

Modern IDE's

Compare/Contrast

Java Development

Eclipse

Vs

NetBeans

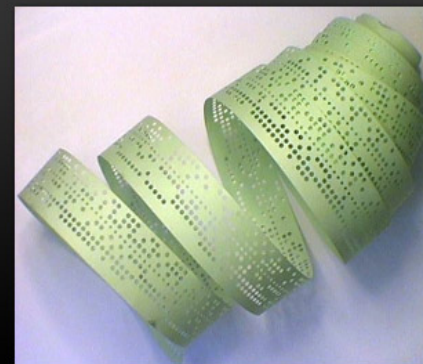
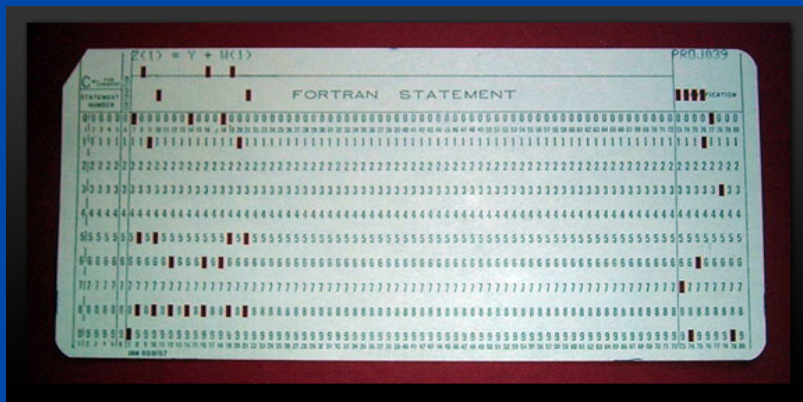
By
Carlos Tafoya

Introduction

- Integrated Development Environments (IDEs) are applications used to aid in software development
 - These programs typically provides 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

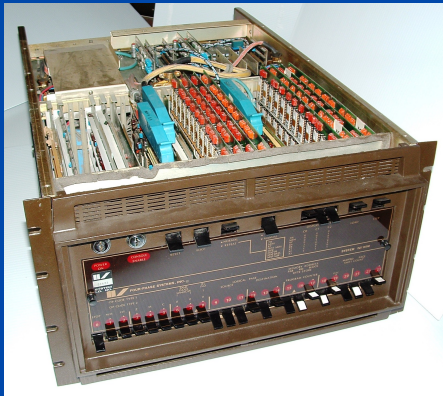
■ MAESTRO I

■ 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

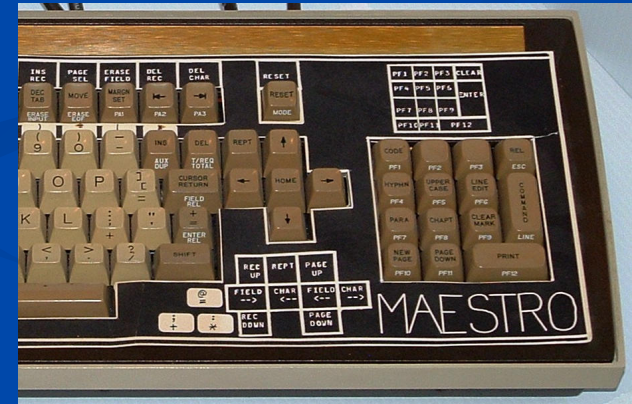
- MAESTRO I
 - The Hardware



CPU



Drives, Printers, etc



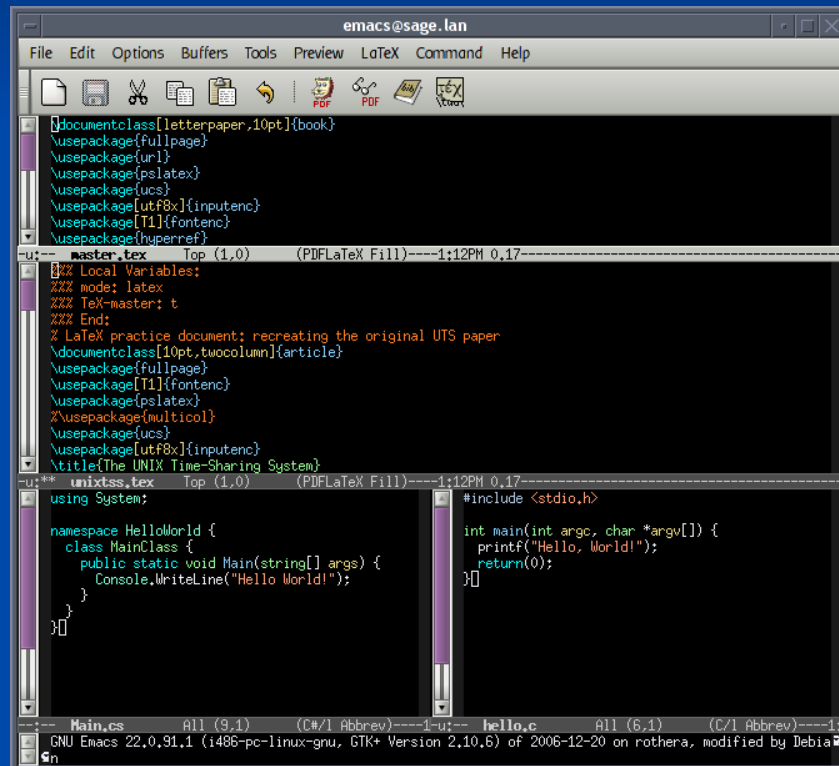
Keyboard

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



The screenshot shows the Emacs text editor window titled "emacs@sage.lan". The interface includes a menu bar (File, Edit, Options, Buffers, Tools, Preview, LaTeX, Command, Help) and a toolbar with icons for file operations and LaTeX. The main workspace is divided into three buffers:

- master.tex** (Top (1,0) (PDFLaTeX Fill)---1:12PM 0.17---):

```
{}documentclass[letterpaper,10pt]{book}
\usepackage{fullpage}
\usepackage{url}
\usepackage{pslatex}
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{hyperref}
```
- unixss.tex** (Top (1,0) (PDFLaTeX Fill)---1:12PM 0.17---):

```
{} Local Variables:
{} mode: latex
{} TeX-master: t
{} End:
{} LaTeX practice document: recreating the original UTS paper
\documentclass[10pt,twocolumn]{article}
\usepackage{fullpage}
\usepackage[T1]{fontenc}
\usepackage{pslatex}
\usepackage{multicol}
\usepackage{ucs}
\usepackage[utf8x]{inputenc}
\title{The UNIX Time-Sharing System}
```
- main.cs** (All (9,1) (C#/1 Abbrev)---1:u:-- hello.c All (6,1) (C/1 Abbrev)---1:):

```
using System;
namespace HelloWorld {
    class MainClass {
        public static void Main(string[] args) {
            Console.WriteLine("Hello World!");
        }
    }
}
```

The bottom buffer shows the C code for **hello.c** (All (6,1) (C/1 Abbrev)---1:):

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    printf("Hello, World!");
    return(0);
}
```

The status bar at the bottom indicates: GNU Emacs 22.0.91.1 (1486-pc-linux-gnu, GTK+ Version 2.10.6) of 2006-12-20 on rothera, modified by Debian.

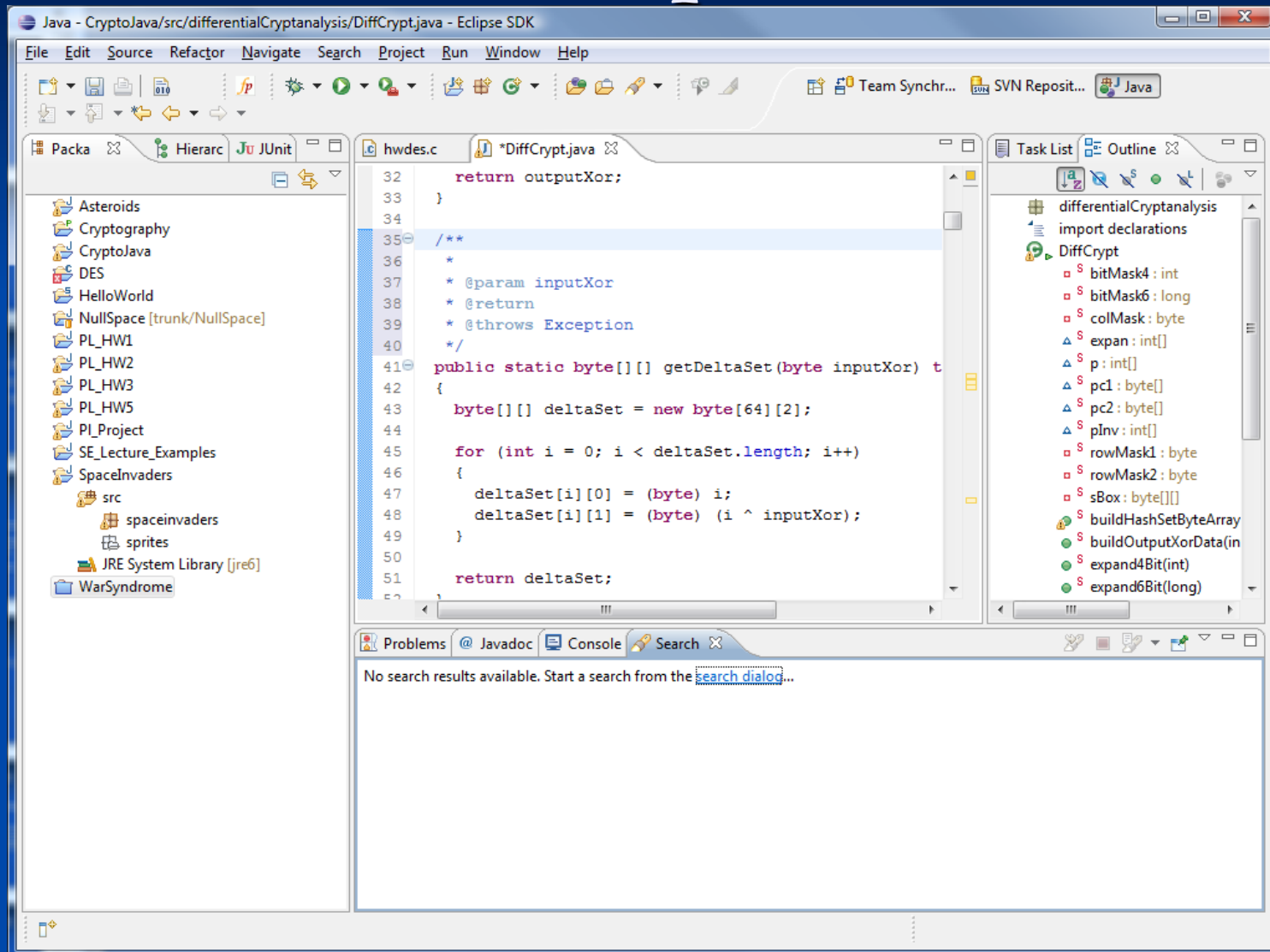
Modern IDE's

- 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..

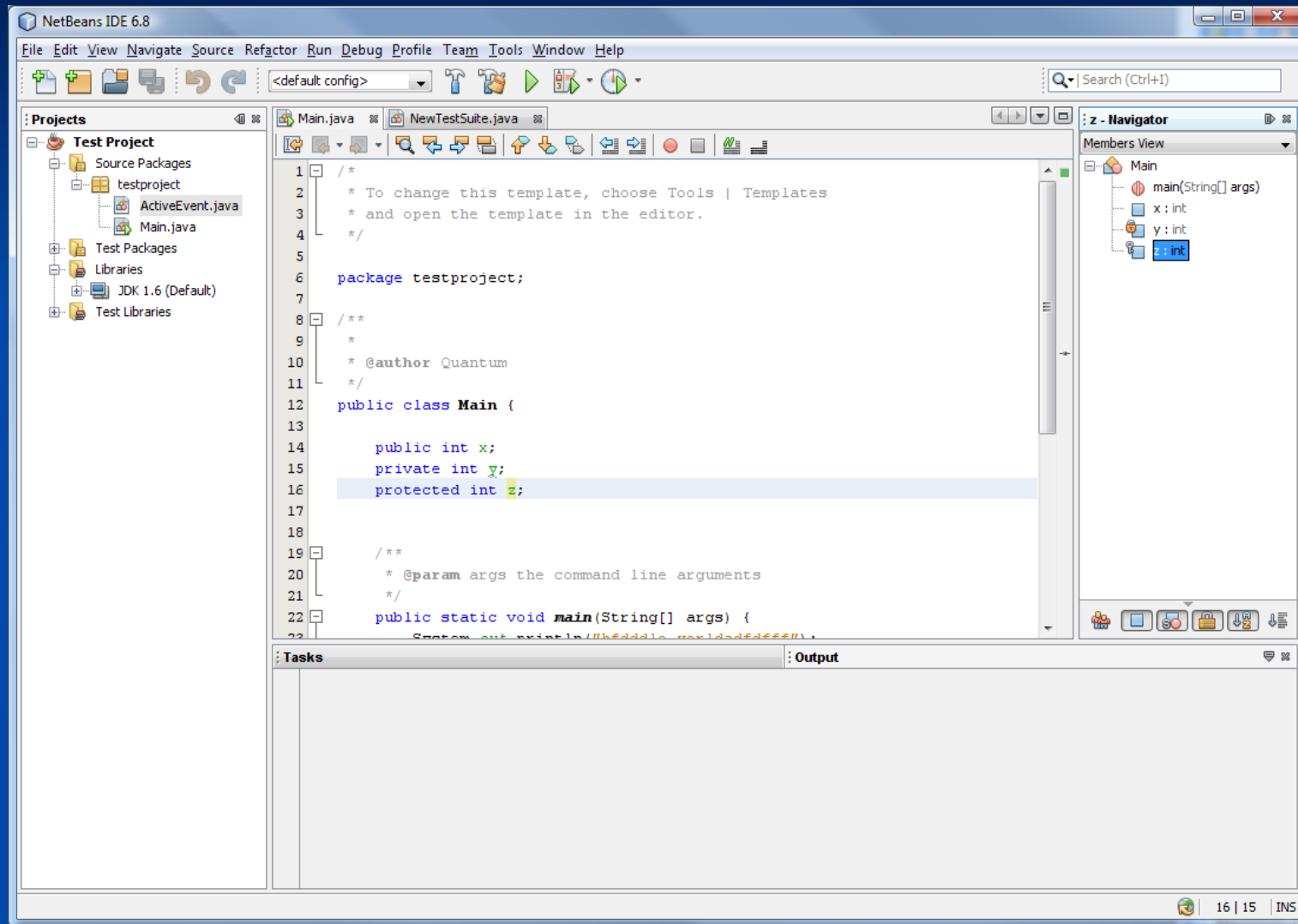
Eclipse and NetBeans

- 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...

Eclipse



NetBeans



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

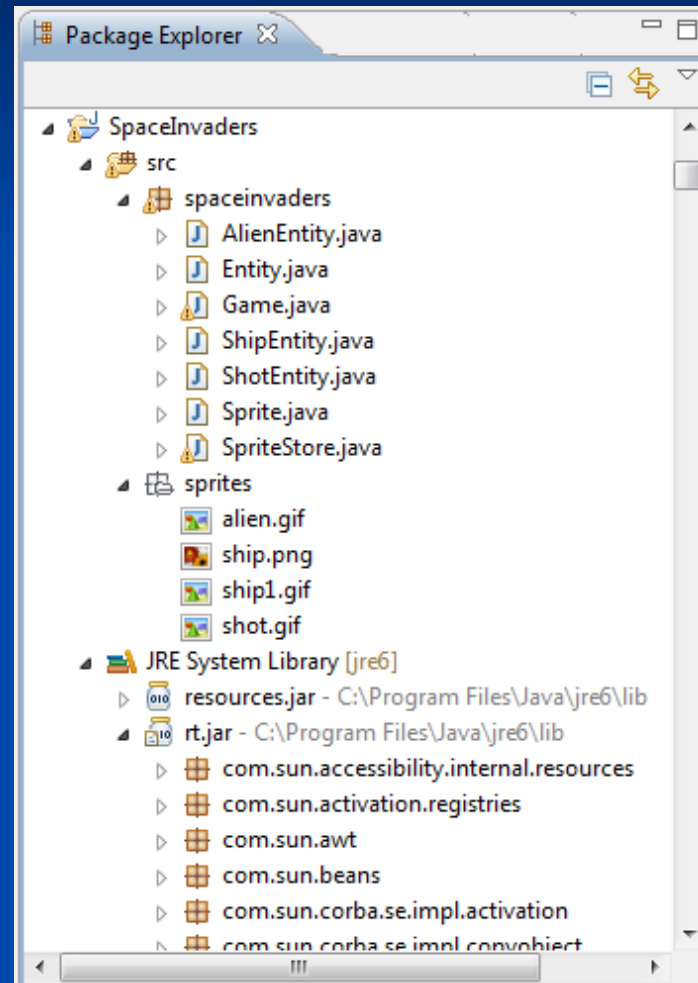
- Java Project
 - A Java project also maintains a model of its contents.
 - This model includes information about the type hierarchy, references and declarations of Java elements.
 - This information is constantly updated as the user changes the Java source code.

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

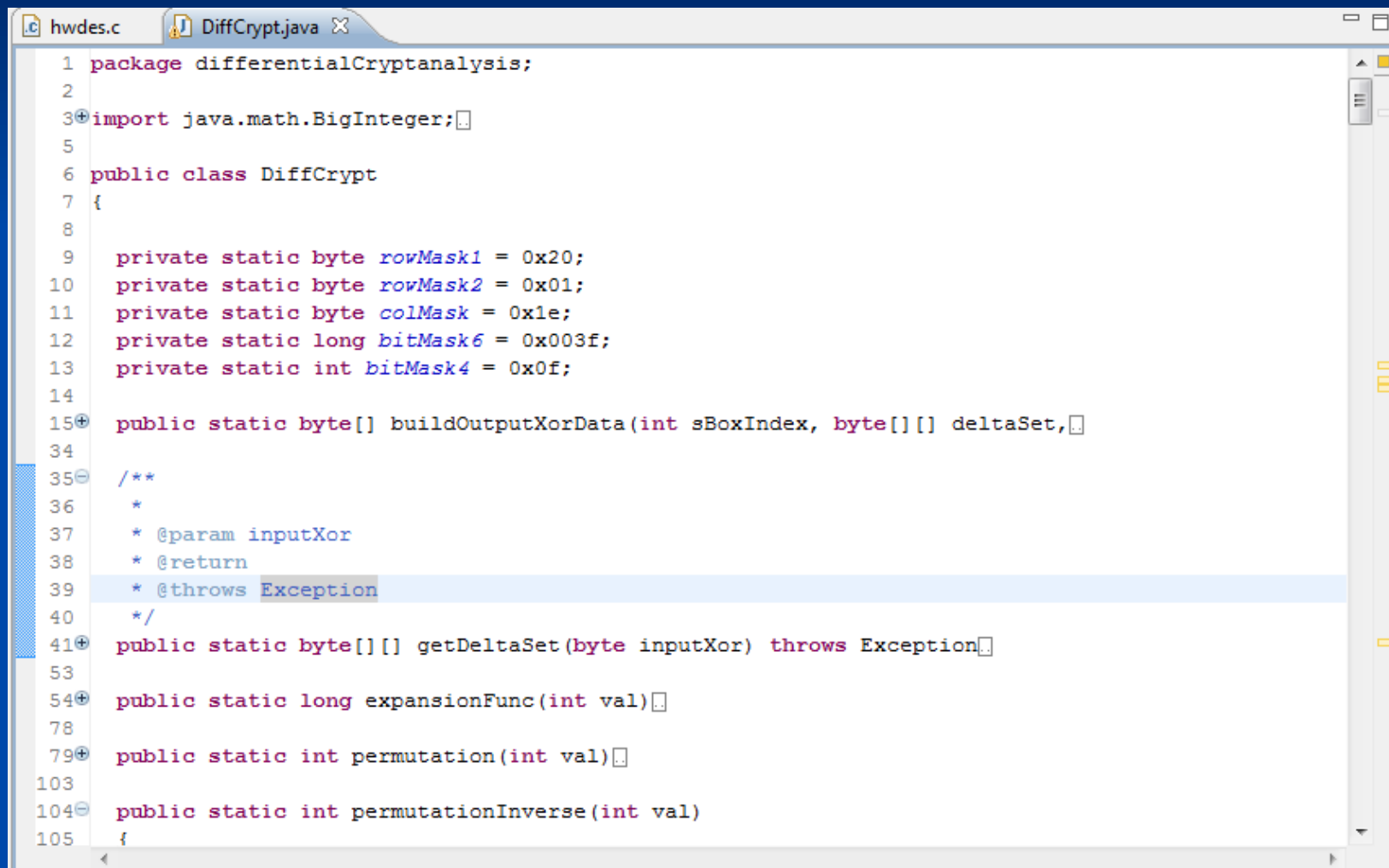
Package Explorer



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

Syntax Highlighting and Code Folding



The image shows a screenshot of an IDE window with two tabs: 'hwdes.c' and 'DiffCrypt.java'. The 'DiffCrypt.java' tab is active, displaying Java code with syntax highlighting. The code is as follows:

```
1 package differentialCryptanalysis;
2
3 import java.math.BigInteger;
4
5
6 public class DiffCrypt
7 {
8
9     private static byte rowMask1 = 0x20;
10    private static byte rowMask2 = 0x01;
11    private static byte colMask = 0x1e;
12    private static long bitMask6 = 0x003f;
13    private static int bitMask4 = 0x0f;
14
15    public static byte[] buildOutputXorData(int sBoxIndex, byte[][] deltaSet,
34
35    /**
36     *
37     * @param inputXor
38     * @return
39     * @throws Exception
40     */
41    public static byte[][] getDeltaSet(byte inputXor) throws Exception
53
54    public static long expansionFunc(int val)
78
79    public static int permutation(int val)
103
104    public static int permutationInverse(int val)
105    {
```

The code is syntax-highlighted: keywords like 'package', 'import', 'public', 'private', 'static', 'byte', 'long', 'int', and 'throws' are in purple; class and method names are in black; literals are in blue; and comments are in grey. The code is also folded: lines 35-40 are collapsed, and line 104 is expanded. The IDE interface includes a scrollbar on the right and a line number gutter on the left.

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)

Content Assist

The screenshot shows an IDE window with several tabs: 'hwdes.c', 'DiffCrypt.java', 'Entity.java', and 'Game.java'. The 'Game.java' tab is active, displaying the following code:

```
61 private boolean firePressed = false;
62 /** True if game logic needs to be applied this loop, normally as a result of a game event */
63 private boolean logicRequiredThisLoop = false;
64
65 /**
66  * Construct our game and set it running.
67  */
68 public Game () {
69     // create a frame to contain our game
70     JFrame container = new JFrame("Space Invaders 101");
71
72     // get hold the content of the frame and set up the resolution of the game
73     JPanel panel = (JPanel) container.getConteContentPane ();
74
75     Returns an array of all the container listeners registered on this
76     container.
77     See Also:
78     addContainerListener
79     removeContainerListener
80     Returns:
81     all of this container's ContainerListeners or an empty
82     array if no container listeners are currently registered
83     Since:
84     1.4
85
86
87
88
89
```

At line 73, the text 'getConteContentPane()' is highlighted, and a content assist popup is visible. The popup contains the following information:

- getContainerListeners() : ContainerListener[] - Container
- getContentPane() : Container - JFrame

At the bottom of the popup, it says: 'Press 'Ctrl+Space' to show Template Proposals'. At the bottom of the IDE window, it says: 'Press 'Tab' from proposal table or click for focus'.

Common Features

■ Java Editor

■ Error/Warning marking

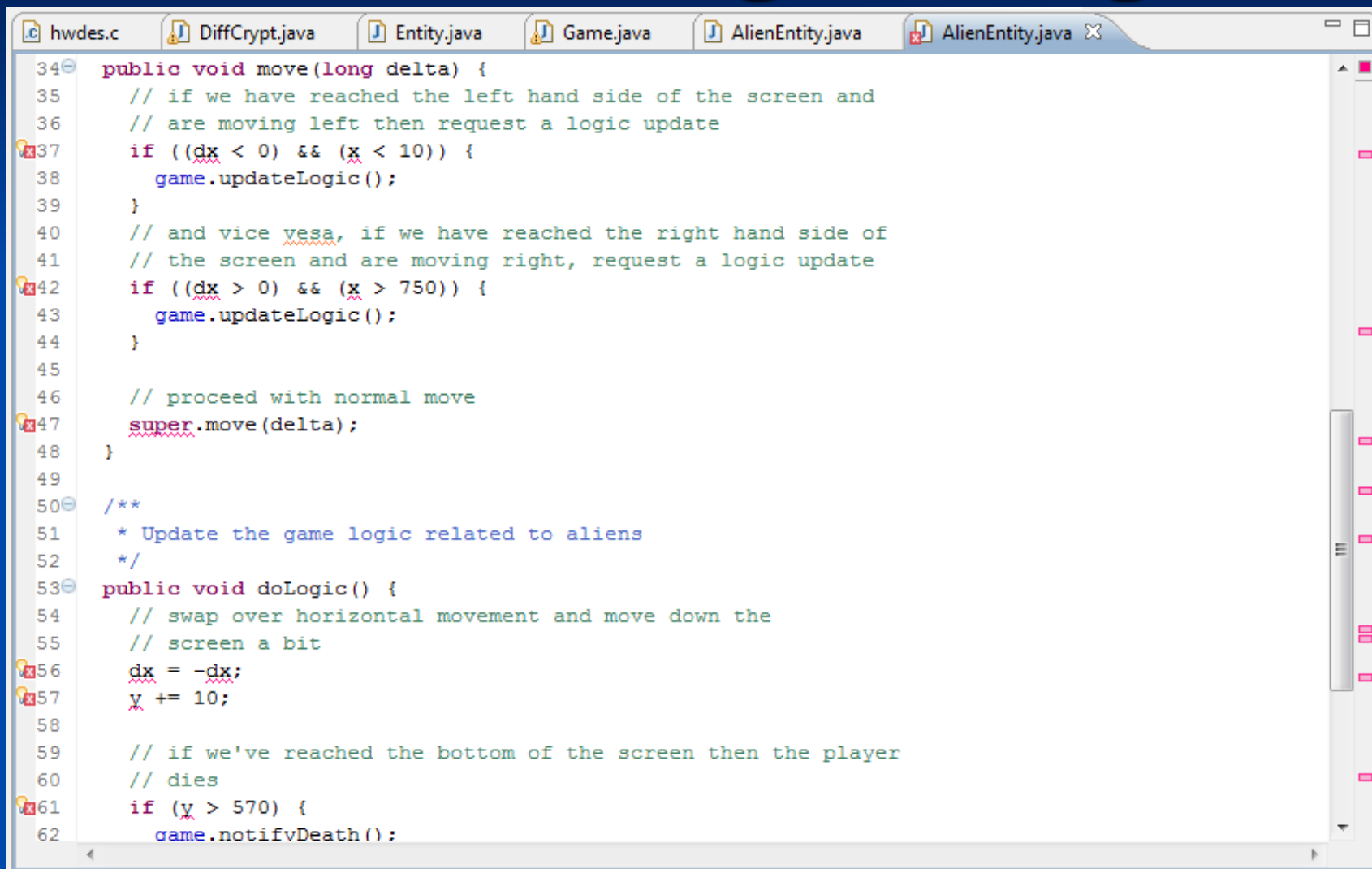
- Both errors and warnings are annotated in the vertical ruler of the source code editor.
- Both errors and warnings are marked within the source code in the editor.

■ 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.

Error/Warning Marking



The screenshot shows an IDE window with several tabs: hwdes.c, DiffCrypt.java, Entity.java, Game.java, AlienEntity.java, and AlienEntity.java. The active tab is AlienEntity.java, which contains the following code:

```
34 public void move(long delta) {
35     // if we have reached the left hand side of the screen and
36     // are moving left then request a logic update
37     if ((dx < 0) && (x < 10)) {
38         game.updateLogic();
39     }
40     // and vice versa, if we have reached the right hand side of
41     // the screen and are moving right, request a logic update
42     if ((dx > 0) && (x > 750)) {
43         game.updateLogic();
44     }
45
46     // proceed with normal move
47     super.move(delta);
48 }
49
50 /**
51  * Update the game logic related to aliens
52  */
53 public void doLogic() {
54     // swap over horizontal movement and move down the
55     // screen a bit
56     dx = -dx;
57     y += 10;
58
59     // if we've reached the bottom of the screen then the player
60     // dies
61     if (y > 570) {
62         game.notifyDeath();
```

The code is annotated with several markers: red 'x' icons on lines 37, 42, 47, and 61; yellow warning icons on lines 37, 42, 47, and 61; and blue squiggly lines under the variable 'x' on lines 37, 42, and 61. The IDE interface includes a scrollbar on the right and a line number column on the left.

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

The screenshot shows an IDE window titled "Java Source Compare" with two panes. The left pane shows the "Local: move(long)" version, and the right pane shows the "Local history: AlienEntity.java Mar 19, 2010 8:46:18 PM" version. A callout box highlights the differences between the two versions.

```
Local: move(long)
29 /**
30  * Request that this alien moved based on time elapsed
31  *
32  * @param delta The time that has elapsed since last move
33  */
34 public void move(long delta) {
35     // proceed with normal move
36     super.move(delta);
37 }

Local history: AlienEntity.java Mar 19, 2010 8:46:18 PM
29 /**
30  * Request that this alien moved based on time elapsed
31  *
32  * @param delta The time that has elapsed since last move
33  */
34 public void move(long delta) {
35     // if we have reached the left hand side of the screen and
36     // are moving left then request a logic update
37     if ((dx < 0) && (x < 10)) {
38         game.updateLogic();
39     }
40     // and vice vesa, if we have reached the right hand side
41     // the screen and are moving right, request a logic update
42     if ((dx > 0) && (x > 750)) {
43         game.updateLogic();
44     }
45
46     // proceed with normal move
47     super.move(delta);
48 }
```

Common Features

- Java Editor
 - Quick Fix/Quick Assist
 - For most problems marked with a 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

Debug - SpaceInvaders/src/spaceinvaders/Game.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Team Synchron... SVN Reposit... Java

Game [Java Application]

- spaceinvaders.Game at localhost:51180
 - Thread [main] (Suspended (breakpoint at line 246 in Game))
 - Game.gameLoop() line: 246
 - Game.main(String[]) line: 447
 - Thread [AWT-Shutdown] (Running)
 - Daemon Thread [AWT-Windows] (Running)
 - Thread [AWT-EventQueue-0] (Running)

C:\Program Files\Java\jre6\bin\javaw.exe (Mar 19, 2010 9:07:14 PM)

Game.java

```
242 // work out how long its been since the last update
243 // will be used to calculate how far the entities
244 // move this loop
245 long delta = System.currentTimeMillis() - lastLoopTime;
246 lastLoopTime = System.currentTimeMillis();
247
248 // Get hold of a graphics context for the acceleration
249 // surface and blank it out
250 Graphics2D g = (Graphics2D) strategy.getDrawGraphics();
251 g.setColor(Color.black);
252 g.fillRect(0,0,800,600);
253
```

Outline

- firingInterval: long
- gameRunning: boolean
- lastFire: long
- leftPressed: boolean
- logicRequiredThisLoop: boolean
- message: String
- moveSpeed: double
- removeList: ArrayList
- rightPressed: boolean
- ship: Entity
- strategy: BufferStrategy
- waitingForKeyPress: boolean

Game0

Console Tasks Problems Executables

Game [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Mar 19, 2010 9:07:14 PM)

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.

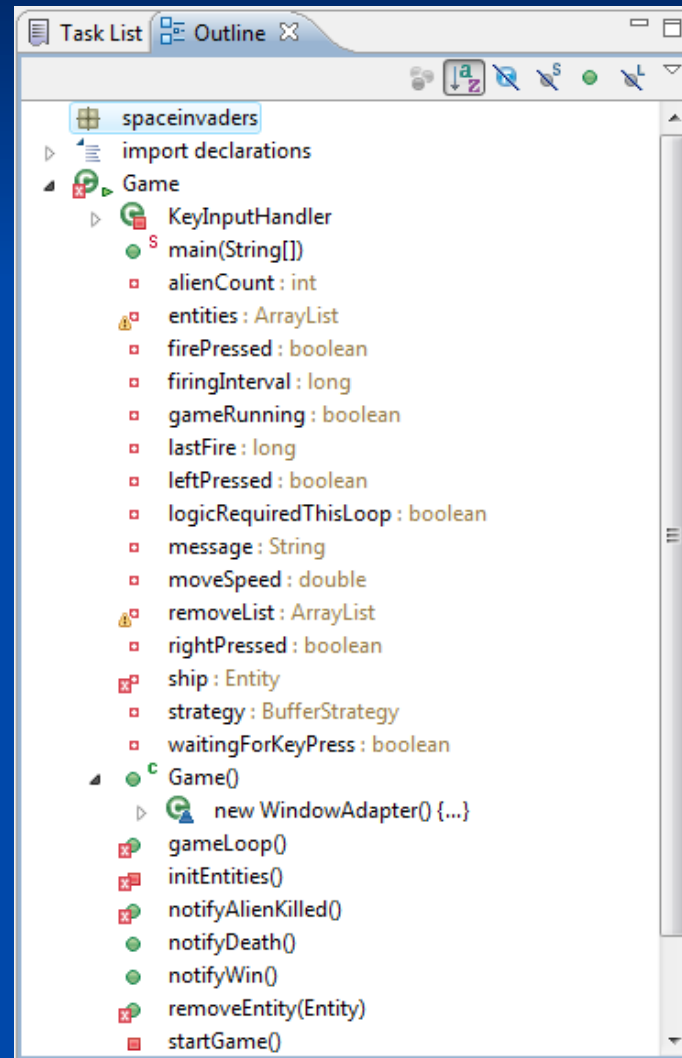
Common Features

- 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

Outline/Navigator



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

The screenshot displays a Java GUI Builder application interface. The main design area shows a window titled "NewJFrame.java" with a toolbar at the top. A lightbulb icon indicates a tip: "The Preview Design button (in the toolbar) enables you to test the design of the form." The design area contains a table with four columns labeled "Title 1" through "Title 4" and five rows. Below the table is a text input field with the value "0" and a spinner. Three buttons are visible: "jButton3" on the left, "jButton1" at the bottom left, and "jButton2" at the bottom right. A slider is positioned at the bottom center.

The **Navigator** panel on the right shows the "Members View" of the "NewJFrame :: JFrame" class. The members listed are:

- NewJFrame()
- initComponents()
- main(String[] args)
- jPanel1 : JPanel

The **Palette** panel on the right lists various Swing components:

- Panel, Tabbed Pane, Split Pane
- Scroll Pane, Tool Bar, Desktop Pane
- Internal Frame, Layered Pane
- Swing Controls**: Label, Button, Toggle Button, Check Box, Radio Button, Button Group, Combo Box, List, Text Field, Text Area, Scroll Bar, Slider, Progress Bar, Formatted Field, Password Field, Spinner, Separator, Text Pane, Editor Pane, Tree, Table
- Swing Menus**
- Swing Windows**

The **Other Components - Properties** panel at the bottom right is currently empty, displaying "<No Properties>".

Differences cont

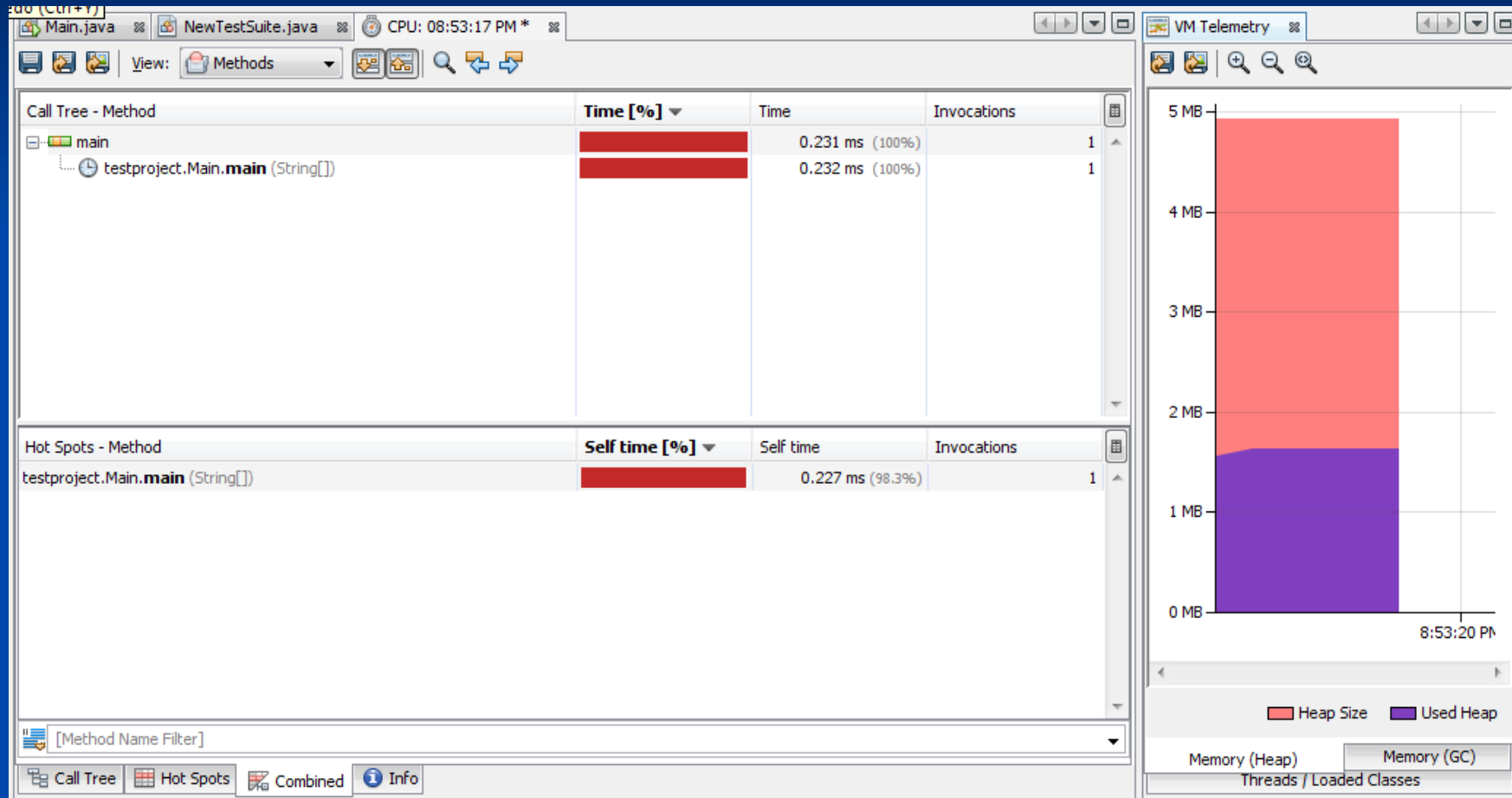
- Configuration Management Support
 - Both Eclipse and NetBeans have support for CVS
 - NetBeans comes with support for Subversion and Mercurial
 - Eclipse doesn't come with support Subversion or Mercurial
 - you can get plug-ins to correct this.

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.