Modern IDE’s
Compare/Contrast

Java Development
Eclipse
Vs
NetBeans

By
Carlos Tafoya
Introduction

- Integrated Development Environments (IDEs) are applications used to aid in software development
  - These programs typically provide many features for authoring, modifying, compiling, deploying and debugging software.
  - The aim is to abstract the configuration necessary to piece together command line utilities in a cohesive unit, which theoretically reduces the time to learn a language, and increases developer productivity.
Brief History Of IDE’s

■ Originally IDE’s weren’t possible
  ■ Early systems could not support one. Programs were prepared using flowcharts, entering programs with punch cards (or paper tape, etc) before submitting them to a compiler.
Brief History Of IDE’s

- IDE’s became possible when developing via a console or terminal.

- MAESTRO-I
  - First IDE
  - MAESTRO I was First Presented in 1975
  - MAESTRO I was created by Softlab Munich
Brief History Of IDE’s

- MAESTRO I
  - The Objective of MAESTRO I was a hardware and software programming tool that could be rented.
    - Cost about as much as a house rental.
  - MAESTRO I was an essential factor in the development of:
    - Software Engineering
    - Origination of Development Environments
    - Man-computer interaction
Brief History Of IDE’s

- MAESTRO I
  - Typical Configuration
    - 96-192 KB RAM memory
    - 6-24 terminals
    - 10-80 MB disc
    - Magnetic Tape Drive
    - Data communication connection
Brief History Of IDE’s

- MAESTRO I
  - The Hardware

CPU

Drives, Printers, etc

Keyboard
Brief History Of IDE’s

- Another historical IDE of note is Emacs
  - Emacs is still used today
  - Some Programmers swear by Emacs
    - Wont use anything else
  - Had a professor that did all his programming in Emacs
    - He even built slides for presentations in Emacs
Brief History of IDE’s

- Emacs Screenshot
Modern IDE’s

- Modern IDE’s
  - Graphical
    - Utilizing a windowing system
    - Menu driven
    - Heavy use of GUI widgets
      - Tab panes
      - Buttons
      - Etc…
  - Customizable
    - Colors and fonts
    - Workspace layout
    - Etc..
Eclipse and NetBeans

- For the remainder of this presentation we will consider Eclipse and NetBeans
- We will concern ourselves with only the Java Development aspect of these two IDE’s
- It should be noted that both IDE’s offer a lot more than just Java development.
  - Support Many Languages
  - Rich Client Programming
  - Etc…
Brief History of Eclipse

- Eclipse began as an IBM Canada project.
- It was developed by Object Technology International (OTI) as a Java-based replacement for the Visual Age family of IDE products.
- In November 2001, a consortium was formed to further the development of Eclipse as open source.
- In January 2004, the Eclipse Foundation was created.
Brief History of Eclipse

- Eclipse cont...
  - The Eclipse Foundation provides four services to the Eclipse community
    - IT Infrastructure
    - IP Management
    - Development Processes
    - Ecosystem Development
Brief History of NetBeans

- NetBeans began as a student project
  - NetBeans was created by Roman Stanek in 1996.
    - Created under the guidance of the Faculty of Mathematics and Physics at Charles University in Prague.
  - In 1997 Roman Stanek formed a company around the project and produced commercial versions of the NetBeans IDE
Brief History of NetBeans

- NetBeans cont...
  - It was bought by Sun Microsystems in 1999.
  - Sun open-sourced the NetBeans IDE in June of the following year. The NetBeans community has since continued to grow, thanks to individuals and companies using and contributing to the project.
Common Features

- Java Project
  - A Java project contains source code, packages, and related files for building a Java program.
    - Java projects can reference other Java Projects
  - It has an associated Java builder that can incrementally compile Java source files as they are changed.
    - NetBeans wasn’t always so good at this. However, new versions do it well.
Common Features

- Java Project
  - A Java project also maintains a model of its contents.
  - This model includes information about the type hierarchy, references and declarations of Java elements.
  - This information is constantly updated as the user changes the Java source code.
Common Features

- **Package Explorer**
  - The Package Explorer view shows the Java element hierarchy of the Java projects in your workbench.
  - It provides you with a Java-specific view of the resources shown in the Navigator.
  - For each project, its source folders and referenced libraries are shown in the tree.
  - You can open and browse the contents of both internal and external JAR files.
Package Explorer

- Spaceinvaders
  - src
    - spaceinvaders
      - AlienEntity.java
      - Entity.java
      - Game.java
      - ShipEntity.java
      - ShotEntity.java
      - Sprite.java
      - SpriteStore.java
    - sprites
      - alien.gif
      - ship.png
      - ship1.gif
      - shot.gif
  - JRE System Library [jre6]
    - resources.jar - C:\Program Files\Java\jre6\lib
    - rtf.jar - C:\Program Files\Java\jre6\lib
      - com.sun.accessibility.internal.resources
      - com.sun.activation.registries
      - com.sun.awt
      - com.sun.beans
      - com.sun.corba.se.impl.activation
      - com.sun.corba.se.implementation
Common Features

- **Java Editor**
  - **Syntax Highlighting**
    - Different kinds of elements in the Java source are rendered in unique colors.
  - **Code Folding**
    - Blocks of code can be collapsed to hide them from the programmer and re-expanded if a programmer needs to view or edit these blocks.
package differentialCryptanalysis;

import java.math.BigInteger;

public class DiffCrypt
{

    private static byte rowMask1 = 0x20;
    private static byte rowMask2 = 0x01;
    private static byte colMask = 0x1e;
    private static long bitMask6 = 0x000f;
    private static int bitMask4 = 0x0f;

    public static byte[] buildOutputXorData(int sBoxIndex, byte[][] deltaSet,)
    
    /**
     * @param inputXor
     * @return
     * @throws Exception
     */
    public static byte[][] getDeltaSet(byte inputXor) throws Exception[]

    public static long expansionFunc(int val[]

    public static int permutation(int val[]

    public static int permutationInverse(int val)
Common Features

- **Java Editor**
  - **Content/Code Assist**
    - In my opinion the most helpful part of the IDE’s
    - View Javadoc information for selected source element
      - Class
      - Method
      - Variable
      - Etc…
  - View and select available members of a Class or a class instance.
    - Selecting a method will auto generate method parameters (makes a best guess at desired parameters)
private boolean firePressed = false;
/** True if game logic needs to be applied this loop, normally as a result of a game event */
private boolean logicRequiredThisLoop = false;
/**
 * Construct our game and set it running.
 */
public Game() {
    // create a frame to contain our game
    JFrame container = new JFrame("Space Invaders 101");
    // get hold the content of the frame and set up the resolution of the game
    JPanel panel = (JPanel) container.getContentPane();

    Returns an array of all the container listeners registered on this container.
    See Also:
    addContainerListener
    removeContainerListener
    Returns:
    all of this container’s ContainerListener or an empty array if no container listeners are currently registered
    Since:
    1.4
Common Features

■ Java Editor

■ Error/Warning marking
  ■ Both errors and warnings are annotated in the vertical ruler of the source code editor.
  ■ Both errors and warnings are marked within the source code in the editor.

■ Code Navigation

■ Treats code like hyperlinks
  ■ References to classes, class instances, and class members can be used to navigate to the source code for the respective elements.
public void move(long delta) {
    // if we have reached the left hand side of the screen and
    // are moving left then request a logic update
    if ((dx < 0) && (x < 10)) {
        game.updateLogic();
    }
    // and vice versa, if we have reached the right hand side of
    // the screen and are moving right, request a logic update
    if ((dx > 0) && (x > 750)) {
        game.updateLogic();
    }
    // proceed with normal move
    super.move(delta);
}
/**
 * Update the game logic related to aliens
 */
public void doLogic() {
    // swap over horizontal movement and move down the
    // screen a bit
    dx = -dx;
    y += 10;
    // if we've reached the bottom of the screen then the player
    // dies
    if (y > 570) {
        game.notifyDeath();
    }
Common Features

- **Java Editor**
  - **Code Templates**
    - Templates are a structured description of coding patterns that reoccur in source code.
    - For example, a common coding pattern is to iterate over the elements of an array using a for loop that indexes into the array.
    - By using a template for this pattern, you can avoid typing in the complete code for the loop.
    - Templates will insert the code into the editor and position your cursor so that you can edit the details.
Common Features

- **Java Editor**
  - **Import Organization**
    - Automatically organizes (add/removes) import statements need by source code.
  - **Refactoring**
    - Java program refactoring is used to make system-wide code changes without affecting the behavior of the program.
    - Users can specify a change in a single place and have that change propagate throughout the code base.
Common Features

- **Java Editor**
  - **Local History**
    - Files and Java elements such as types and their members change in time. A 'snapshot' of what they look like at a point in time (as saved in the local history) is called an edition.
    - A file can be replaced with an edition from the local history
    - User can compare and/or replace individual Java elements (types and their members) with editions from the local history.
    - Users can restore Java elements (and files) deleted from the workbench that have been kept in the local history.
Local History

```java
public void move(long delta) {
    // proceed with normal move
    super.move(delta);
    // if we have reached the left hand side of the screen at
    // are moving left then request a logic update
    if ((dx < 0) && (x < 10)) {
        game.updateLogic();
    }
    // and vice versa, if we have reached the right hand side
    // the screen and are moving right, request a logic upd
    if ((dx > 0) && (x > 750)) {
        game.updateLogic();
    }
    // proceed with normal move
    super.move(delta);
}
```
Common Features

- **Java Editor**
  - **Quick Fix/Quick Assist**
    - For most problems marked with an error/warning, the Java editor can offer corrections.
  - **Javadoc**
    - **Built in support for Javadoc**
      - Allows for content assist when generating Javadoc
      - Can auto create portions of Javadoc
Common Features

- **Java Editor**
  - **Code Formatter**
    - Will auto format code to a users specification
      - Indentation
      - Braces
      - White Space
      - Blank Lines
      - New Lines
      - Control Statements
      - Line Wrapping
      - Comments
Common Features

- Wizards
  - The IDE’s come with a plethora of wizards to aid in the creation of Java Source code.
  - Wizards include:
    - Java Project
    - Classes
    - Interfaces
    - Enumerations
    - Etc…
Common Features

- **Source Code Generation**
  - The IDE can generate pieces of source code based on properties of a Class.
    - Create Stubs to Override/Implement methods from the classes hierarchy
    - Generate Getters and Setters for local variables
    - Generate Delegate methods for wrapped objects
    - Generate `hashCode()` and `equals()` methods
    - Generate Constructors given specified fields
    - Generate Constructors from Super class
Common Features

- Workbench Searching
  - You can perform file, text or Java searches.
  - Java searches operate on the structure of the code.
    - You can search for uses of a specific Class, class instance, or class member
  - File searches operate on the files by name and/or text content.
  - Text searches allow you to find matches inside comments and strings
Common Features

- Java Project Execution
  - Projects can be executed from within the IDE
    - Provides a console for input and output
    - Console also provided error stream for stack traces and other errors.
  - Can define arguments to pass to program
  - Can define arguments to pass to the JVM
Common Features

- Debugging
  - Allows users to step through the execution of a program
  - Define break points
  - Monitor state of variables
  - Perform expression evaluation
  - Suspend Threads
  - Change values of variables during execution
Debugger
Common Features

- **JUnit Integration**
  - JUnit is a simple test framework used to write repeatable tests.
  - Users can run a suite of tests from within the IDE.
  - IDE allows users to see results of running the unit tests.
    - Show all test that pass along with runtime.
    - Shows all tests that fail along with any associated error message.
Common Features

- Javadoc Exportation
  - Users can export Javadoc
  - Users can specify visibility of exported documentation
    - Public
    - Private
    - Package
    - Protected
Common Features

- **Class Outline/Navigator view**
  - Displays an outline of the Java file
    - package declaration, import declarations, fields, types and methods.
  - Allows users to quickly navigate to various portions of a Java source file.
  - Differentiates class members based on accessibility.
    - Public, Private, Protected, and Package
Differences

- Drag and Drop
  - Both IDE’s allow dragging and dropping files (internal to the IDE or external) into the IDE
    - Earlier versions of NetBeans had Pathetic support for drag and drop
  - Only eclipse allows dragging and dropping files from within the IDE to applications outside the IDE
Differences cont

- **GUI Builder**
  - **NetBeans offers a GUI Builder tool**
    - Tool is good for rapid prototyping of a GUI
    - Not so good for building complex GUI’s
    - Code generated by GUI builder is awful
  - **Eclipse doesn’t offer a GUI builder standard**
    - There are plug-ins that give Eclipse this capability.
      - Jigloo for example
      - Similar interface to NetBeans GUI builder.
GUI Builder
Differences cont

- Configuration Management Support
  - Both Eclipse and NetBeans have support for CVS
  - NetBeans comes with support for Subversion and Mecurial
  - Eclipse doesn’t come with support Subversion or Mecurial
    - you can get plug-ins to correct this.
Differences

- **Coder Profiler**
  - The latest version of NetBeans includes an easy to use code profiler
    - Measures Execution time
    - Counts Method invocation
    - Find hot spots (Cool!!!!)

- **Eclipse lacks a code profiler**
  - Surprise, surprise, there are plug-ins to give eclipse a code profiler.
Code Profiler
Conclusion

- Comparable IDE’s
  - NetBeans offers more out of the box functionality than Eclipse
  - Plug-ins exist to add missing functionality to Eclipse
  - Many other plug-ins exist for both Eclipse and NetBeans
  - In the end both IDE’s offer generally the same capabilities
  - Choosing the right IDE for you is a matter of preference.
    - Aesthetics
    - User interface
    - Familiarity
    - Etc…
Future IDE’s

- Visual IDEs
  - There is growing interest in visual programming
  - Visual programming allows users to create new applications by moving programming building blocks or code nodes to create flowcharts or structure diagrams which are then compiled or interpreted.
  - These flowcharts often are based on the Unified Modeling Language.