Eclipse Tutorial and First Program

Java & Eclipse Platform

Oulu, Spring 2005

The Eclipse Platform is designed for building integrated development environments (IDEs) that can be used to create applications as diverse as web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans. In this tutorial we will show the (very) basic functionality of this program, which is call "The Platform" when talking about Java Developing Tools. Visit www.eclipse.org for further information.

Eclipse is released open source software license, so you can download for free from its website: http://www.eclipse.org/downloads/index.php. To install it, we need a Java Virtual Machine, avaliable in:

http://java.sun.com/j2se/1.5.0/download.jsp

We select JRE 5.0 Update 1 for our platform, now we are ready to download Eclipse and the Visual Editor:

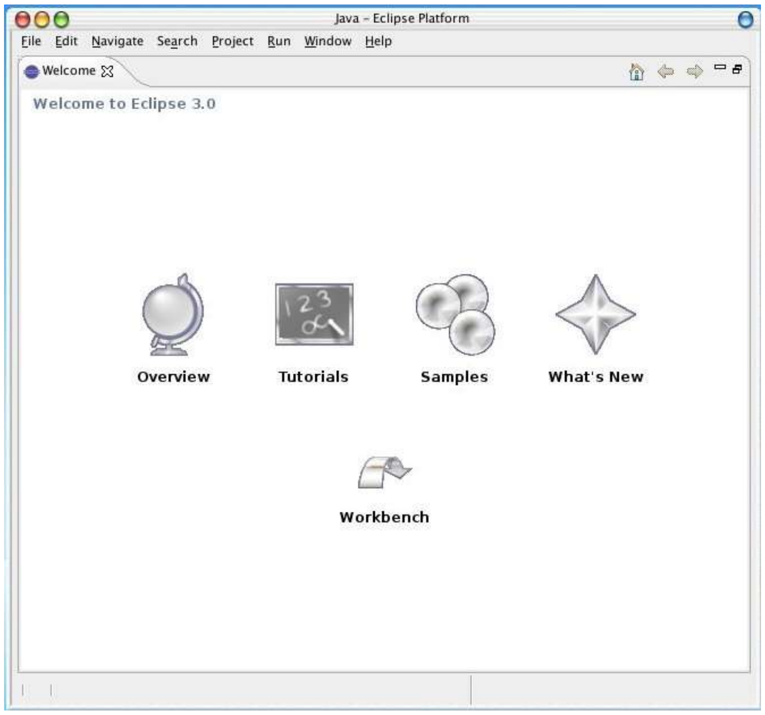
http://download.eclipse.org/downloads/drops/R-3.0.1-200409161125/index.php

http://download.eclipse.org/tools/ve/downloads/

Remember to install also the required plugins to run Visual Editor: Java Eclipse Modeling Framework (EMF) Model Runtime and GEF Runtime. To install the Java VM, click on the installer and follow the steps, for Eclipse and the VE:

1- DeCompress Eclipse into a folder (usually where the Programs Binaries are)
2- DeCompress VisualEditor, GEF and EMF taking care of putting the files in the right folder (plugins and features)
3-Run Eclipse /installation_folder/eclipse

If all the files are correct a Welcome window like this will appear:



These are the avaliable options for Eclipse, to create projetcs we should click on Workbench, then we will be in the following screen; Eclipse works with a Tab system, it means that each item is displayed in the same place all the time and we can select wich Tabs we want to display in each moment. The default view contains:

- 1 Navigator Tab: This is the place where our projects will be located

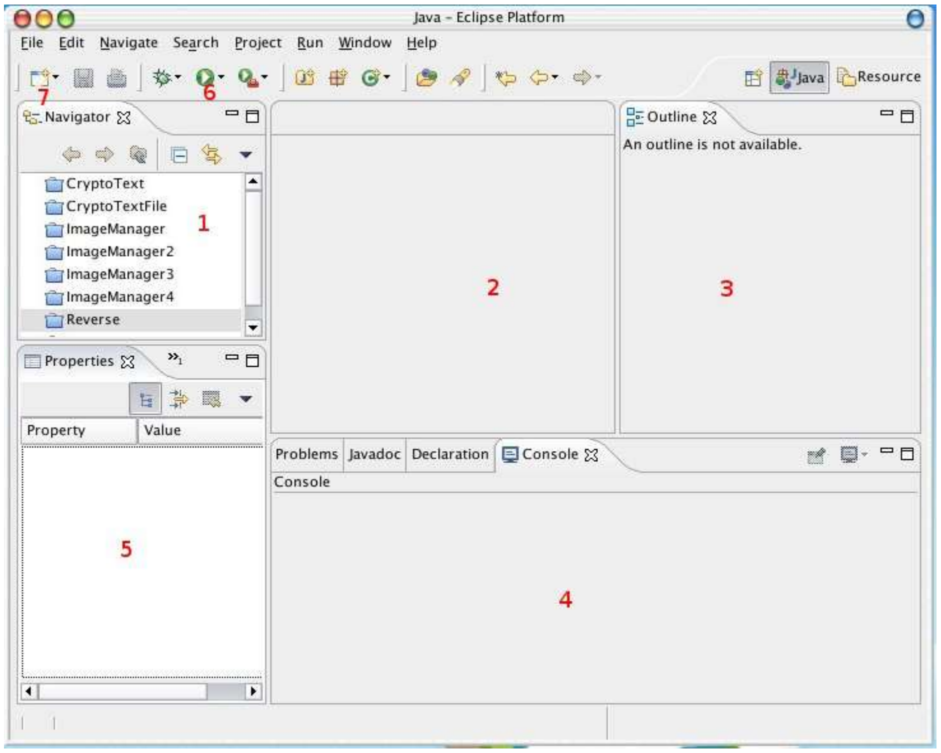
- 2 Java View: The code of our programs as the Visual Editor preview will be displayed here

- 3 Outline: All the functions and the includes of the code
- 4 Console: The Java Standard Output will be displayed here

- 5 Properties Tab: When using the Visual Editor, if we click on a item, the properties of this object could be changed by change the values in this tab

- 6 Run Button: We can save time by clicking this button instead of going to the menu. The first time that we run a project, Eclipse will prompt for the type and way to run it

- 7 New... To add new classes, visual classes we can click here instead of clicking on the File Menu

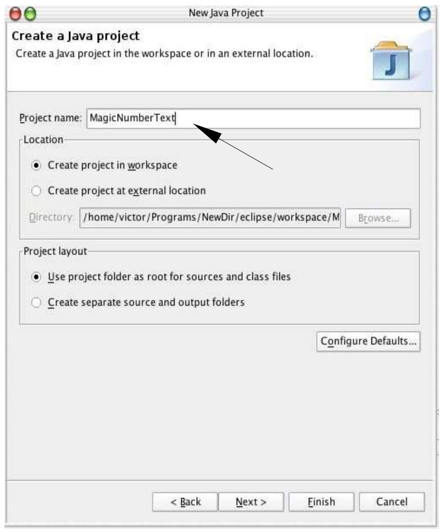


First Program: Good Luck Number

Lets create our first program with Eclipse, we will write both Command Line and GUI (Graphic User Interface) versions.

<u>First we will create the Command Line Version:</u>

1-Create a new project with Eclipse: Click on the 7 icon -> create a Java Project ->Finish


New Java Project

Create a Java project

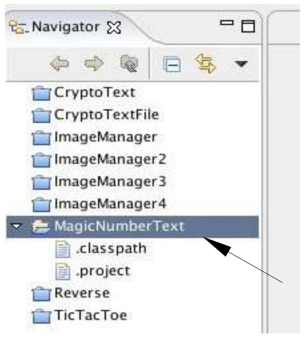
Create a Java project in the workspace or in an external location.

Project name: MagicNumberText

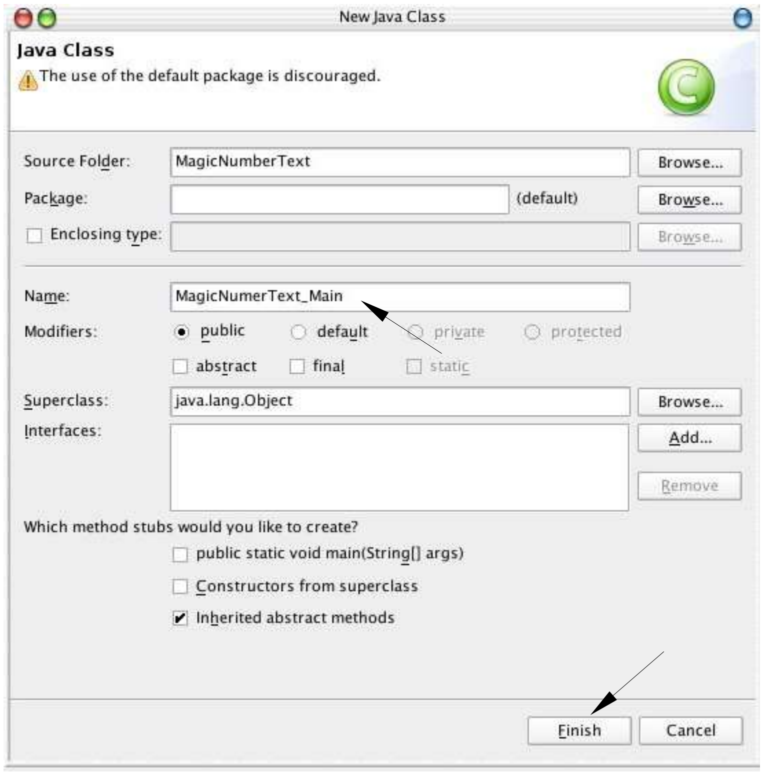
Location
- (●) Create project in workspace
- ( ) Create project at external location

Directory: /home/victor/Programs/NewDir/eclipse/workspace/M

Project layout
- (●) Use project folder as root for sources and class files
- ( ) Create separate source and output folders



When the projetc is created the name and the files will appear in the Navigator Tab,we can add the first Class: Again we click on the 7 icon and we will select the “Class” option, a window like the following will open:


New Java Class

Java Class

The use of the default package is discouraged.

Source Folder: MagicNumberText

Package: (default)

Enclosing type:

Name: MagicNumerText_Main

Modifiers: [✓ public] default private protected

abstract final static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

[✓ Inherited abstract methods]

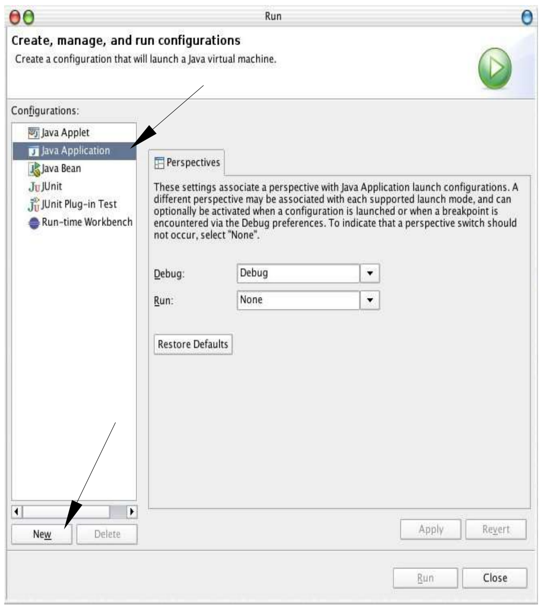
We should give a number and click: Finish. Now in the central part (Java View,2) we have the code of the Program, Eclipse adds a header with some information about the author and the data:

```
/*
 * Created on Feb 25, 2005
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */

/**
 * @author Victor Arroyo Valle
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class MagicNumerText_Main {

}
```

Now is time to add some code to our program, we will start with the typical Hello World!, if you have used Java before this shouldn't be too difficult:

```
public class MagicNumerText_Main {
          public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Now is time to run the program for the first time, to make it easy, press the icon 6 on the Eclipse toolbar; if its the first time that we run the program, a configuration window will appear prompting us about the type of Java program that we want to run:



Run

Create, manage, and run configurations

Create a configuration that will launch a Java virtual machine.

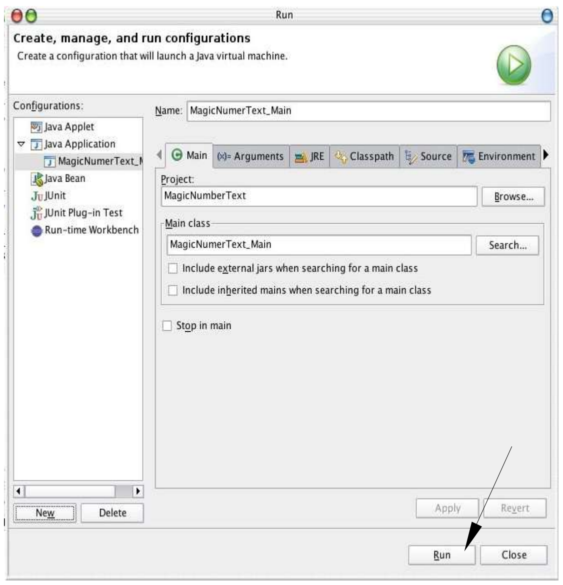
Configurations:

Perspectives

These settings associate a perspective with Java Application launch configurations. A different perspective may be associated with each supported launch mode, and can optionally be activated when a configuration is launched or when a breakpoint is encountered via the Debug preferences. To indicate that a perspective switch should not occur, select "None".

Debug: Debug

Run: None



Run

Create, manage, and run configurations

Create a configuration that will launch a Java virtual machine.

Configurations:

Name: MagicNumerText_Main

Project: MagicNumberText

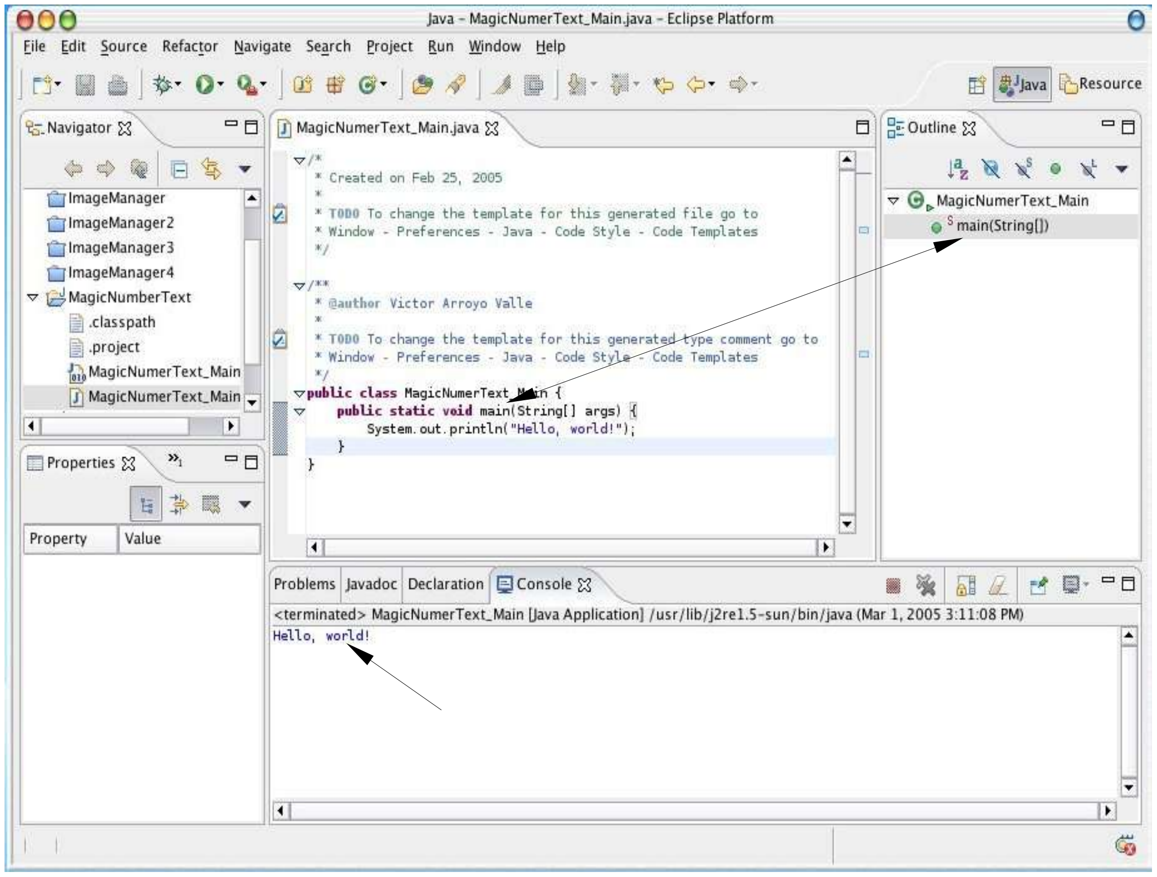
Main class: MagicNumerText_Main

☐ Include external jars when searching for a main class

☐ Include inherited mains when searching for a main class

☐ Stop in main

We will select Java Application, then New to create a new launch script and Run to execute the program. The rest of the options should be leaved by default. After clicking on Run the main Eclipse Screen will appear again with the result (if there arent compilation errors). We can read the Standard Output in the Console:



```
/*
 * Created on Feb 25, 2005
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */

/**
 * @author Victor Arroyo Valle
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class MagicNumerText_Main {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

```
<terminated> MagicNumerText_Main [Java Application] /usr/lib/j2re1.5-sun/bin/java (Mar 1, 2005 3:11:08 PM)
Hello, world!
```

Now is time to add some functionality to the program, this means incorporate Input/Output and interaction with the user. In Programming that means read the user input, do some kind of operation with this data and return a result. For our first program we will calculate the “Lucky Number”.

We need to promt for the user name, this will be the Input, the operation will be calculate a numerical value for each character of the name, add this values and get a one-digit number, which will be the luky number. This result should be returned to the user. Lets see it in a example:

Lets insert: Victor. In the Ascii Code, each character has a value that we can easily obtain by using the method charAt():

i	***0***	***1***	***2***	***3***	***4***	***5***
userName(i)	V	I	C	T	O	R
charAt(i)	118	105	99	116	111	114

So is quite easy obtain the Σ of each value:

charAt(i)	***118***	***105***	***99***	***116***	***111***	**114**
Σ (charAt(i))	663					

Now we should calculate just one number from this 663; to get it:

663 ->6+6+3 = 15
15-> 1+5 = **6 =>This is the Lucky number!!**

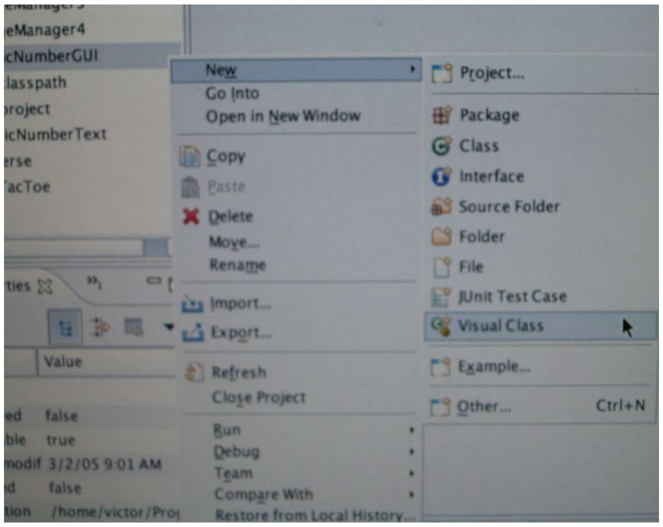
Lets write the code:

```
// Using the print() method lets us keep the cursor on the same line of output as our
printed text.
//This makes it look like a real prompt (instead of having the user's response appear on
the line below our prompt).

System.out.println("Welcome to the Lucky Number Program!");

//  Prompt the user to enter their name
System.out.print("Enter your name: ");

//  Open up standard input
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

// Create a variable where to save the Name
String userName = null;

//  To read the username from the command-line we need to use try/catch with the
readLine() method

try {
        userName = br.readLine();
        } catch (IOException ioe) {
                System.out.println("IO error trying to read your name!");
                System.exit(1);
        }

//Now in "userName" we have the User Input, lets make the operations
int i;
int aux=0;
```

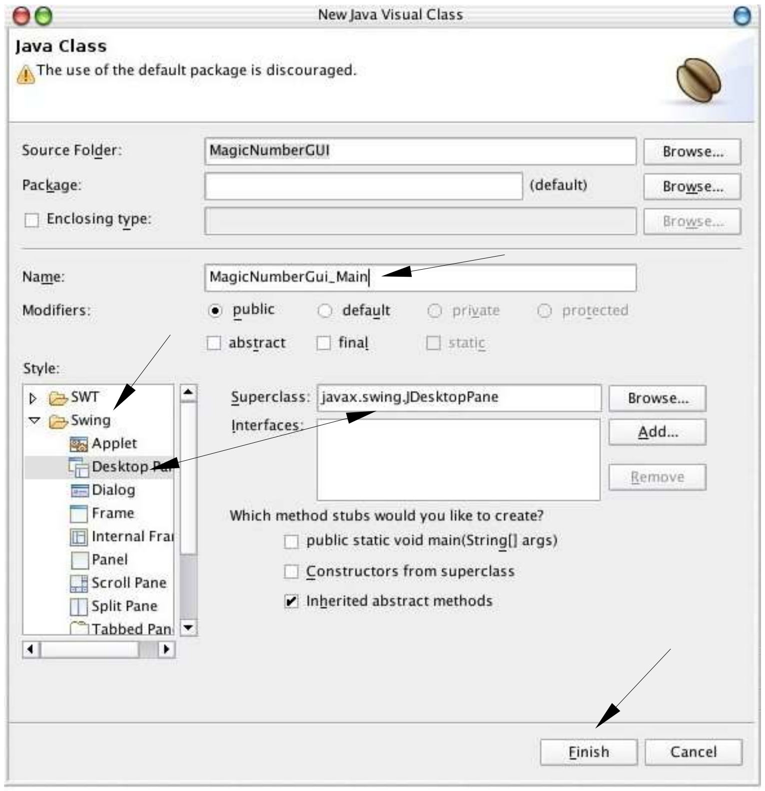
```
//"aux" calculates the the sum of the characters of the Name, ie for "victor","aux" is 663
for (i=0 ; i < userName.length() ; i++) {
        aux+=userName.charAt(i);
}

//The next step is calculate the result by adding every digit of aux: 663=6+6+3=15
=1+5=6

int result=0;
while (aux > 0 ) {
        result+= (aux%10);
        aux/=10;
        if (aux==0 && result>10) { aux=result; result=0; }
}

//For finishing, print the result
System.out.println(userName + ", your Lucky Number is... "  + result);
```

Now we will give our program a “real” Graphical user Interface. To make this task easy, we will use the Visual Editor included in Eclipse. First of all we should save our MagicNumerText Project, to do it, click with the right button on the name of the project and choose the option: ***Close Project***. Then we can import (ie to take it home) by selecting the option: **Export.**

To create the new Visual Project we have to follow the same first step than with the Text Version: Click on the 7 icon -> create a Java Project -> Finish. Now comes the diference, instead of creating a new Class we will add a Visual Class:



To perform this menu, click with the right button on the name of the projet in our case we will call it MagicNumberGUI and select New -> Visual Class; the following window will open:



Java Class

The use of the default package is discouraged.

Source Folder: MagicNumberGUI

Package: (default)

☐ Enclosing type:

Name: MagicNumberGui_Main

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Style:

Superclass: javax.swing.JDesktopPane

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

We will select Swing Style, because is powerful and widely used.the Swing API has 17 public packages:

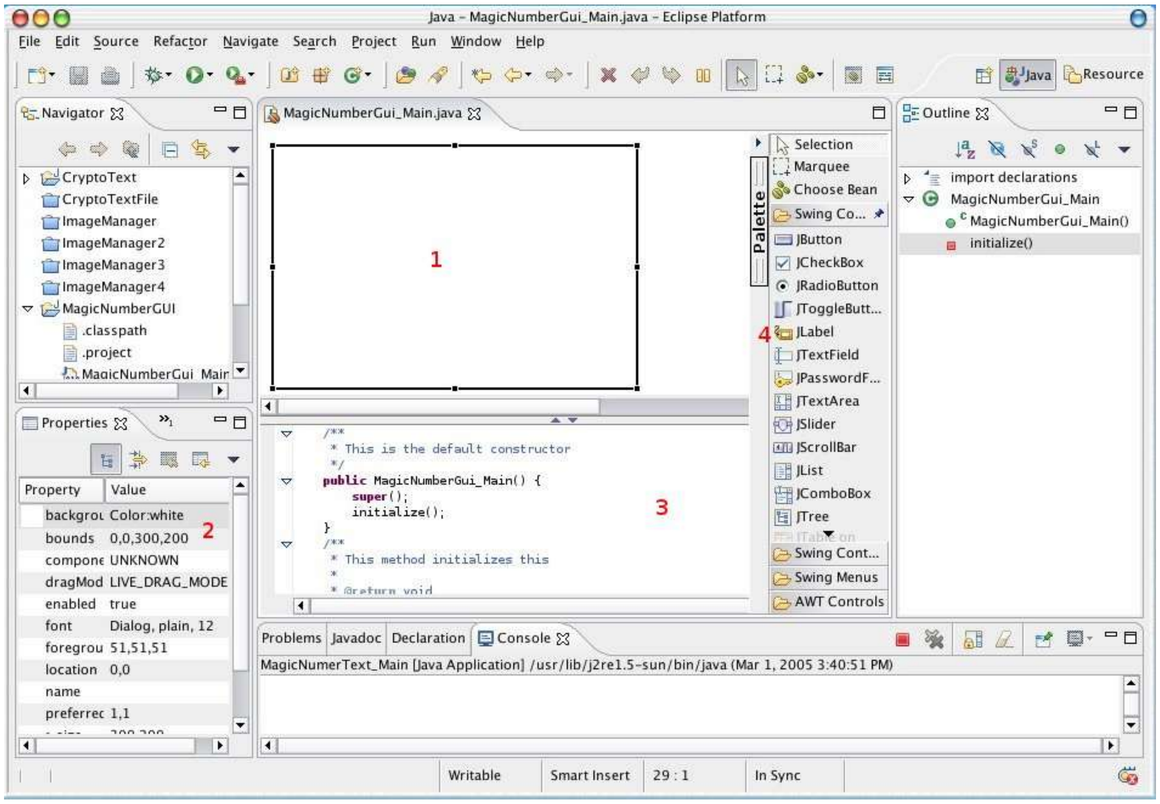
javax.accessibility	javax.swing.plaf	javax.swing.text.html
javax.swing	javax.swing.plaf.basic	javax.swing.text.parser
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.rtf
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.tree
javax.swing.event	javax.swing.table	javax.swing.undo
javax.swing.filechooser	javax.swing.text	

Enough for our Programs. After clicking on "Finish" Eclipse will show the standard editor view with all the required components to create Visual Aplications.

- **1** Preview, we can see how our program looks like while we are coding it
- **2** Properties View: To change the properties of each component, just click on it and

change the values in the properties box. The changes will be inmediatly updated in the code

- 3 Editor, here we will modify and add new code to our programs
- 4 Palette: All the Swing avaliable components are here, click on each one and draw directly over the preview window



Property	Value
backgrou	Color:white
bounds	0,0,300,200
compone	UNKNOWN
dragMod	LIVE_DRAG_MODE
enabled	true
font	Dialog, plain, 12
foregrou	51,51,51
location	0,0
name	
preferrec	1,1

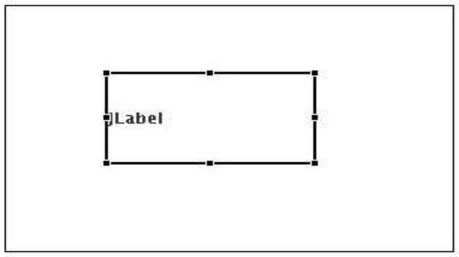
```
    /**
     * This is the default constructor
     */
    public MagicNumberGui_Main() {
        super();
        initialize();
    }
    /**
     * This method initializes this
     *
     * @return void
```

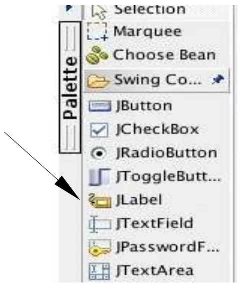
```
MagicNumerText_Main [Java Application] /usr/lib/j2re1.5-sun/bin/java (Mar 1, 2005 3:40:51 PM)
```

After creating the new Visual Class we can run the program inmediatly, because Eclipse has added the code required to make the project ready. Click on the Run icon and choose the “Run” option to create a new launching profile. For this visual projects we should select “Java Bean” and “New” to create a new configuration. To finish, click on Run to build and run the program. A empty window will appear in the screen; now is time to add some stuff to our empty window. The most easy component that we can add is a Label, select it in the Palette Menu:

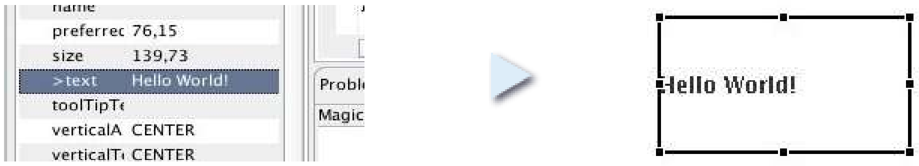
Swing Components -> JLabel

and “draw” the size of the label in the preview:

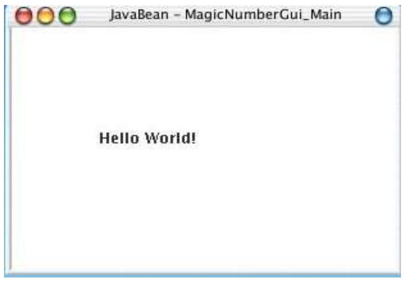




Then we can go to the Properties Tab and change the Text:



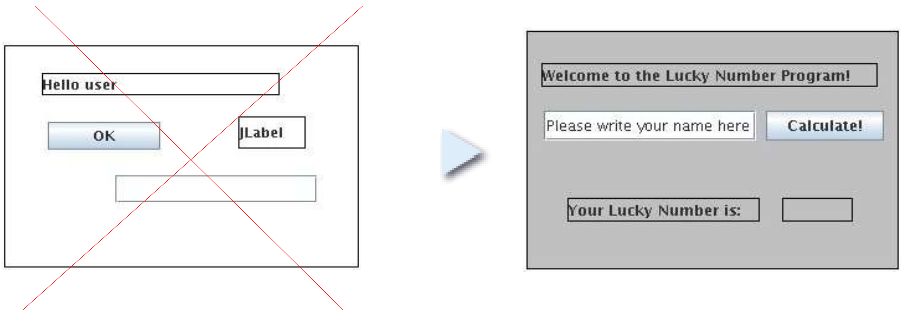
Now we can try to run the program and we will get our first Hello World! in Swing:



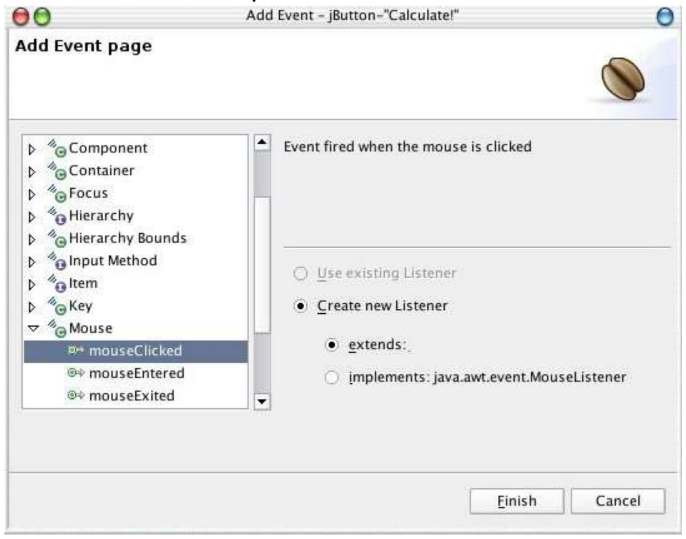
Lets start with the Magic Number program. In the text version, we asked the user to write his/her name. In Swing there's a component that makes it easy: JTextField. Select it on the Palette menu and "draw" it on the Preview. Eclipse will add the code for us:

```
private JTextField getJTextField() {
        if (jTextField == null) {
                jTextField = new JTextField();
                jTextField.setBounds(22, 55, 171, 25);
        }
        return jTextField;
}
```

Then we need an “Action!” button, like when we pressed the Return key in the Command Line version; Usually in the GUI programs it means a “OK” Button. Once more Swing provides us the Button Component to be used easily by selecting it from the Palette and drawing it over the Preview.
And for Finishing the GUI part, we need a place where to write the result, we can use a JLabel. The design of the screen should be clear and easy to understand:


Hello user
OK
JLabel
Welcome to the Lucky Number Program!
Please write your name here
Calculate!
Your Lucky Number is:

We can change the default values of the Visual Components: Size, Color, Location... by “playing” with the options in the Properties Tab. As much as you work in the Visual Interface, as much professional will it look like. Usually it means that the users will be happy of using our programs if our programs are nice to the users.
After the Design Part, comes the Coding phase. Now is time to retake our Command Line Version because the we have done the basic and the most “difficult” part of our program: The Algorithm.
Thinking about the Text Version, the program should execute the user action whenever the “Return” key pressed, this means that it should read the user input, process it and write it in the Standard Output. In the GUI version “Return” means action Button (in this case “Calculate!”. So we should make “things happen” whenever this button is pressed. In Java this means Events, to add an Event to a Button, we should click whith the right button on the Button and select the option Events -> Add Events:


Add Event - jButton-"Calculate!"

Add Event page

Event fired when the mouse is clicked

( ) Use existing Listener
(●) Create new Listener
(●) extends:
( ) implements: java.awt.event.MouseListener

Due this is a Mouse Event, we will select “mouseClicked” from the Mouse Event. After clicking on Finish, Eclipse will add the code:

```
private JButton getJButton() {
        if (jButton == null) {
                jButton = new JButton();
                jButton.setBounds(193, 67, 95, 25);
                jButton.setText("Calculate!");
                jButton.addMouseListener(new java.awt.event.MouseAdapter() {
                        public void mouseClicked(java.awt.event.MouseEvent e) {
                                System.out.println("mouseClicked()");
                        }
                });
        }
        return jButton;
}
```

If we run the program and we make “click” on the Calculate! button the Standard Output (=Console) will show the text: mouseClicked(). With this we can start adding code to perform the operation; to read the user input Swing provides a easier method:

```
String userName = jTextField.getText();
```

instead of the try/catch way of the Text Version:

```
//  Open up standard input
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

// Create a variable where to save the Name
String userName = null;

//  To read the username from the command-line we need to use try/catch with the
readLine() method
try {
        userName = br.readLine();
} catch (IOException ioe) {
        System.out.println("IO error trying to read your name!");
        System.exit(1);
}
```

Now as the last version in userName we have the information provided by the user, the algorithm is exactly the same than the text version. Though we have used the same variable, we can just copy and paste the code. If we want to write the result in the window instead of using the standard output, we will use the JLabel that we created before. By using the method setText(), the operation is quite easy:

```
jLabel1.setText("" + result);
```

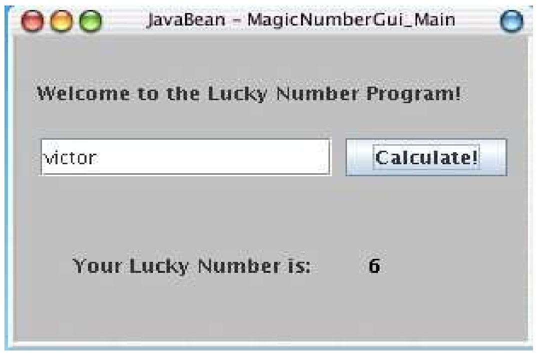
The comiles come from the setText method, it needs a string variable to be displayed, and we are providing an integer, so we can use the sum operator which is overloaded for the type integer to get a string in the same way that we would do with:

"he" + "llo" = "Hello"
"" + "hello" = "Hello"
"" + 123 = "123"

Lets take a final look at the code:

```
public void mouseClicked(java.awt.event.MouseEvent e) {
        //Read the user Input
        String userName = jTextField.getText();

        //Now in "userName" we have the User Input, lets make the operations
        int i;
        int aux=0;
        for (i=0 ; i < userName.length() ; i++) {
        //"aux" calculates the the sum of the characters of the Name, ie for "victor",
        //"aux" is 663
                aux+=userName.charAt(i);
        }

        //The next step is calculate the result by adding every digit of aux:
        //663=6+6+3=15=1+5=6
        int result=0;
        while (aux > 0 ) {
                result+= (aux%10);
                aux/=10;
                if (aux==0 && result>10) { aux=result; result=0; }
        }

        //Write it in the Result Box
        jLabel1.setText("" + result);
}
```

This is the result:


JavaBean - MagicNumberGui_Main
Welcome to the Lucky Number Program!
victor
Calculate!
Your Lucky Number is:
6

Good Luck!