang package.

Some important Java API packages (libraries) are:

- java.applet (applet classes)
- java.awt (abstract window toolkit package)
- java.io (input/output package)
- java.lang (Java language package)
- java.net (Java networking package)
- java.util (Java utility package)
- java.beans (reusable Java bean packages)

An incomplete list of other packages for applications includes:

- Utility classes (date, time, event handling, complex data structures such as hash tables, stacks, and vectors, etc) - java.util package
- Text manipulation - java.text package
- Mathematic classes and constants - java.math package
- Graphics user interface and graphics - java.awt
- JavaBeans are classes or sets of classes designed as a reusable component. A bean class is packaged in a JAR file.
- Lightweight components - javax.swing
- Java Foundation Classes - Windowing
- I/O Streams - java.io
- TCP/IP Networking (Socket - TCP socket, ServerSocket - server TCP socket, InetAddress - host IP address, DatagramSocket - UDP socket) and

Applications such as FTP (File Transfer Protocol), HTTP (HyperText Transfer Protocol), and SMTP (Simple Mail Transfer Protocol) - java.net

- Applets (java.applet)
- Remote Method Invocation (java.rmi)
- Native Language Interface (for using C/C++ modules within a Java program)
- Encryption and Security (java.security)
- Database Access through JDBC (java.sql)
- Naming and Directory Service
- CORBA (Common Object Request Broker Architecture) support (org.omg.CORBA)
- Internet Server & Plug-Ins
- Telephony
- Java Embedded APIs: APIs for cellular phones, and appliances
- Java Commerce: An Internet-based API for providing secured economic transactions across an insecure network.

Java Beans

- JavaSoft Definition: “A Java Bean is a reusable software component that can be manipulated visually in a builder tool.”
- For building GUI-intensive applications similar to Visual Basic and Delphi environment through control objects
- Needs to implement serializable interface for maintaining persistence

Java Enterprise Computing Packages

It consists of the four separate libraries for accessing an organization resources and applications:

- Java DataBase Connectivity (JDBC)
- Interface Definition Language (IDL)
- Remote Method Invocation (RMI)
- Enterprise Java Beans

- Java's new component model for enterprise applications
- Combines service-side components with distributed object technologies such as CORBA (Common Object Request Broker Architecture) and Java RMI (Remote Method Invocation)
- It considers the requirements of business systems: security, resource pooling, persistence, concurrency, and transactional integrity

1.3 Java Programming Environment and Tools

Java Development Tools and Web Sites

- Borland Jbuilder (http://www.borland.com/jbuilder/)
- Sun Java Forte Integrated Development Tools (http://www.sun.com/forte/ffj/)
- Symentic.com (http://www.visualcafe.com/products/visual_cafe/)
- Borland Jbuilder (http://www.borland.com/jbuilder/foundation/)
- IBM VisualAge (http://www-106.ibm.com/developerworks/java/)
- IBM WebSephere

SUN Java	http://java.sun.com/
Microsoft	http://www.microsoft.com/java
MAC OS	http://applejava.apple.com/
IBM	http://www.ibm.com/Java/tools/jdk.html

Development Tools

Java™ 2 Platform, Standard Edition (J2SE™) and Java™ 2 Platform, Enterprise Edition (J2EE™)

- Available through www.java.sun
- A complete development and deployment platform for the dot-com age
- J2SE software includes a set of tools for creating graphical user interfaces (GUIs), accessing enterprise resources (databases, directories, CORBA-based backend systems), fine- grained security, and many development APIs and functions

- System requirements
    - Microsoft Windows 2000, NT, 98, and 95 operating systems running on the Intel Architecture platform
    - Pentium 166-MHz or faster processor
    - At least 32 Mbytes of physical RAM is required to run GUI applications. 48 Mbytes is recommended for applets running within a browser using the Java Plug-in product.
    - 65 Mbytes of free disk space is required to install Java 2 SDK software. When installing the separate documentation download bundle, an additional 120 Mbytes of free disk space is needed.

IBM® (www.ibm.com)

- VisualAge for Java

- IBM® WebSphere™ preview technologies for Windows® offering is a member of the WebSphere family that consists of a collection of emerging, or new versions of existing, e-business infrastructure technologies

1.4 Java versus C++

Java Programs

- Java stand alone applications (console or window-based)
- Java applets (Web browser)
- Java servlets (server side)
- Java distributed remote objects

Java Native Methods

- A mixed-language solution (not 100% pure Java)
- Support only calling a C/C++ function (native code) from Java
- The needs
    - Reuse C/C++ functions or codes

    - Access system features or devices (such as serial port, digital I/O devices, etc.)
    - Maximize code execution speed
- Disadvantages
    - Losing program portability

Java as a Better C++

- Java was developed at Sun Microsystems as an "Internet appliance" programming language with C++ like syntax
- Java code is collected into packages. A package may span several files.

Java and C++: Similarities

- C++ style statements, declarations, and expressions
- C++ like operators
- Constructors
- Garbage collection
- Streams
- Low level data: byte, char, int, long, float, double
- Interfaces, packages
- Classes and similar access privileges
- support exceptions (Similar to C++, but stronger)
- support persistent data
- support multi-threaded programming
- support distributed applications (TCP/IP-based, Internet distributed)
- have better integration of multi-file projects than C++
- have a data structures library including (classes)
    - BitSet
    - Date
    - Dictionary
    - Hashtable
    - Observable
    - Properties
    - Random
    - Stack
    - StringTokenizer
    - Vector
    - Enumeration is called Iterator in other programming languages.

Java and C++: Differences

- No pre-processors and header files
- No structs -- use classes
- No unions -- use inheritance
- No pointers -- use references to objects
- No goto
- No typedef -- it is automatic, actually unitialized data.
- No global variables
- No implicit type conversions
- No overloaded operators
- No standalone functions, friend functions, virtual functions

Java	**C++**
final	const
import	include
finalize	destructor
single inheritance	multiple inheritance
true OOP	Hybrid
no template	template or parameterized class
no destructor	may have
no dynamic mem. management	may have dynamic memory management
garbage collection	malloc(), dealloc()
no operator overload	operator overload
no call by reference	call by reference

Java vs. C/C++ efficiency differences

- Automatic Garbage collection (slow)
- Java is slower than Interpreted Just In-time (slow)

Java vs. C++, a paper, copyright by Ulrich Stern (http://sprout.stanford.edu/uli/java_cpp.html)

Abstract

This page compares the execution speed of Java versus C++ using two different types of sort algorithms -- a generic bubble sort and the sort algorithms from the Java and C++ libraries. The runtimes of the algorithms are measured in several different execution environments, so they can also be used to compare machines and operating systems.

1.5 Stand-Alone Java Programs

Stand-Alone Programs

- Programs that runs on the computer in the same manner as any other Window-based or text-based applications
- A Java stand-alone program is a class that has a main () method

1.6 Java Programs (Applets and Servlets) for Internet-based Applications

Java Applet

- Applets are objects that run inside a Java-enabled Web browser or an applet viewer and are normally used to perform the following functions: dynamic Web page, computations, animation, and sound
- A Java applet does not have a main() method
- An applet is downloaded along with a HTML page from Internet or Intranet
- Restrictions:
    - Cannot manipulate local computer's file system
    - Can only communicate with the host computer from which it was downloaded
- An applet is an extension of the Java class called **java.applet.Applet**
- Defined in the java.awt package

Java Applets vs. Applications

Java Applications

- Standalone
- Begin with a main() method
- Do not require browsers
- Can be networked
- Used for client/network apps

Java Applets

- Run on the Web
- Begin with an init() method
- Used for Internet or intranet applications
- Applet Classes

  `java.applet.Applet`
  `javax.swing.JApplet`

Java Networking and Distributed Applications

Java networking applications can be created through the inclusion of the following classes:

```
java.net.*
java.net.URL
java.net.URLConnection
java.net.DatagramSocket
```

1-7 Getting Started with Java

- Download the Java 2 Platform Standard Edition (J2EE) or Java 2 Standard Edition (J2SE) through the web site: http://java.sun.com, and install Java SDK. Be sure that path to the javac.exe compiler and java.exe run time machine are set correctly.
- Necessary steps for creating and running a Java program includes
    - Use a text editor such as (notepad) to create a program and name it with.java as the extension, for example: MyProg.java
    - Compile the java source code to a class file or byte code (MyProg.class):
        - c:\ javac MyProg.java
    - Run java virtual machine (Interpreter) to execute the MyProg byte code:
        - c:\java MyProg

Programming Errors

- Syntax error - violation of language syntax rule
- Run-Time Errors - illegal operations, div by zero
- Logic Errors - designing errors or bugs

1.8 Java Programming Examples

Example 1-1: The first Java example for text-based environment.

```
// jaHello.java
// 1. Edit the program using the Notepad
// 2. Set path for Java compiler, JVM, etc:
//    UnderWindows 95/98, insert into AUTOEXEC.BAT a line of the following
//    SET PATH=C:\jdk\bin; %PATH%
//     -- where C is the drive that installed the JDK; it may be D drive
//     -- jdk should be actual directory such as d:\jdk1.3.1\bin
//    Under Windows 2000, set up the PATH (System Environment Variables):
//    Control panel > System > Advanced tab > Environment Variables;
//    Scroll this window to find PATH variable, and add the following line
//    to it; If JDK is installed in the C drive:
//    C:\jdk\bin;
//    Under Linux/Unix systems
//    C shell (Solaris default) - add the line to the end of your
//    ~/.cshrc file:
//          set path=(/usr/local/jdk/bin $path)
//    Bourne Again shell (Linux default) - add the following line to your
//    ~/.bashrc or ~/.bash_profile file:
//          export PATH=usr/local/jdk/bin:$PATH
// 3. For Winows95/98 - Open MS-DOS Window; for Windows 2000 - run cmd.exe
//    use command Window; for Linux/UNIX - run under a shell
// (Using SUN Java SDK)
// 4. Compile javac jaHello.java
//   c:\jdk.1.3.1\bin\javac jaHello.java   .... case sensitive
// 5. Execute java jaHello.class
//   c:\jdk.1.3.1\bin\java jaHello

import java.lang.System;
public class jaHello{
      public static void main(String args[]){
      System.out.println("Hello Triple Crown!");
      }
       }
```

Output:

```
E:\LinJava\LinJavaExs>java jaHello
Hello Triple Crown!
```

Example 1-2: Examples of creating and accessing String objects.

```
/** HelloJava.java
 * @version 1.10 24 April 2001
 * @author Paul I-Hai Lin
 */

public class HelloJava
{  public static void main(String[] args)
   {
      String message[] = new String[5];
      message[0] = "Welcome to Java Programming Language!";
      message[1] = "Deaprtment of Electrical and Computer Engineering
      Technology";
      message[2] = "Purdue University Fort Wayne Campus";
      message[3] = "Instructor: Prof. Paul Lin";
      message[4] = "April 25, 2001";

      int i;
      int stringLen = message.length;
      for (i = 0; i < stringLen; i++)
         System.out.println(message[i]);
   }
```

Output:

```
E:\LinJava\LinJavaExs>java HelloJava
Welcome to Java Programming Language!
Deaprtment of Electrical and Computer Engineering Technology
Purdue University Fort Wayne Campus
Instructor: Prof. Paul Lin
April 25, 2001
```

Example 1-3: A Java program receives input to the program from command line arguments.

```
/**
 * readArg.java
 */
class readArg {
      public static void main(String[] args) {
            for (int i = 0; i < args.length; i++)
                  System.out.print(i == 0 ? args[i] : " " + args[i]);
            System.out.println();
      }
}
```

Output:

```
E:\LinJava\LinJavaExs>java readArg 1st 2nd 3rd 4th
1st 2nd 3rd 4th
```

Example 1-4: Using GUI dialog box to display a message.

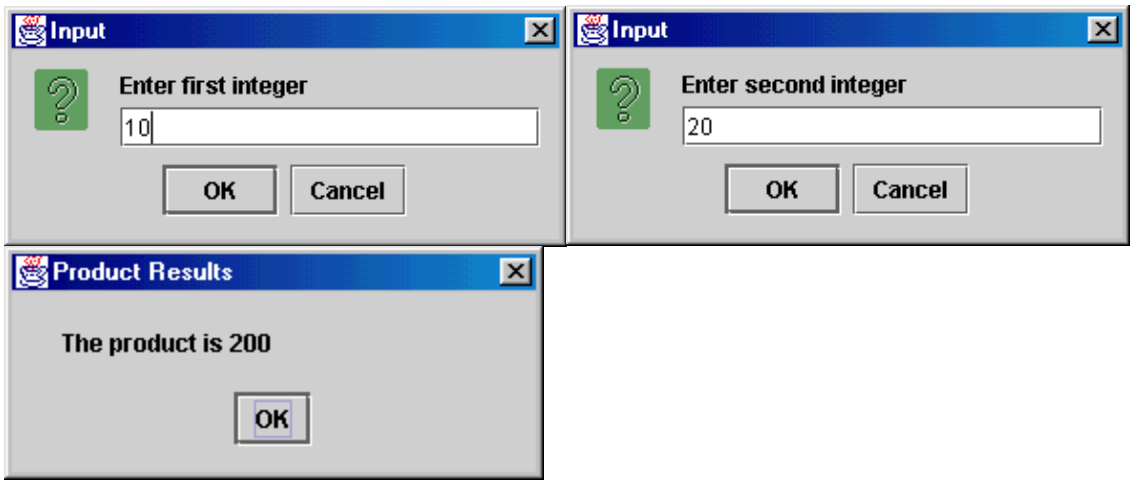
```
// SWDialogB.java
// Printing message in a dialog box
//
// E:\LinJavaExs\1_Basics>javac SWDialogB.java
// E:\LinJavaExs\1_Basics>java SWDialogB
//
// Click OK on the dialog box to exit the program.

import javax.swing.JOptionPane;

public class SWDialogB {
   public static void main( String args[] )
   {
      JOptionPane.showMessageDialog(
         null, "Hello\nTriple Crown!" );

      System.exit( 0 );  // exit the program
   }
}
```

Result:

Example 1-5: Using two GUI components: prompt and dialog boxes.

```
// MathOp.java
//
// This program performs the following tasks:
// 1. Display a prompt dialog box for the user to input first number
// 2. Display a second prompt dialog box for the user to input second
//    number
// 3. Converts two number strings to integer type
// 4. Find the product of the two numbers
// 5. Display the result in a dialog box
//
// You must enter number string in the text field area, then click OK
button, otherwise,
// an exception will be thrown.

import javax.swing.JOptionPane;

public class MathOp {
   public static void main( String args[] )
   {
      String numS1, numS2;
      int num1, num2, result;

      // Input first number string
      numS1 = JOptionPane.showInputDialog( "Enter first integer" );

      // Input second number string

      numS2 = JOptionPane.showInputDialog( "Enter second integer" );

      // convert strings to numbers
      num1 = Integer.parseInt( numS1 );
      num2 = Integer.parseInt( numS2 );

      // multiply the numbers
      result = num1 * num2;

      // Show the product
      JOptionPane.showMessageDialog(
         null, "The product is " + result, "Product Results",
         JOptionPane.PLAIN_MESSAGE );

      System.exit( 0 );   // exit the program
   }
}
```

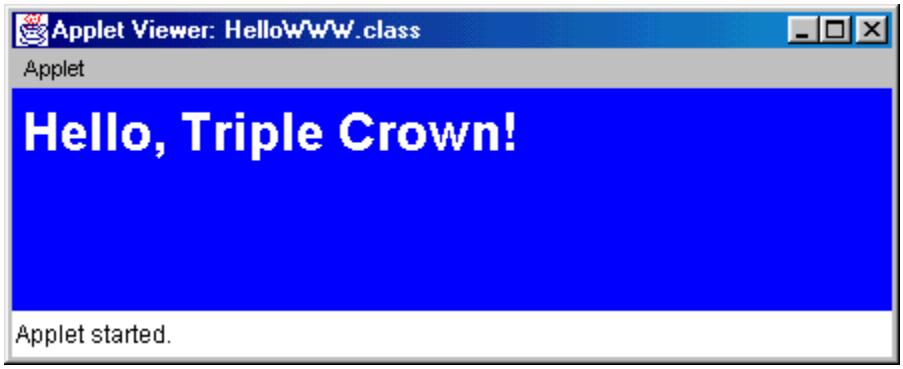
Result:



Example 1-6: A simple Java Applet example.

```
//HelloWWW.java
// 1. Compile this source code to Hellowww.class
// 2. Preapre a HTML file that include the following line:
//    <APPLET CODE="HelloWWW.class" WIDTH=700 HEIGHT=50>
// 3. Use the appletviewer.exe to run the applet
//    path = c:\jdk1_3_0_02\bin\appletviewer
//    E:\LinJavaExs\1_Basics\appletviewer HelloWWW.html
//     or
//    E:\LinJavaExs\1_Basics\appletviewer hellowww.html
//
// 4. Use a Java enabled browser to run the applet: assume that
//    the IE5 is used. Future Internet Explorer will not support
//    Java Applet
//
//
import java.applet.Applet;
import java.awt.*;

public class HelloWWW extends Applet {
  private int fontSize = 26;

  public void init() {
    setBackground(Color.blue);
    setForeground(Color.white);
    setFont(new Font("Helvetica", Font.BOLD, fontSize));
  }
  public void paint(Graphics g) {
    g.drawString("Hello, Triple Crown!",
                 5, fontSize+5);
  }
}
```

```
=====================================================================
<!-- HelloWWW.html -->

<HTML>
<HEAD>
  <TITLE>HelloWWW: A Simple Applet Test.</TITLE>
</HEAD>

<BODY BGCOLOR="white">
<H1>HelloWWW: A Simple Applet Test.</H1>
<P>
<APPLET CODE="HelloWWW.class" WIDTH=700 HEIGHT=50>
  <B>Error! A Java enabled browser must be used.</B>
</APPLET>

</BODY>
</HTML>
=====================================================================
```



Problems

1. Navigate and explain the Java directories of a computer system that installs either the J2SE JDK or J2EE JDK.

2. Compile and execute examples 1-1 through 1-6 on your computer. Are the results the same as shown in this chapter.