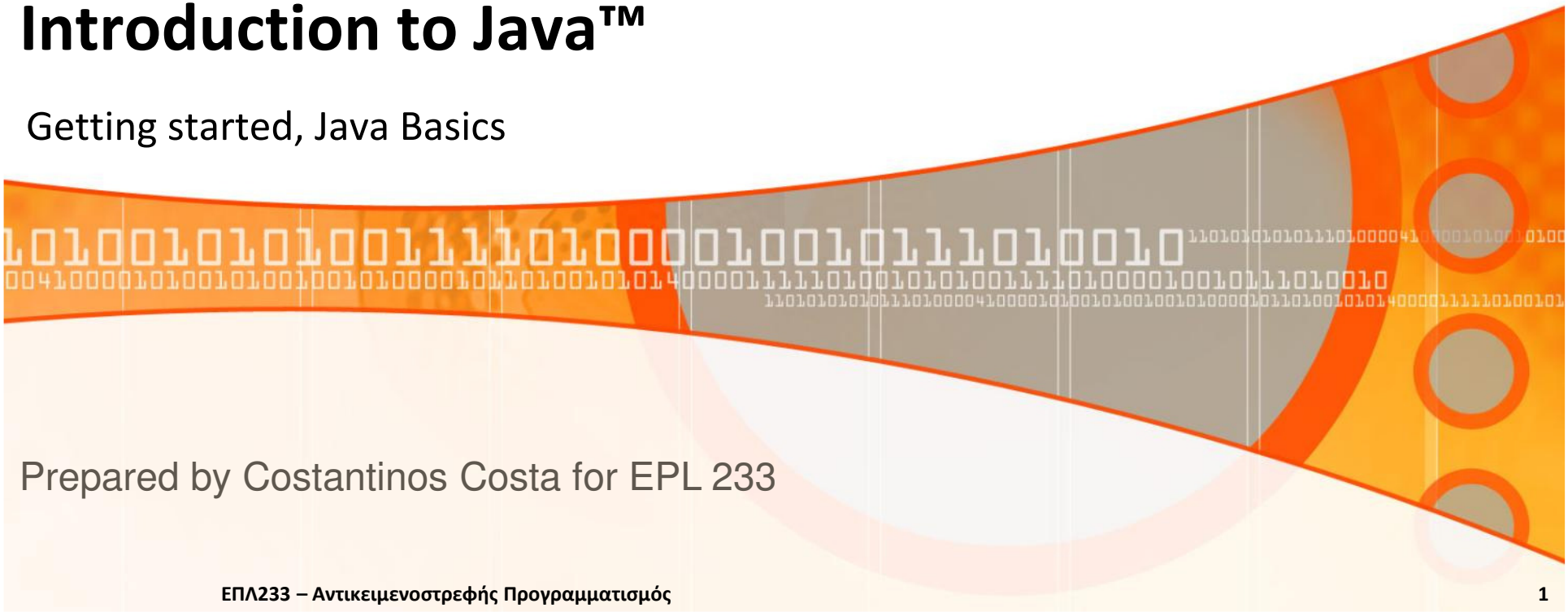
Introduction to Java™

Getting started, Java Basics



Prepared by Costantinos Costa for EPL 233

Creating a Java program

- Required software (minimum):
  - Java Development Kit for your platform
    (includes the Java compiler, Java Virtual Machine, Appletviewer, Libraries and other utilities)
  - A text editor
- Recommended software:
  - An IDE (Integrated Development Environment)
    - There are many IDEs available (e.g. NetBeans, Eclipse, etc.)
    - We will use Eclipse
- The CLASSPATH variable
  - Tells the Java compiler and Java virtual machine where the libraries are stored
  - It should include the current directory (.)
  - Usually set in a batch file for easy execution (more on this in a later lab)

Creating Your First Application - HelloWorldApp

To create this program, you will:

- Create a Java source file (*.java).
- Compile the source file into a bytecode file. The Java *compiler*, **javac**, takes your source file and translates its text into instructions that the *Java Virtual Machine* (Java VM) can understand.
- Run the program contained in the bytecode file. The Java VM is implemented by a Java *interpreter*, **java.** This interpreter takes your bytecode file and carries out the instructions by translating them into instructions that your computer can understand.

Create a Java Source File

- If you are NOT USING an IDE:
  - The Java files you create should be kept in a separate directory (with mkdir).
  - Start a text editor.
  - Type the code and store it in a file HelloWorldApp.java
- If you are USING an IDE:
  - Each IDE has its own procedures
  - Generally you create some sort of project and you add your classes (code) under that project
  - We will demonstrate this with Eclipse

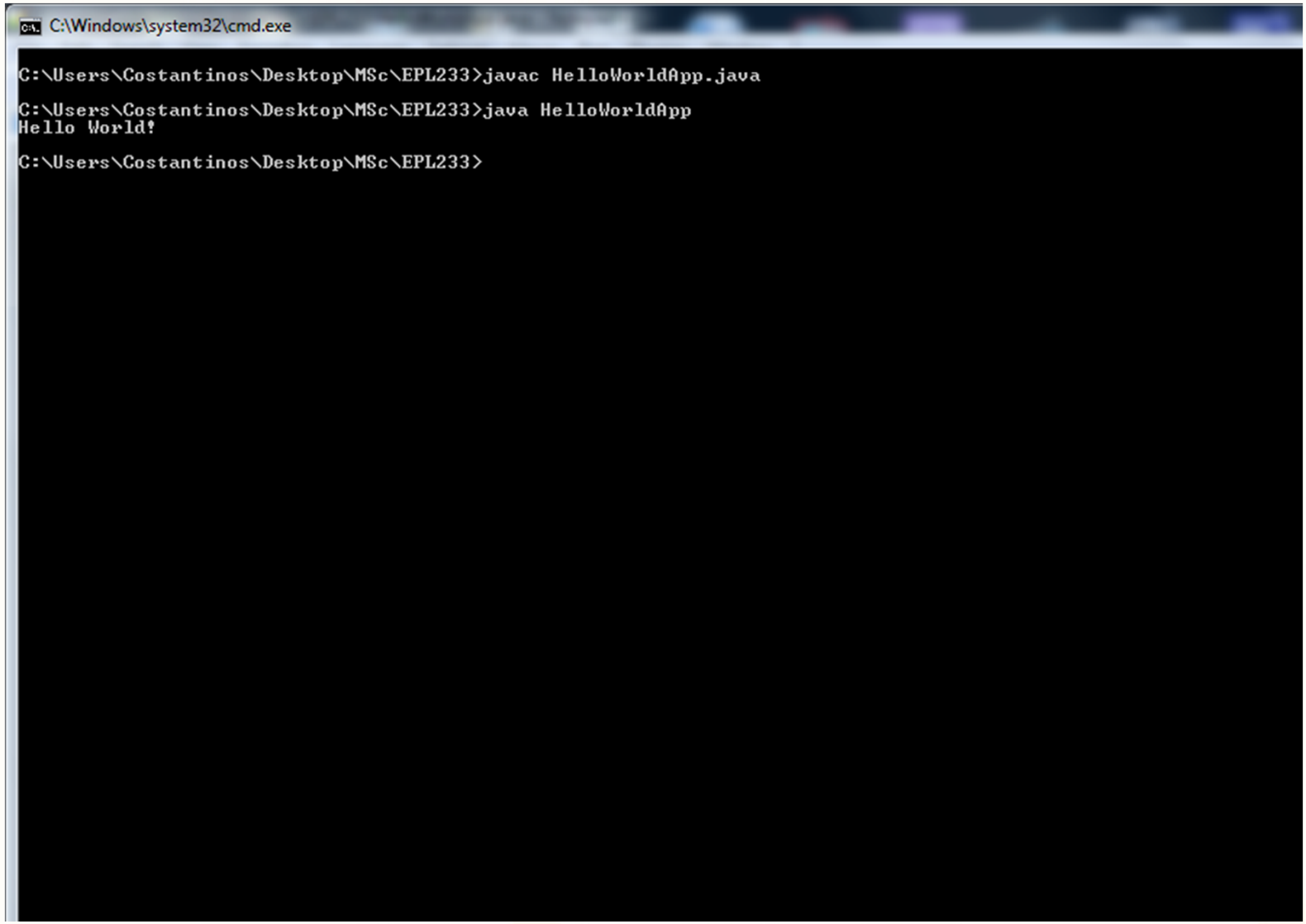
HelloWorldApp.java

```
/**
   * The HelloWorldApp class implements
   * an application that simply displays
   * "Hello World!" to the standard output.
   */
public class HelloWorldApp {
   public static void main(String[] args){
      System.out.println("Hello World!");
   }
}
```

Compile the Source File

- If you are NOT USING an IDE:
  - javac *HelloWorldApp.java*
  - If your prompt reappears without error messages, congratulations. You have successfully compiled your program.
  - A *HelloWorldApp.class* file is created
- If you are USING an IDE:
  - Depends on your IDE
  - Eclipse does background compiling as you write your code
    - Possible to instruct full code rebuild
  - Most (if not all) IDEs allow you to export your *.java and .class files*

Compile the Source File



What is Eclipse?

- Eclipse is an open source project
  - http://www.eclipse.org
  - Consortium of companies, including IBM
  - Launched in November 2001
  - Designed to help developers with specific development tasks
- Consists of four separate projects:
  - Eclipse Project
  - Eclipse Tools Project
  - Eclipse Technology Project
  - Eclipse Web Tools Platform Project

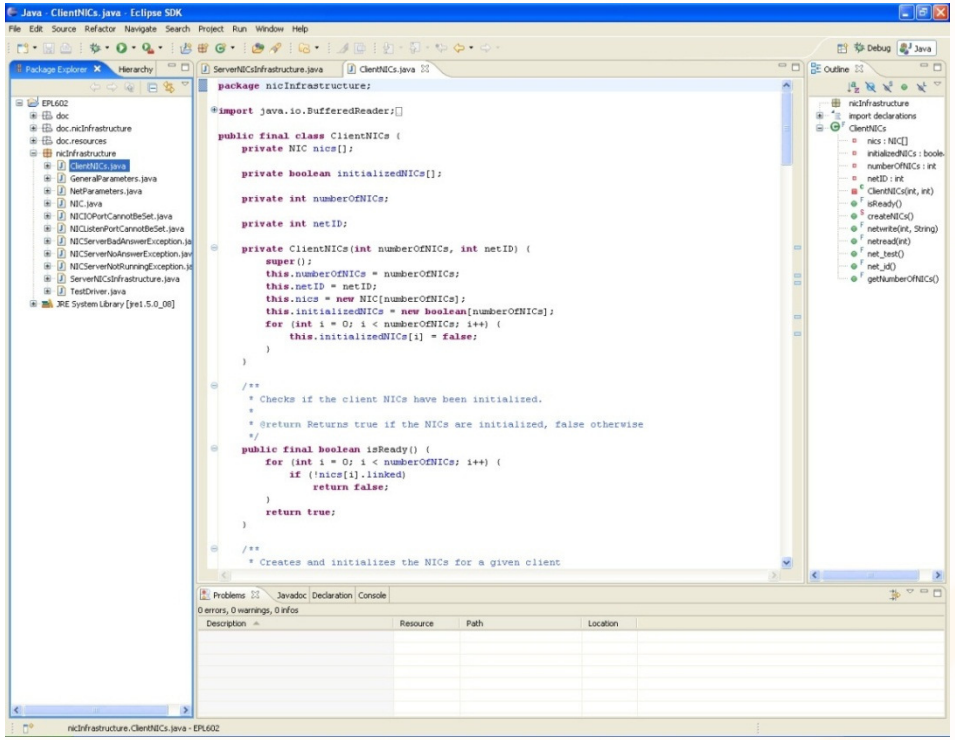
What you get with Eclipse

- A fully featured Java Development Toolkit (JDT)
  - Used for writing Java programs
  - Some features may be missing (e.g. limited GUI builder) but we are getting there
- The Plug-in Development Environment (PDE)
  - Used for extending Eclipse
  - Also used to convert eclipse to a product base for your own product

The Eclipse Platform Motivation

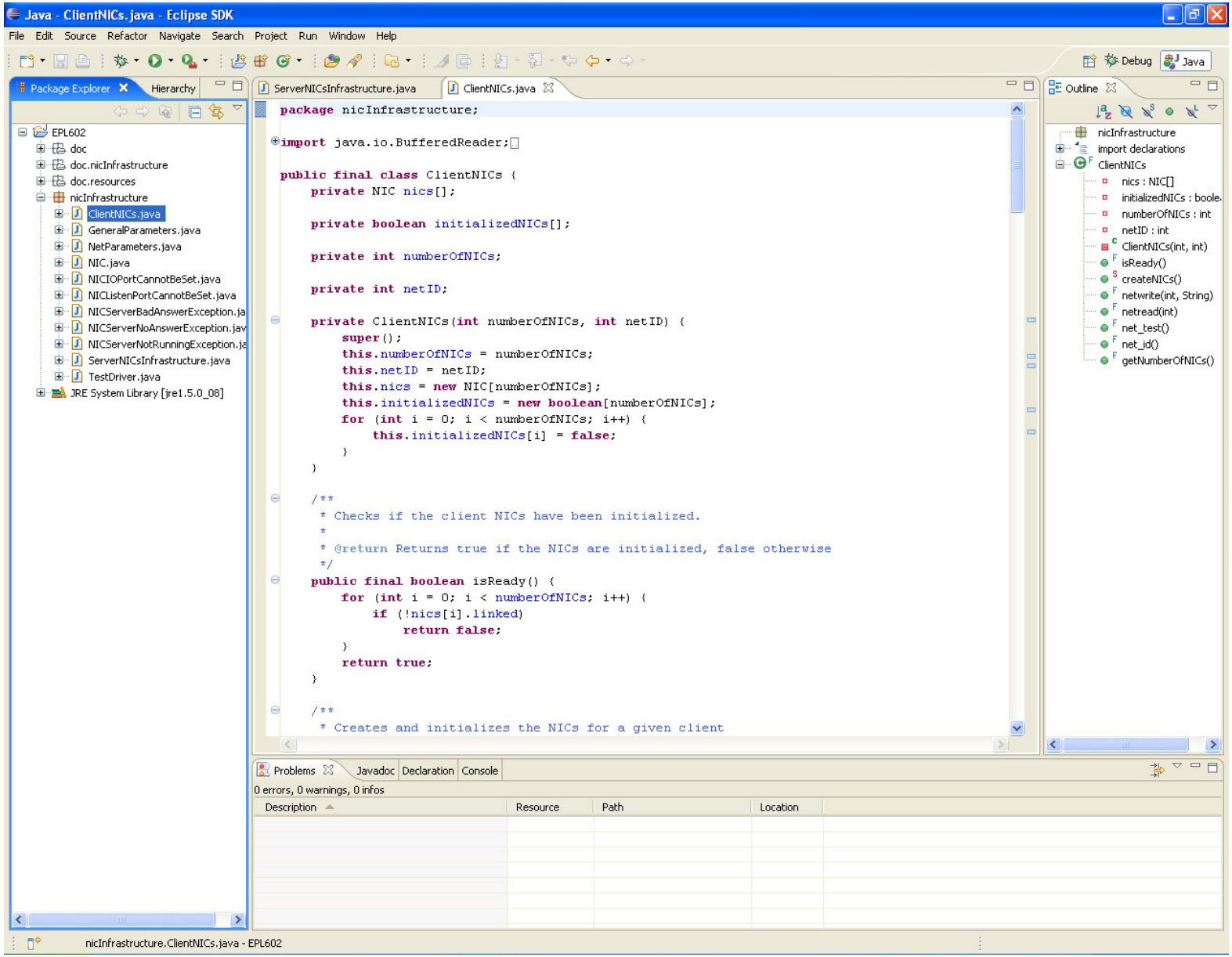
- Application development tools construction support
  - Independent tool vendors support
  - GUI and non-GUI application development support
- Numerous content types support
  - Java, HTML, C, XML, ...
  - Easy integration of tools
  - Use of Java language for writing the tools
  - Multiple operating systems support
- The Eclipse purpose is to provide the necessary services for integrating software developing tools
  - Implemented as plug-ins

Eclipse Workbench

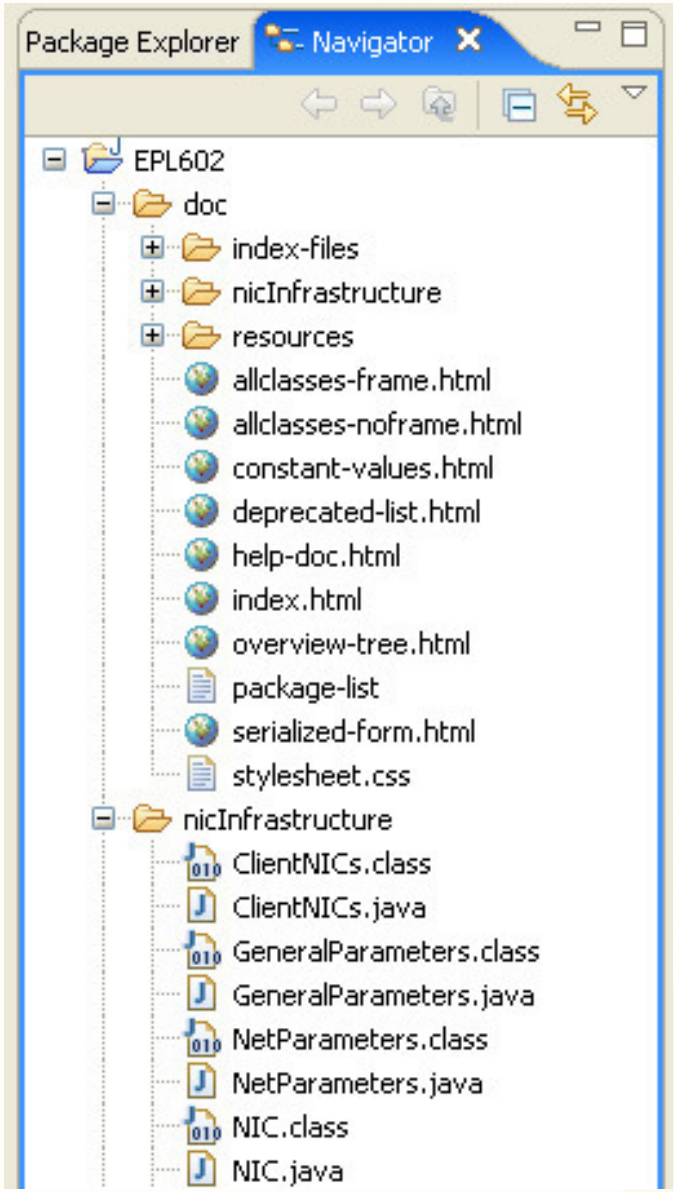


- Represents the desktop development environment
  - It contains set of tools for resource management
  - It provides common way of navigating through the resources
  - Organized into perspectives containing views
- Multiple workbenches can be opened at the same time

Eclipse

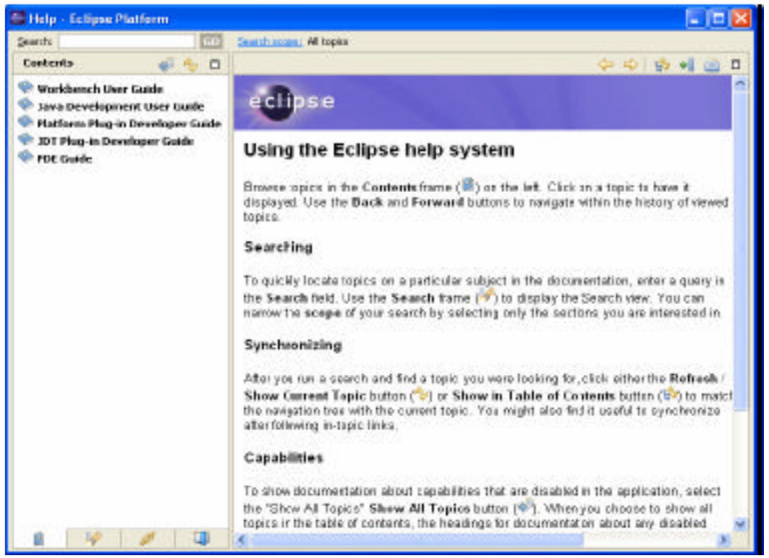

Java - ClientNICs.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
Package Explorer
Hierarchy
EPL602
doc
doc.nicInfrastructure
doc.resources
nicInfrastructure
ClientNICs.java
GeneralParameters.java
NetParameters.java
NIC.java
NICIOPortCannotBeSet.java
NICListenPortCannotBeSet.java
NICServerBadAnswerException.ja
NICServerNoAnswerException.jav
NICServerNotRunningException.ja
ServerNICsInfrastructure.java
TestDriver.java
JRE System Library [jre1.5.0_08]
ServerNICsInfrastructure.java
ClientNICs.java
package nicInfrastructure;
import java.io.BufferedReader;
public final class ClientNICs {
private NIC nics[];
private boolean initializedNICs[];
private int numberOfNICs;
private int netID;
private ClientNICs(int numberOfNICs, int netID) {
super();
this.numberOfNICs = numberOfNICs;
this.netID = netID;
this.nics = new NIC[numberOfNICs];
this.initializedNICs = new boolean[numberOfNICs];
for (int i = 0; i < numberOfNICs; i++) {
this.initializedNICs[i] = false;
}
}
/**
* Checks if the client NICs have been initialized.
*
* @return Returns true if the NICs are initialized, false otherwise
*/
public final boolean isReady() {
for (int i = 0; i < numberOfNICs; i++) {
if (!nics[i].linked)
return false;
}
return true;
}
/**
* Creates and initializes the NICs for a given client
Outline
nicInfrastructure
import declarations
ClientNICs
nics : NIC[]
initializedNICs : boole
numberOfNICs : int
netID : int
ClientNICs(int, int)
isReady()
createNICs()
netwrite(int, String)
netread(int)
net_test()
net_id()
getNumberOfNICs()
Problems
Javadoc
Declaration
Console
0 errors, 0 warnings, 0 infos
Description
Resource
Path
Location
nicInfrastructure.ClientNICs.java - EPL602

Workspace



- Represents users data
- It is a set of user defined resources
  - Files
    - Contain arbitrary number of bytes
  - Folders
    - Contain other folders or files
- Projects
  - Collections of files and folders

Help system



- An extensible documentation system
- Used for creating and publishing documentation
- There are two different documentation styles:
  - Help style documentation is published in the user guide
  - API documentation is published in the programmer guide
- Help content is in HTML format
- Help navigation is in XML format

Team suport

- Provides support for:
  - Versioning
  - Configuration management
  - Integration with team repository
- Includes a client for Concurrent Versions System (CVS)
- Allows team repository provider to hook into the environment
  - Team repository providers specify how to intervene with resources
- Has optimistic and pessimistic locking support

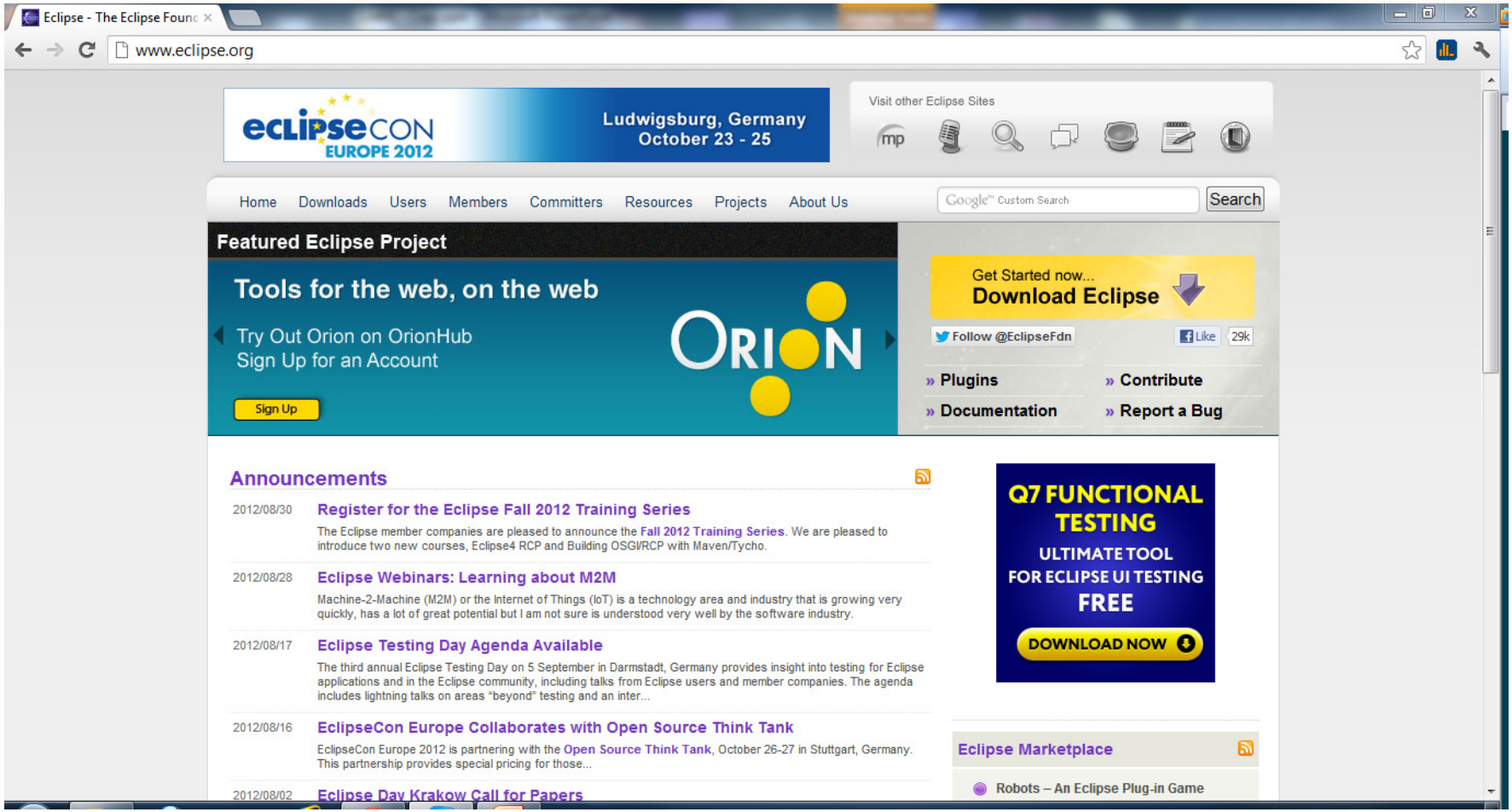
How is Eclipse Used?

- As an IDE - Integrated Development Environment
  - Java Development Tooling (JDT) is used for building Java code
  - Provides a set of workbench plug-ins for manipulating Java code
    - Java projects, packages, classes, methods, ...
  - Java compiler is built in
    - Used for compiling Java code
    - Creates errors (special markers of code) if compilation fails
- As a product base
  - Its flexible architecture used as a product framework
    - Reuse plug-in architecture
    - Create new plug-ins
    - Customize the environment

Language and platform neutral

- Eclipse is (programming) language neutral
  - It is used mainly for as a Java IDE but…
  - … there are plug-ins that serve as C/C++, Cobol and C# IDEs
- Eclipse is (human) language neutral
  - The plug-in mechanism allows to add different languages (through *plug-in fragments*)
- Even though written entirely in Java, Eclipse is not strictly platform neutral
  - This is due to the fact that Eclipse uses the operating system's native graphics
    - Only available to platforms where SWT has been ported

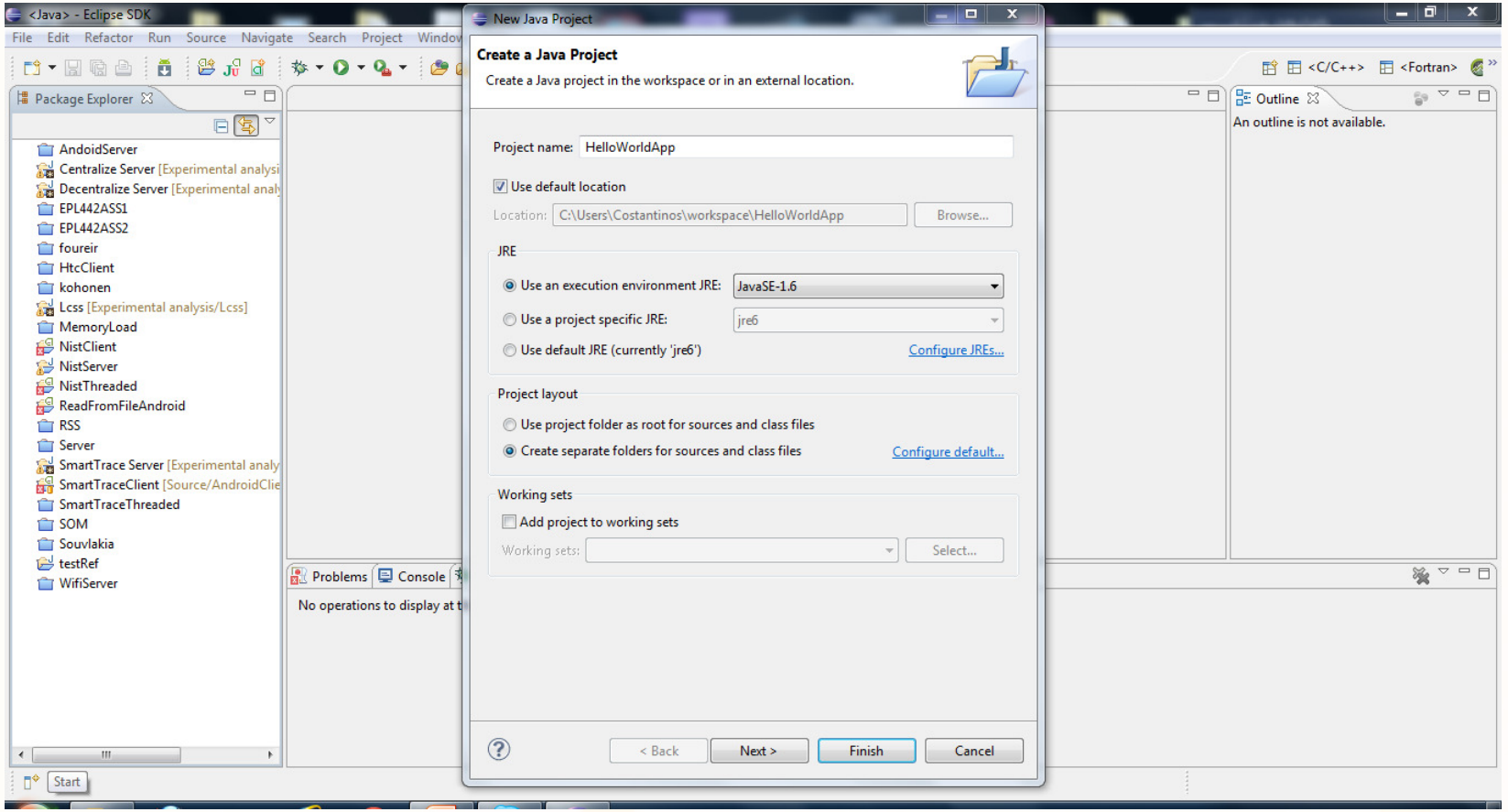
Getting Eclipse



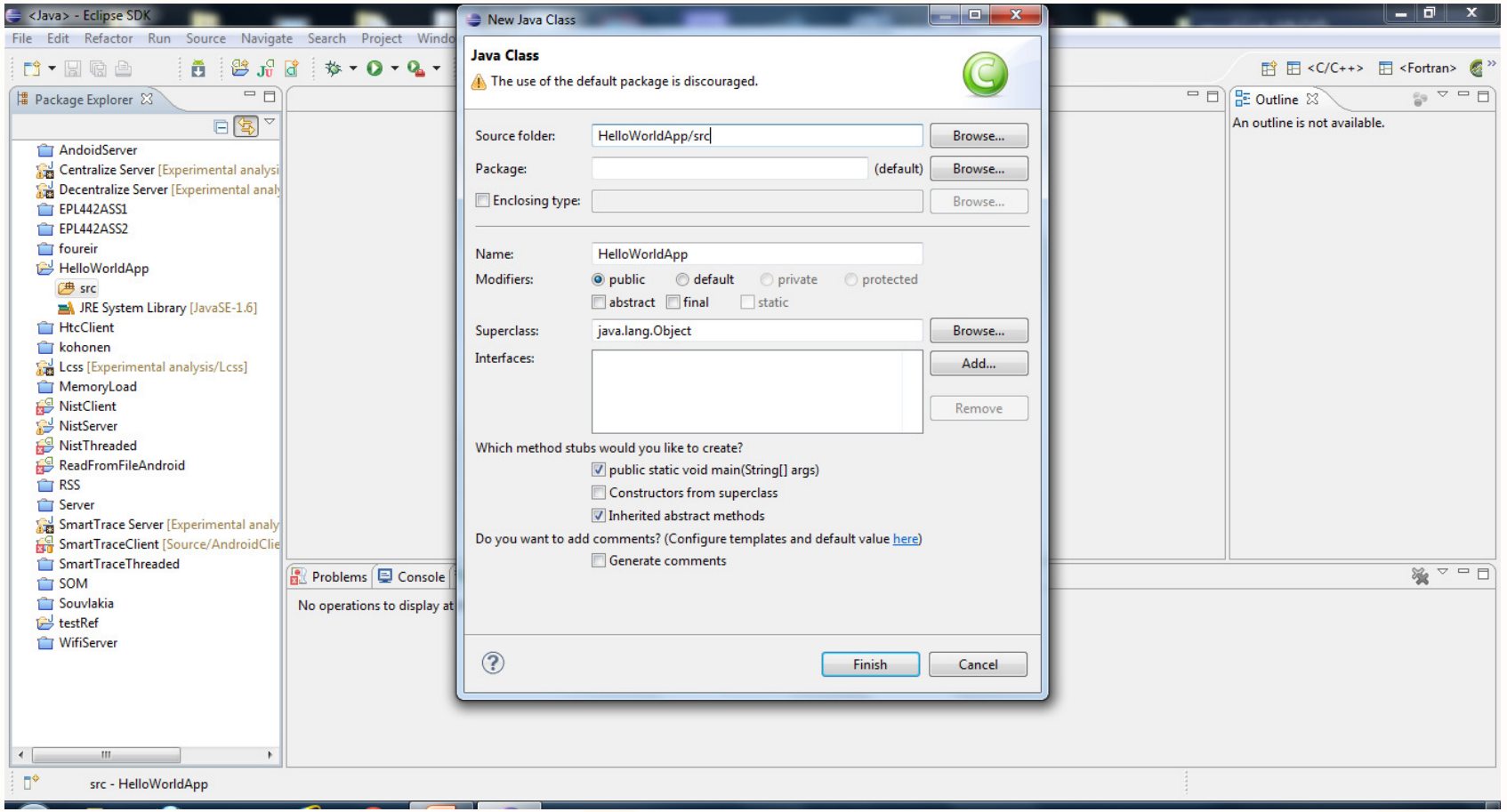
http://www.eclipse.org

- Main point for finding Eclipse resources
  - Downloads
  - Articles
  - Discussion groups
  - Bugs
- Contains various resources for using Eclipse
- Contains references to other Eclipse related sites

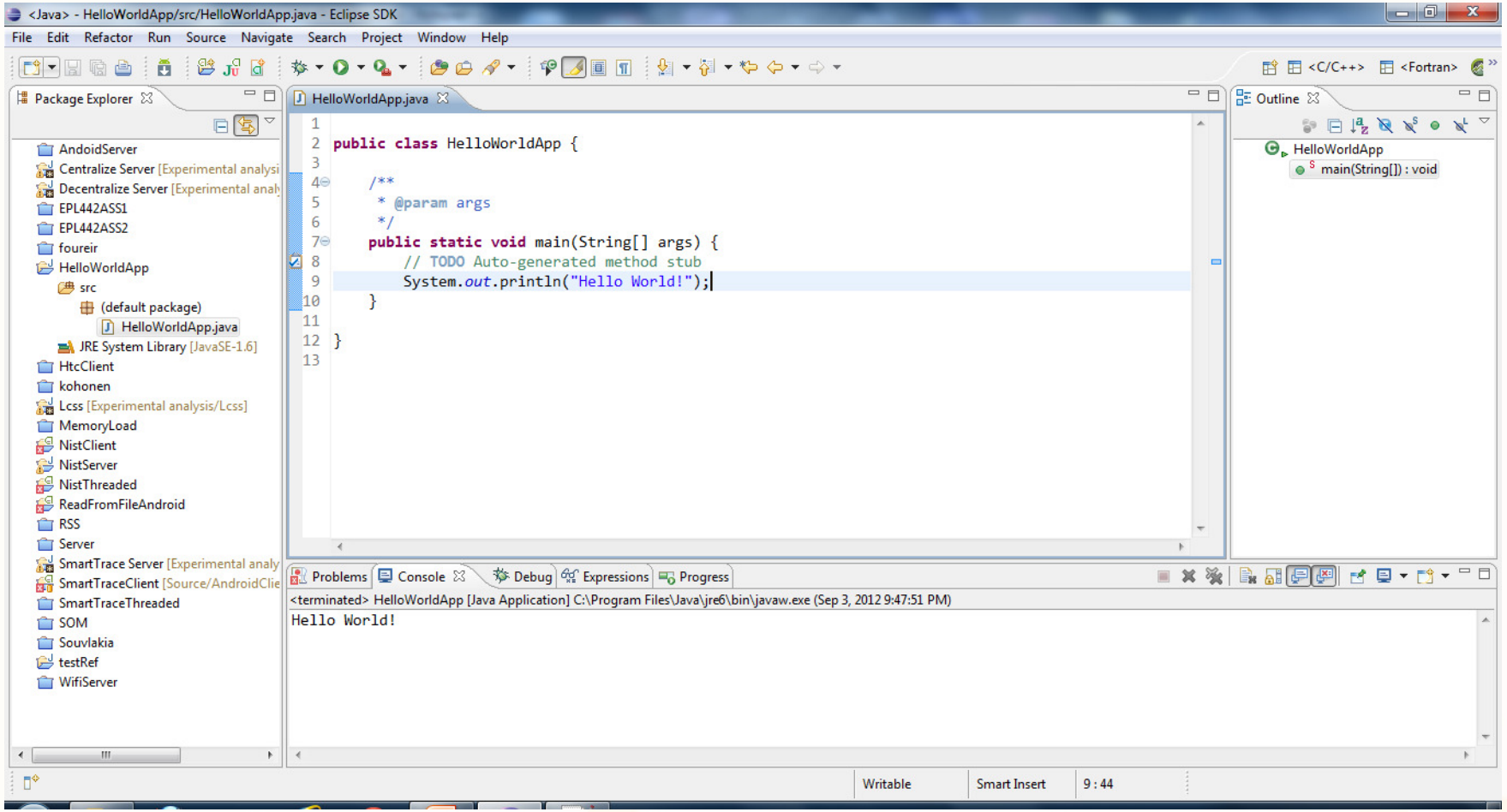
Create new Application



Create a new class



Run it


<Java> - HelloWorldApp/src/HelloWorldApp.java - Eclipse SDK
File Edit Refactor Run Source Navigate Search Project Window Help
Package Explorer
AndoidServer
Centralize Server [Experimental analysi
Decentralize Server [Experimental anal
EPL442ASS1
EPL442ASS2
foureir
HelloWorldApp
src
(default package)
HelloWorldApp.java
JRE System Library [JavaSE-1.6]
HtcClient
kohonen
Lcss [Experimental analysis/Lcss]
MemoryLoad
NistClient
NistServer
NistThreaded
ReadFromFileAndroid
RSS
Server
SmartTrace Server [Experimental analy
SmartTraceClient [Source/AndroidClie
SmartTraceThreaded
SOM
Souvlakia
testRef
WifiServer
HelloWorldApp.java
1
2 public class HelloWorldApp {
3
4 /**
5 * @param args
6 */
7 public static void main(String[] args) {
8 // TODO Auto-generated method stub
9 System.out.println("Hello World!");
10 }
11
12 }
13
Outline
HelloWorldApp
main(String[]) : void
Problems Console Debug Expressions Progress
<terminated> HelloWorldApp [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Sep 3, 2012 9:47:51 PM)
Hello World!
Writable
Smart Insert
9 : 44