J++ Tutorial

- Upgrading Visual J++ to Java 1.1
- Applet and Application
- Building a single J++ file
- Building a Java project (multiple files)
- Creating an Applet with Applet Wizard
- Using the Resource Wizard
- Using the Visual J++ debugger

Upgrading Visual J++ to Java 1.1

- If you are using MicroSoft Visual J++ 1.1, it actually is Java 1.02.
- To check the version of your compiler
  - run jvc.exe,
  - or view the file properties for jvc.exe,(right click on mouse, select properties, select the “Version” tab, and click on “Product Version”.

Upgrading Visual J++ to Java 1.1 (continued)

- Upgrading the compiler to Java 1.1
  - Back up your old compiler (*jvc.exe*). If you accepted the default installation directory J++ suggested, *C:\Program Files\DevStudio*, then rename *jvx.exe* to *jvx.foo* in *C:\Program Files\DevStudio\SharedIDE\bin*
  - Download the new version of the java compiler from John Harding's site or Microsoft.
  - Extract the files from *compiler.zip* into the *SharedIDE\bin* subdirectory
- Installing the new Java Virtual Machine
  - Download the updated Java Virtual Machine (*msjavx86.exe* ~ 4.6MB)
  - Run *msjavx86.exe* to install the Java VM
- Visit http://ei.cs.vt.edu/~ray/cs2704/ to download the required files

Applets and Applications

- There are 2 types of Java programs, applets and applications.
- An applet:
  - runs within a browser window (Internet Explorer)
  - must define a class derived from the Applet class.
  - does not have a *main()* method but is controlled by the browser.
- An application
  - is a standalone program.
  - defines a class with a *main()* method that controls its execution.

Building a Single J++ File

- From the File menu, click New. The New dialog box appears.
- Click the Files tab, select Java source file. Click OK.
- Edit your source file. Then from File menu, click Save, save your file (with extension .java) to your directory.
- From Build menu, click Build. Answer yes when a dialog appears: “Would you like to create a default project workspace?”
- Execute your applet or application (continued)

To Execute an Applet

- In the *Project* menu, click *Settings*.
- The “Project Settings” dialog box appears. Click the *Debug* tab.
- In the drop-down list labeled Category, click *General*.
- In the “Class for debugging/executing” text box, type the Applet-derived class name without a file extension.
- If you are programming an applet, select *Browser* for the “Debug/execute project under” text box.
- Click *OK*. Then in the *Build* menu, click *Execute*.

To Execute an Application

- In the *Project* menu, click *Settings*.
- The “Project Settings” dialog box appears. Click the *Debug* tab.
- In the drop-down list labeled *Category*, click *General*.
- In the “Class for debugging/executing” text box, type the class name (without extension) that contains the *main()* method.
- In the “Debug/execute project under” text box, select *Stand-alone interpreter*.
- Click *OK*. Then in the *Build* menu, click *Execute*.

Creating a Java Project

- From the *File* menu, click *New*. The *New* dialog box appears.
- Click the *Projects* tab.
- Select *Java Project*.
- In the *Project Name* text box, type the name of the project.
- In the *Location* text box, type the directory you installed the files in.
- Click *OK*. Visual J++ creates a new project.

Adding Files to the Project

- From the *Project* menu, click the *Add To Project* submenu.
- From the *Add To Project* submenu, click *Files*. The Insert Files into Project dialog box appears.
- In the *File Name* text box, type *.JAVA (or *.HTML). The * represents the names of source files.
- Select all the files. Click *OK*. Visual J++ adds the files to the project.

Building the Project

- From the Build menu, click *Build*. This command builds the entire project. You can also compile individual source files using the *Compile* command in the *Build* menu.
- Note: With the Java language, there isn't a one-to-one mapping between .*JAVA* files and .*CLASS* files. The Java compiler creates a .*CLASS* file for each class declared in the .*JAVA* file.

Running the Project

- From the *Build* menu, click *Execute*.
- The *information for the Running Class* dialog box appears.
- If you are running an applet, type the Applet-derived class name, and select *Browser*.
- If you are running an application, type the class name that contains the *main()* method, and select the *Stand-alone interpreter* option.

Creating an Applet or Application with the Applet Wizard

- To create new Java applet projects, Visual J++ provides Applet Wizard, a tool that generates a set of starter files.
- You can use the starter files as a template for developing your own Java applet or application**.**
- There is a fundamental difference between applications and applet: an applet automatically has a graphical user interface (a portion of the HTML page that contains it), while an application is by default text.
- Applet Wizard generates code to handle this difference by defining a class derived from the *Frame* class.

Running Applet Wizard

- Select Java Applet Wizard.
- In the Location text box, type the directory you want to save your project files (Generally the default directory \Program Files\DevStudio\MyProjects)
- In the Name text box, type the project name.
- Click OK. (if there is error message “invalid path specified”, this means you have no right to access \Program Files\DevStudio\MyProjects, then switch to your own directory.)
- There should be several Java Applet Wizard dialogs box that appear sequential, select the options you prefer.

Running Applet Wizard (Continued)

- The *New Project Information* dialog box appears, summarizing the settings and features Applet Wizard will generate for you when it creates your project. If you want different options, you can cancel Applet Wizard and run it again.
- Applet Wizard creates all the necessary files and opens the project.

Using the Resource Wizard

- With traditional Windows development tools, user-interface elements like menus and dialogs are specified using resource files separate from the source files.
- In Java, by contrast, menus and dialogs are created in the source code, using Java's Abstract Window Toolkit (AWT).
- A resource Wizard is a tool for converting Windows resource templates into equivalent Java code.
- For the purposes of this scenario, create a new project ; Once you've created this project, you are ready to proceed with using Resource Wizard.

Major Steps in Using the Resource Wizard

- Creating a resource template
- Adding a resource to a resource template (.RCT) file
- Running the Resource Wizard
- Using the Resource Wizard-generated Resource

Creating a Resource Template

- From the *File* menu, click *New*. From the *Files* tab, click *Resource Template*.
- To give your project's resource template a name, fill in the *File Name* text box .
- The *Location* text box should contain the path of the project's directory .
- Click *OK*.

Adding a resource to a resource template (.RCT) file

- Let's add a dialog resource to a resource template (.RCT) file:
  - From the *Insert* menu, click *Resource*. Click *Dialog* and then *OK*.
  - Visual J++ opens a dialog editing window.
  - Press *ALT+ENTER* to display the property page for the dialog. Change the *ID* of the dialog resource to *NewDialog*.
  - Add an *edit* box, a *button*, and a *static text control* to the *dialog* by dragging the control icons from the *Controls Toolbox palette* to the *NewDialog* resource.
  - Close the dialog editing window. Notice that the project's .RCT file folder now has a Dialog folder that contains the *NewDialog* dialog resource you've just created.
- Close the *resource template window* before running the Resource Wizard

Running the Resource Wizard

- To run Resource Wizard
  - From the *Tools* menu, click *Java Resource Wizard*.
  - In the *File name edit* box, specify the .RCT file you created previously, or click *Browse* to find it.
  - Click *Finish*. A dialog box appears displaying the names of the files generated. Click *OK*.
- For each resource in the resource template, Resource Wizard creates a *.JAVA* file containing a class whose name is the *ID* of the resource.
- In addition, when there are dialogs in the resource template, Resource Wizard also creates a separate file for a *DialogLayout* class, which is used by the *dialog* class(es) to manage the layout of *controls*.

Preparing to Debug a Visual J++ Project

- To debug a Java application
  - From the *Project* menu, choose *Settings*.
  - Select the *Debug* tab.
  - In the *Class for debugging/executing* text box, type the name of the class that contains the application's *main()* method.
    - When typing the class name, remember that Java class names are case sensitive.
  - Click *Stand-alone interpreter*.
  - Click *OK*. From the *Build* menu, click *Start Debug*.
  - Click *Go*, *Step Into* or *Run to Cursor*.

Preparing to Debug a Visual J++ Project (Continued)

- To debug a Java applet
  - From the *Project* menu, choose *Settings*.
  - Select the *Debug* tab.
  - In the *Class for debugging/executing* text box, type the name of the class that contains the applet's *init()* method.
    - When typing the class name, remember that Java class names are case sensitive.
  - Click *Browser*.
  - Click *OK*. From the *Build* menu, click *Start Debug*.
  - Click *Go*, *Step Into* or *Run to* Cursor.