Eclipse IDE

Lecture 3

What is an IDE?

- Software development involves many different tasks
  - eg Coding, testing, compiling, running, debugging
- Could use a separate tool for each
  - eg Write code with your favorite text editor
- IDE = “Integrated Development Environment”
  - A single tool that supports all these tasks
  - Can address software-specific issues better than separate general-purpose tools (eg text editors)
- Typical features include
  - Syntax highlighting to emphasize structure
  - Code completion (“intellisense”)
Eclipse Background

- **Origin**
  - late 90’s: Code base developed by IBM Canada
  - 2001: Released as open source “Eclipse Project”
  - 2004: Formed fully independent industrial consortium, the “Eclipse Foundation”
    - Most developers are employed by member companies
- **Many different parts, each its own product line:**
  - Core platform, language tools, plug-ins,
- **Solution:** bundled releases (annual)
  - June 2006: “Callisto” (uses 3.2 platform)
  - June 2007: “Europa” (uses 3.3 platform)
  - June 2008: “Ganymede” (uses 3.4 platform)
  - June 2009: “Galileo” (uses 3.5 platform)
  - June 2010: “Helios” (uses 3.6 platform)
  - June 2011: “Indigo” (uses 3.7 platform)

Eclipse Features

- **Multilanguage**
  - First love will always be Java
  - Close seconds: C/C++, Web (XML/HTML/CSS)
  - Perl, Python, Ruby, Rails, Mathematica...
- **Continuous compilation**
  - No compile/build button
  - Syntax/compile-time errors checked as you type
- **Extensible through plug-ins**
  - About 1000 on www.eclipseplugincentral.com
  - FindBugs: statically identifies possible errors
  - Checkstyle: audits code for “good style” violations
  - djUnit: calculates coverage metrics for test cases
Strengths, weaknesses

- **Positives**
  - Price
  - Extensibility through plugins
  - Configurability through many preference settings
  - Support for team collaboration

- **Negatives**
  - Footprint (memory and on disk)

Options for Eclipse Installations

- **Windows lab machines (Baker, Caldwell)**
  - Eclipse icon on desktop
  - Version 3.5(?); Does not have full JDK!

- **Linux login servers**
  - See class web site (under “Resources”)
  - Log in to stdlinux using X-Win32 or VNC
  - Add /class/cse421/local to your path
  - Run start-eclipse
  - Version 3.7 (and Java JDK 1.6)

- **Install at home (Windows, Linux, Mac)**
  - See class web site (under “Resources”)
  - Version 3.7 (and Java JDK 1.6)
Workspace

- Eclipse always uses a workspace
  - Prompted at startup for directory
- A workspace contains
  - Projects (i.e., source files, packages, and libraries)
  - Personal preferences for IDE (e.g., code formatting)
- Generally need just one workspace
  - Use working sets to reduce clutter
- Multiple workspaces useful for:
  - Multiple Eclipse installations (1 per version)
  - Consultants separating work for multiple clients
- To move preferences between workspaces, use “export”
- For this course, 1 workspace is best choice

Workspace Selection Dialog

[Image of Workspace Selection Dialog]
First Launch of Eclipse

Views and Editors

- Building blocks of user interface
- Editor: Associated with input activity
- View: Shows support information
  - Navigate a hierarchy of information
  - Open an editor
  - Display properties for the active editor
- Examples:
  - Java code editor (for writing code)
  - Problems view (compilation errors/warnings)
  - Console view (terminal IO of running program)
  - Class hierarchy view (relating components)
  - Tasks view (todo items)
  - Navigator (file browser)
Examples of Editors and Views

Perspectives

- A perspective is a particular layout of editors and views
  - Tools are the ones most useful for accomplishing a particular task
- Examples of basic perspectives
  - Java (for writing code)
  - Debug (for debugging a program)
  - Resource (for browsing files)
  - Team Synchronizing (for managing collaborative projects)
Example: Java Perspective

First Program

- Launch Eclipse
- Open Java perspective
- Create a project
  - File > New > Project, select “Java Project”
  - Name the project (eg HelloWorldProject)
- Create a class within the project
  - File > New > Class
  - Name the class (HelloWorld)
  - Select checkbox to create main()
First Program Continued

- Auto-generates boiler plate code
  ```java
  public class HelloWorld {
      /**
       * @param args
       */
      public static void main(String[] args) {
          // TODO Auto-generated method stub
      }
  }
  ```
  - TODO item automatically added to Tasks view
  - Boiler plate comments added too

First Program Continued

- Insert code in main method
  ```java
  System.out.println("Hello World");
  ```
  - Syntax error appears (temporarily)
  - Command completion after
    - System.
    - System.out.

- Run application
  - Run > Run as > Java application
  - Console view appears with output
Run Configurations

- Controls which project is run and how that project is run
  - Green button runs current file (more or less)
  - Equivalent to "$ java classname"

- Run > Run Configurations...
  - Select “Java application”
  - Add command-line arguments (see Args tab)
    - Appended to "$ java classname"

- Advantage: same program can be run in different ways, each stored as its own run configuration

Managing Run Configurations
### Personalizing Eclipse

- **Window > Preferences**
  - General > Appearance > Colors and Fonts
  - General > Editors > Text Editors, Show Line Numbers
  - Java > Code Style / Editor
- **Preferences saved in workspace**
  - For multiple workspaces, export preferences to a file, then import in other workspace
  - File > Export > General > Preferences
- **General advice: avoid tweaking too much**

### Extending Eclipse

- **Easy to install powerful plug-ins**
  - recommended: FindBugs, Checkstyle, ECF
- **Plug-ins can impact performance**
  - Consume memory and can slow start up
  - eg Aptana for Web development
- **Important advice: Do NOT install plug-ins manually into Eclipse install directory**
  - Use Eclipse’s installation manager
  - Help > Install New Software...
  - Select “—All Available Sites—”, or
  - Create a “Add...” and enter the URL for the plugin
Useful Keyboard Shortcuts

- Format: ctrl+shift+F
- Open a class (type): ctrl+shift+T
- Go to current item’s declaration: F3
- Autocomplete suggestions: ctrl+space
- Find references to this item: ctrl+shift+G
- Move between methods: ctrl+shift+up/down
- Go to next error: ctrl+.  
- Fix error suggestions: ctrl+1

Supplemental Reading

- Eclipse menu: Help > Welcome
  - Gives original start-up screen with tool overview, tutorials, and code samples
- Eclipse menu: Help > Tips & Tricks...
  - Gives long list of short-cuts and hints
- IBM developerWorks
  - “Getting Started with the Eclipse Platform”
  - “Introduction to Eclipse for Visual Studio Users”
Summary

- An IDE supports all code development tasks
- Eclipse basics
  - Installation
  - Workspaces and projects
  - Editors, views and perspectives
- Hello World tutorial with Eclipse
  - Run configurations
- Customization
- Tips & Tricks