

Corso di Laurea in Ingegneria Informatica e Automatica
A. A. 2016/2017

Programmazione orientata agli oggetti

Luca Iocchi, Massimo Mecella

Esercitazione 1



Sommario

- Programma Java
- Ciclo di compilazione in Java
- IDE
 - Esempio Eclipse
 - Esempio NetBeans
- API Java
 - Esempio classe String

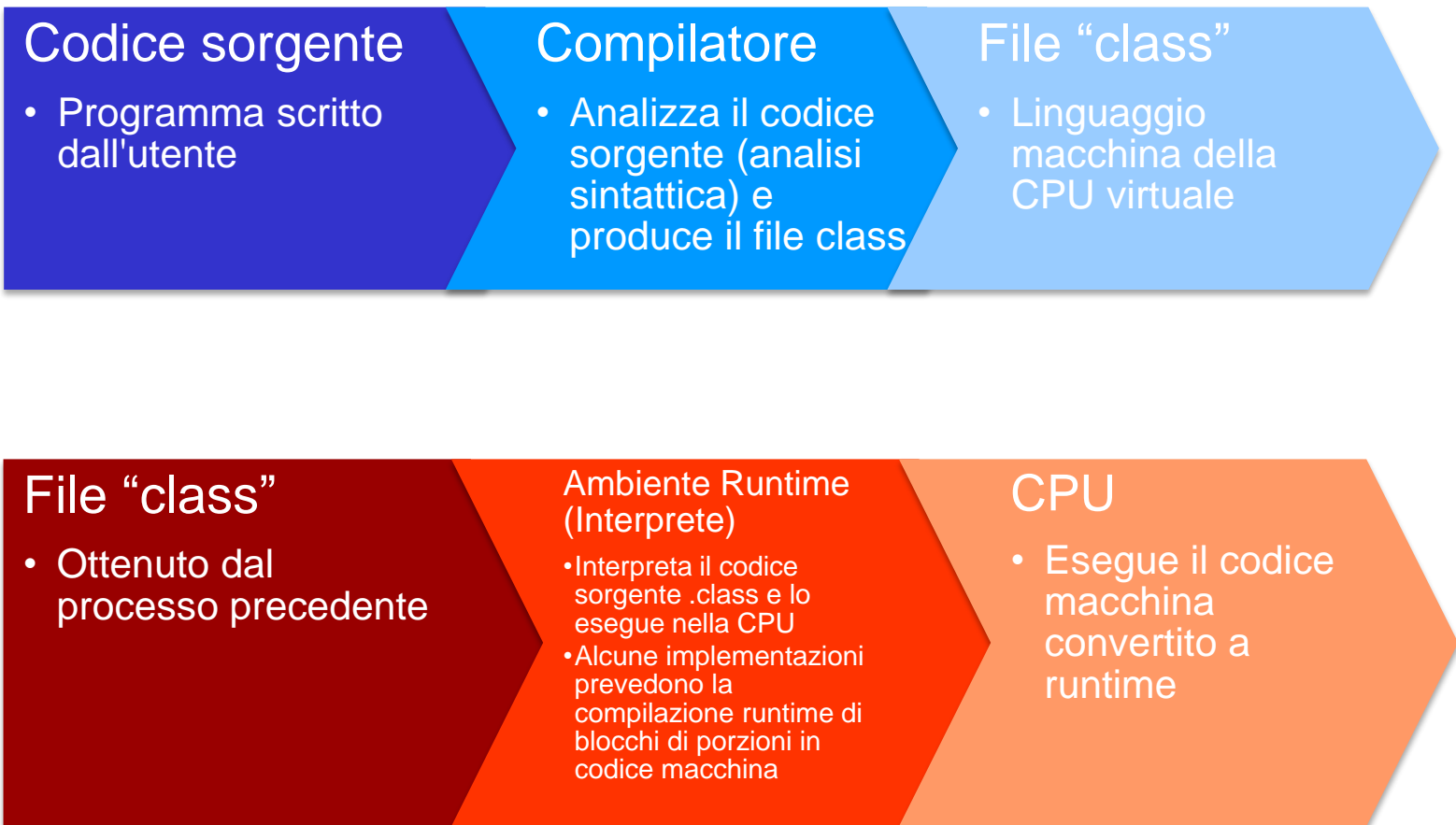


Programma Java

- Viene scritto il **codice sorgente** in accordo alla sintassi propria del linguaggio
- Il codice sorgente viene analizzato dal compilatore del linguaggio il quale produce dei file con estensione '.class' contenente bytecode.
- Il bytecode prodotto viene interpretato dall'interprete Java ed il file specificato viene eseguito (deve contenere un metodo `main()`).
Eventuali altri file .class necessari vengono caricati a runtime (collegamento dinamico)

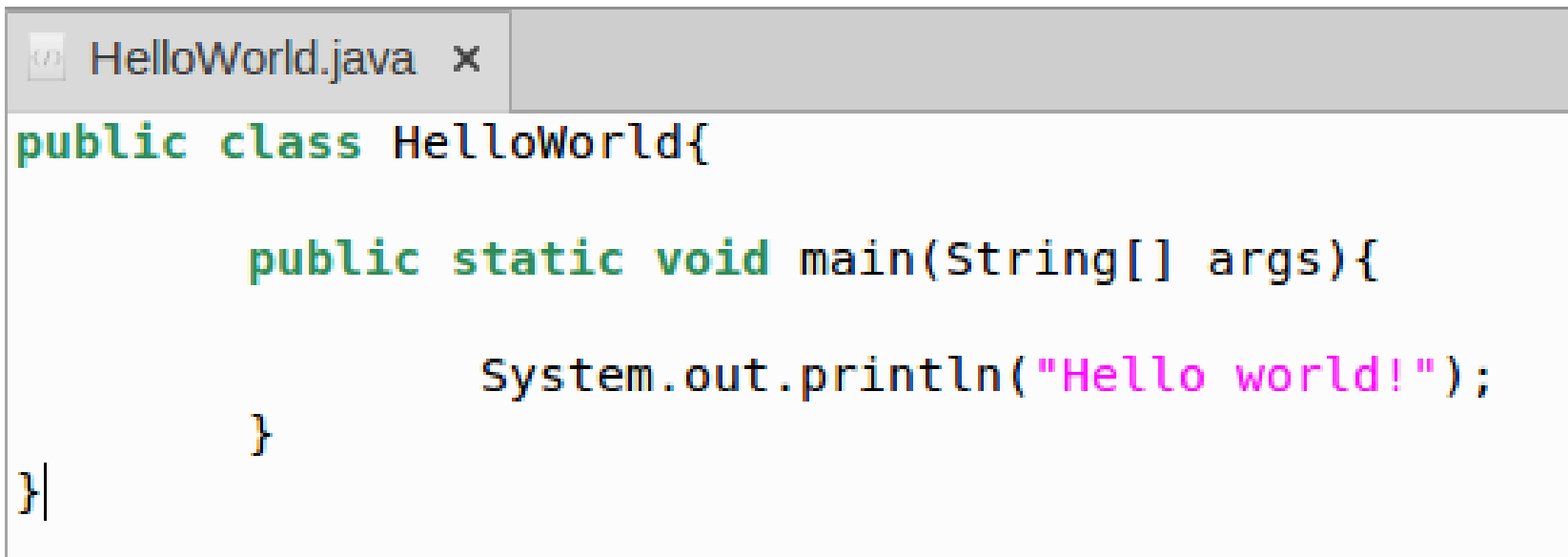


Programma Java



Programma Java Esempio

Consideriamo il classico programma HelloWorld:
Scriviamo il codice sorgente in un file di testo che chiamiamo *'HelloWorld.java'*



```

HelloWorld.java x
public class HelloWorld{

    public static void main(String[] args){

        System.out.println("Hello world!");

    }

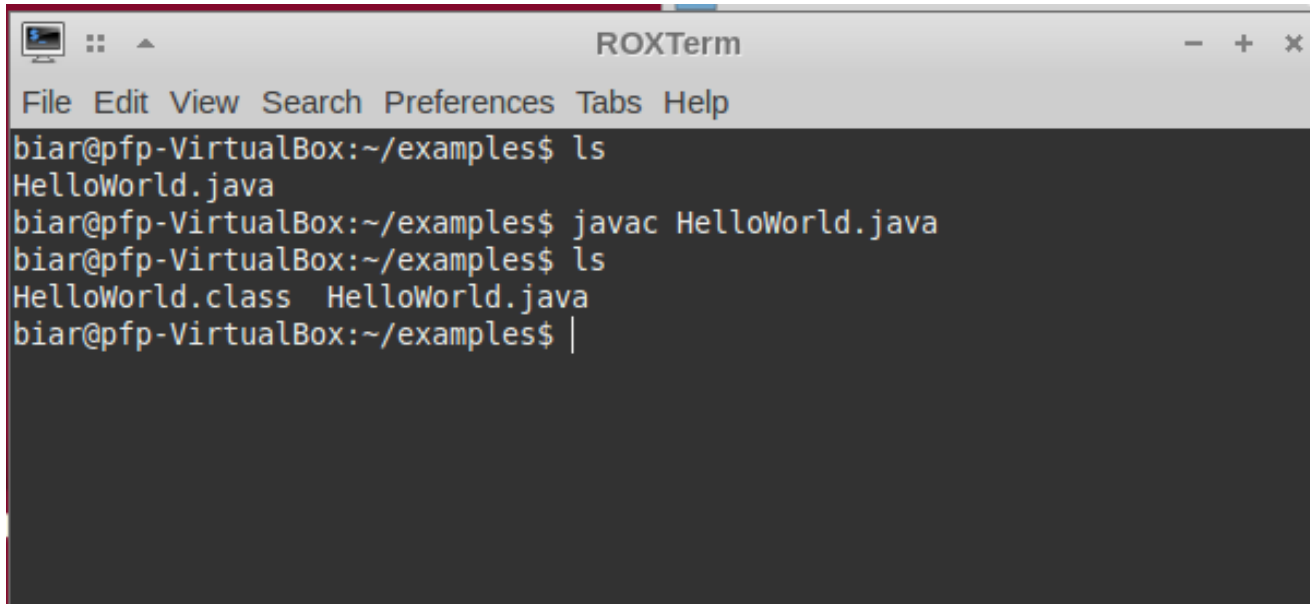
}
```



Programma Java Esempio

Per compilare il programma, posizionarsi da terminale nella directory in cui è contenuto il file ed eseguire il comando:

javac HelloWorld.java

A screenshot of a terminal window titled "ROXTerm". The window has a menu bar with "File", "Edit", "View", "Search", "Preferences", "Tabs", and "Help". The terminal shows the following commands and output:

```
biar@pfp-VirtualBox:~/examples$ ls
HelloWorld.java
biar@pfp-VirtualBox:~/examples$ javac HelloWorld.java
biar@pfp-VirtualBox:~/examples$ ls
HelloWorld.class HelloWorld.java
biar@pfp-VirtualBox:~/examples$ |
```

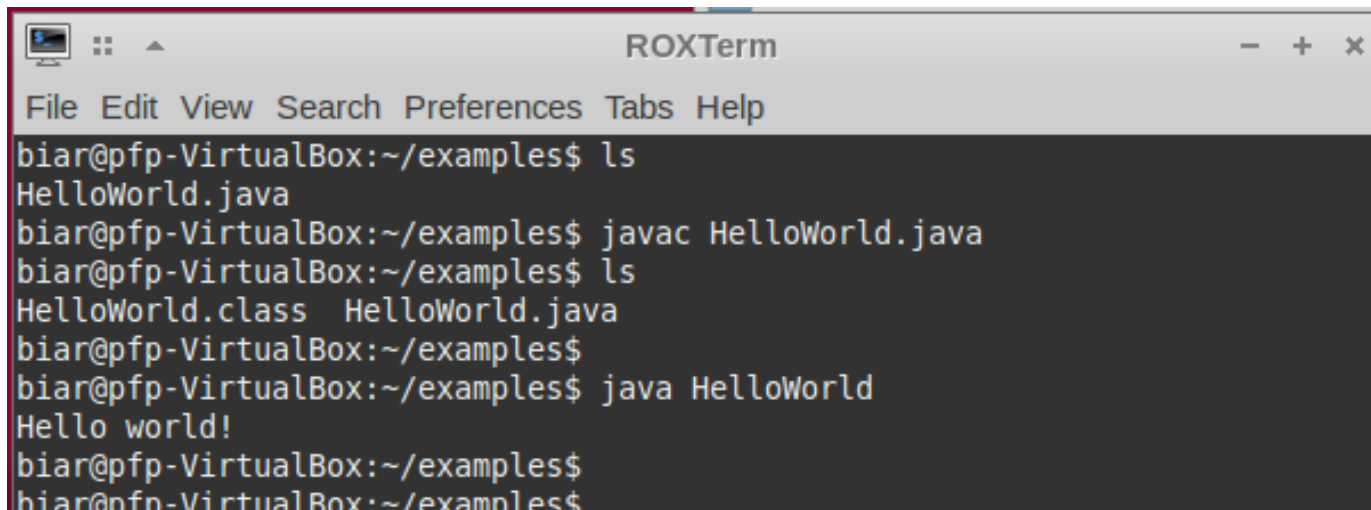
In questo modo viene prodotto il file *HelloWorld.class*



Programma Java Esempio

Ora si può eseguire il programma con il comando:

java HelloWorld



```
ROXTerm
File Edit View Search Preferences Tabs Help
biar@pfp-VirtualBox:~/examples$ ls
HelloWorld.java
biar@pfp-VirtualBox:~/examples$ javac HelloWorld.java
biar@pfp-VirtualBox:~/examples$ ls
HelloWorld.class HelloWorld.java
biar@pfp-VirtualBox:~/examples$
biar@pfp-VirtualBox:~/examples$ java HelloWorld
Hello world!
biar@pfp-VirtualBox:~/examples$
biar@pfp-VirtualBox:~/examples$
```

Nota: per compilare ed eseguire programmi Java devono essere installati JDK e JRE (già installati nella VM del laboratorio)



IDE

- Per scrivere il programma precedente è stato utilizzato un classico editor di testo (gedit).
- Solitamente per programmi più complessi si usano particolari strumenti più complessi chiamati Integrated Development Environment (IDE). Gli IDE permettono una gestione più completa del codice e delle varie classi, e segnalano in tempo reale eventuali errori di compilazione.
- Esempi di IDE Sono Eclipse e NetBeans (entrambi installati nella VM)



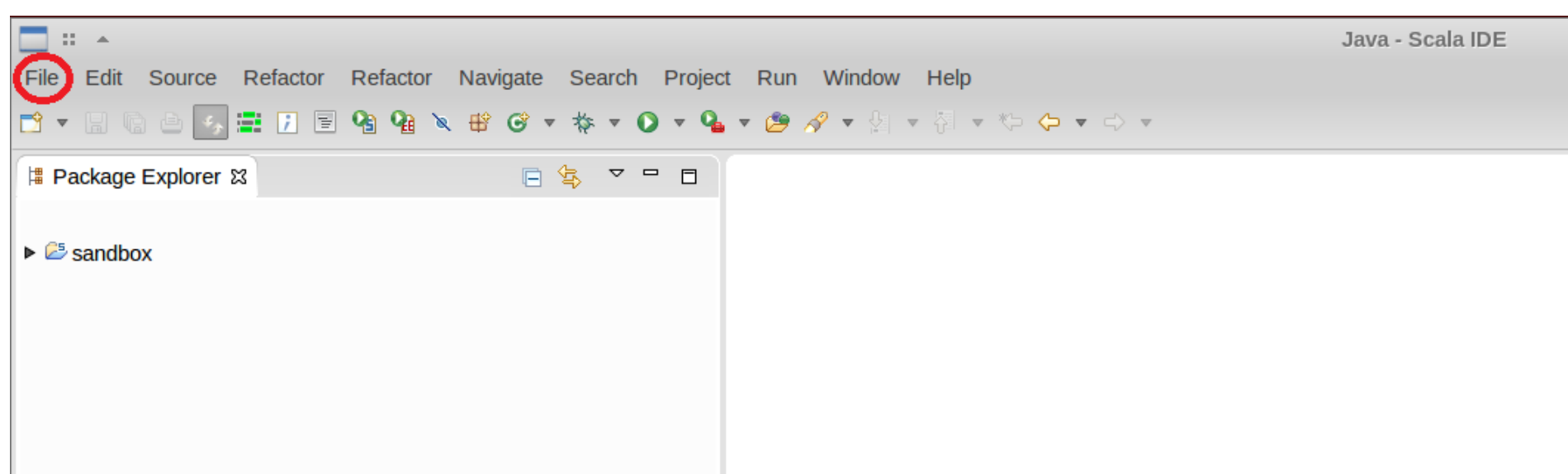
IDE

- In particolare ogni IDE deve avere più componenti:
- un editor per il codice sorgente;
- un compilatore e un interprete;
- un debugger.



IDE Esempio: Eclipse

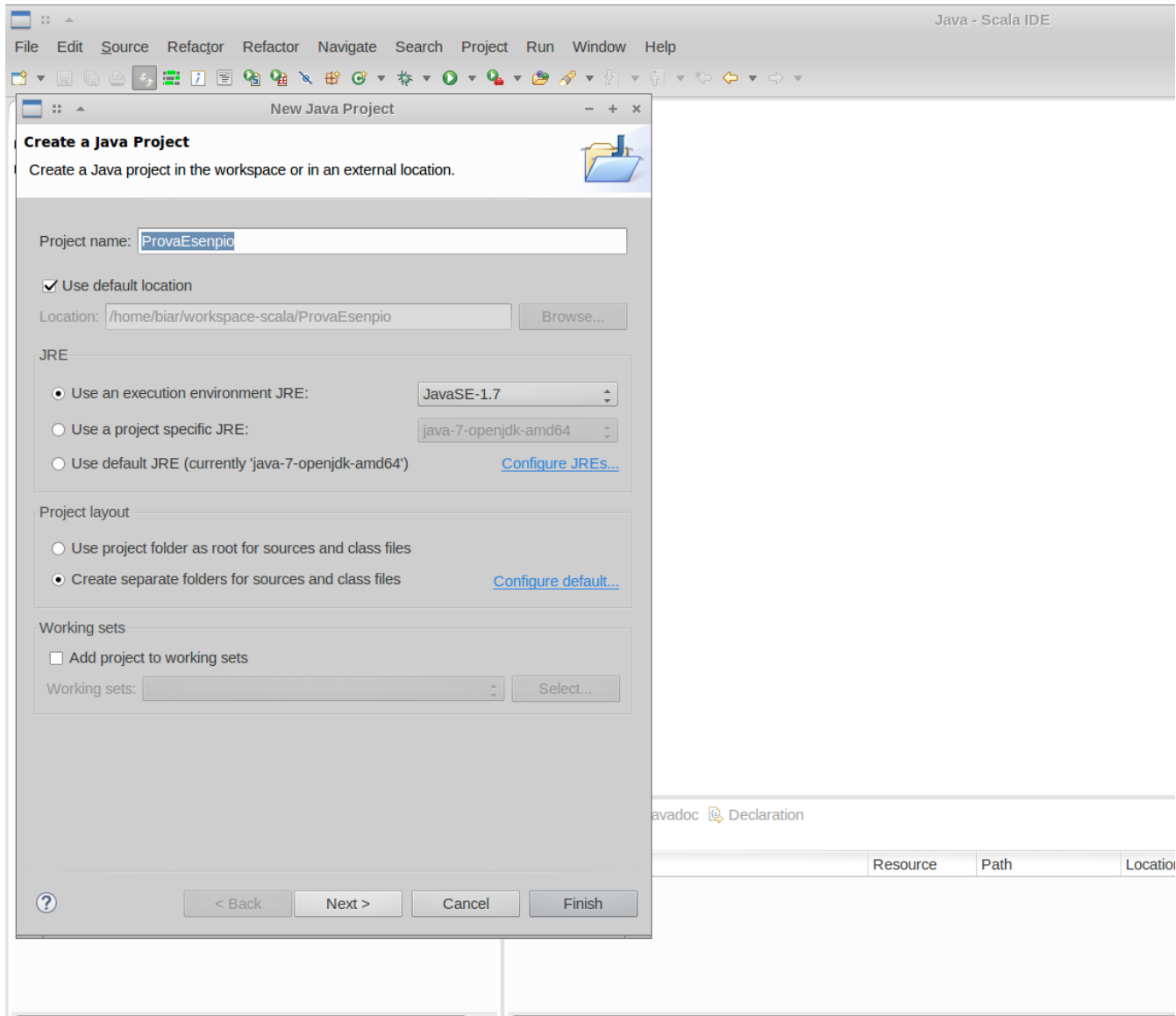
In Eclipse tutti gli elementi di un programma (codice sorgente, file .class, librerie di sistema ecc...) sono contenuti in un Progetto. Per creare un nuovo progetto vuoto:



File → New → Java Project

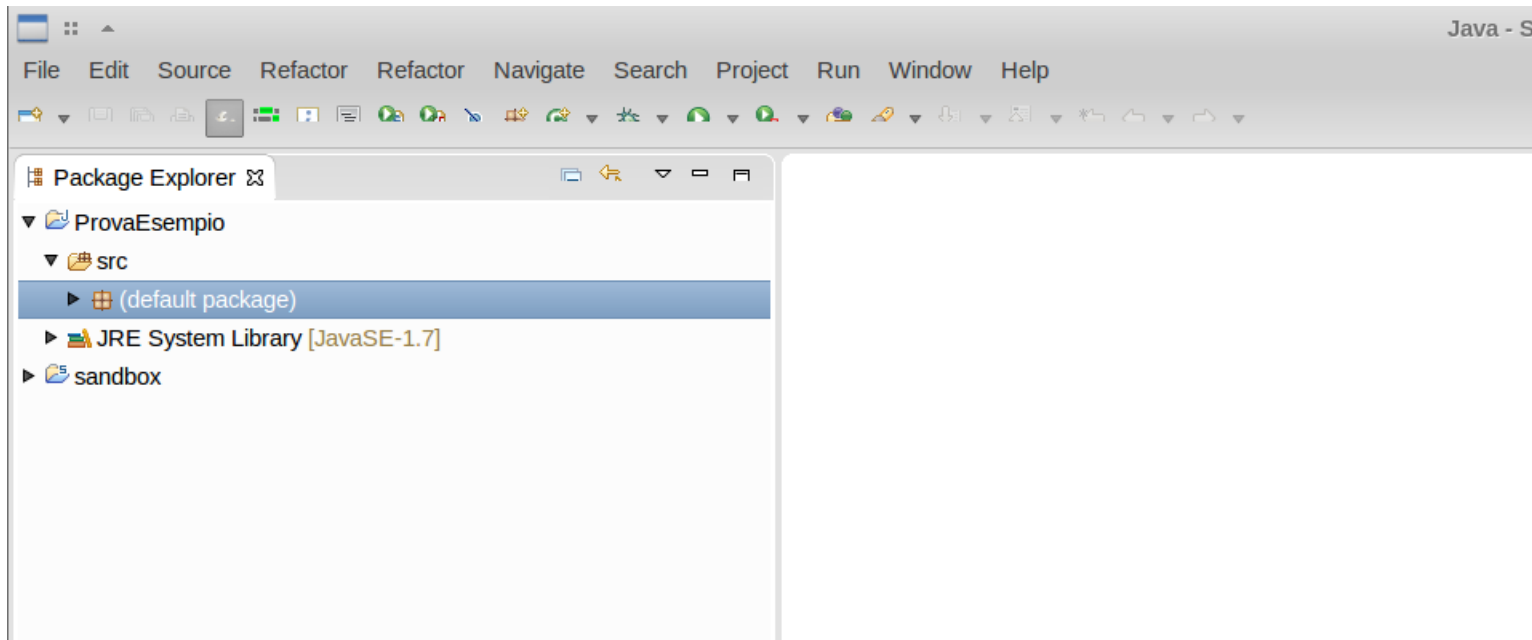


IDE Esempio: Eclipse



IDE Esempio: Eclipse

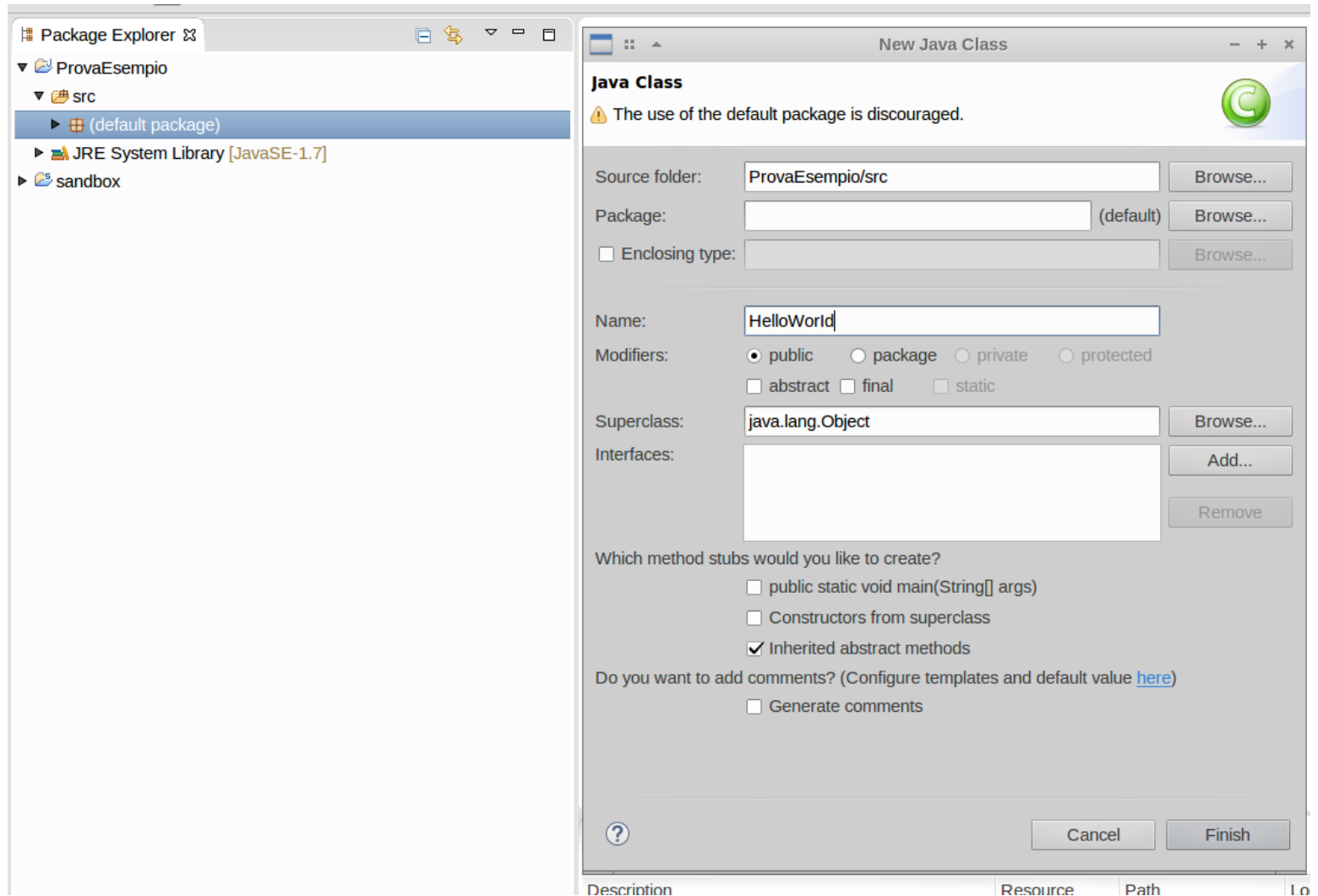
Una volta creato il progetto, possiamo creare classi e packages. Al momento, creiamo le classi nel package di default:



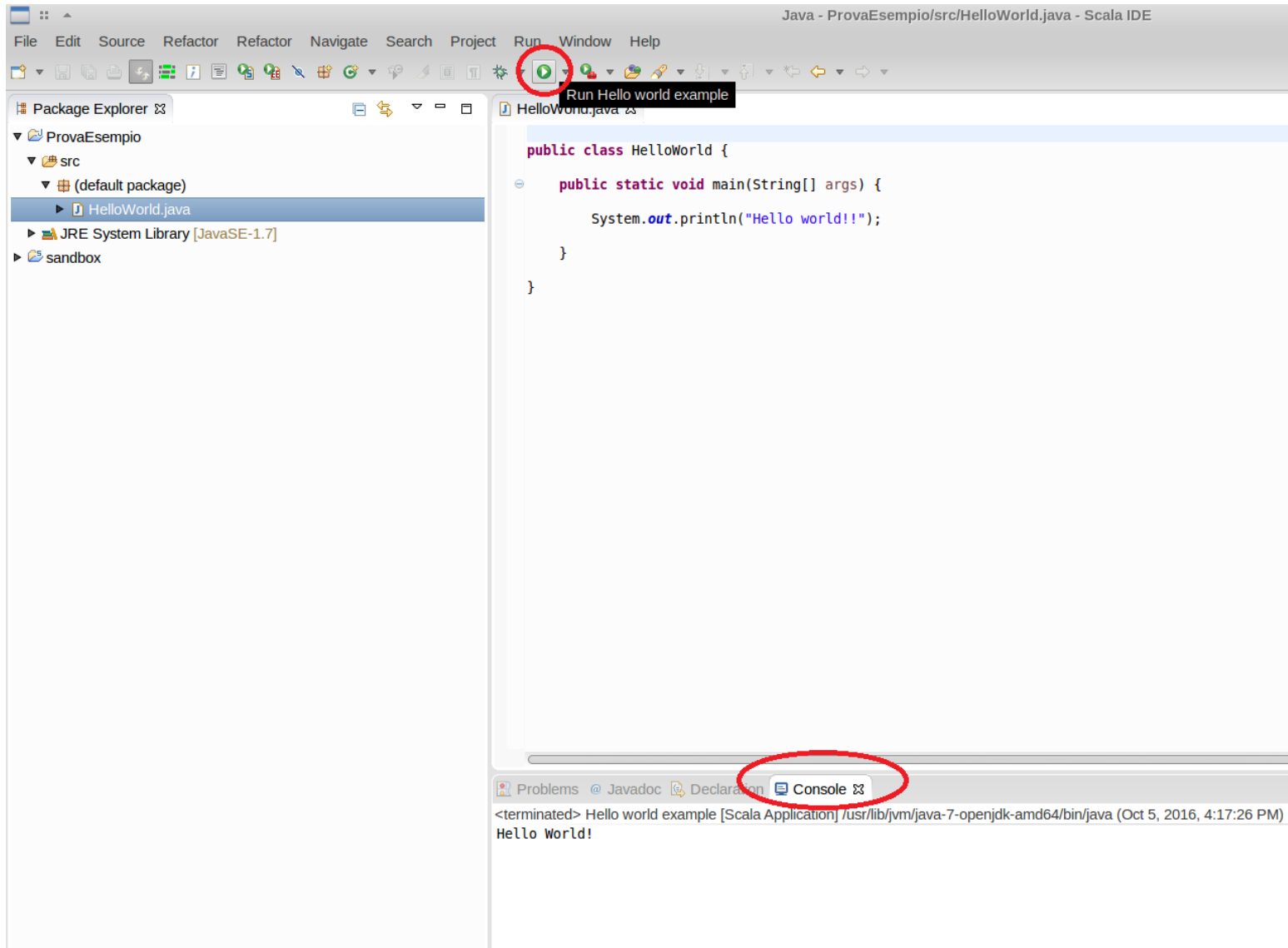
Da ProvaEsempio → src → (default package) col tasto destro selezionare New → Class



IDE Esempio: Eclipse

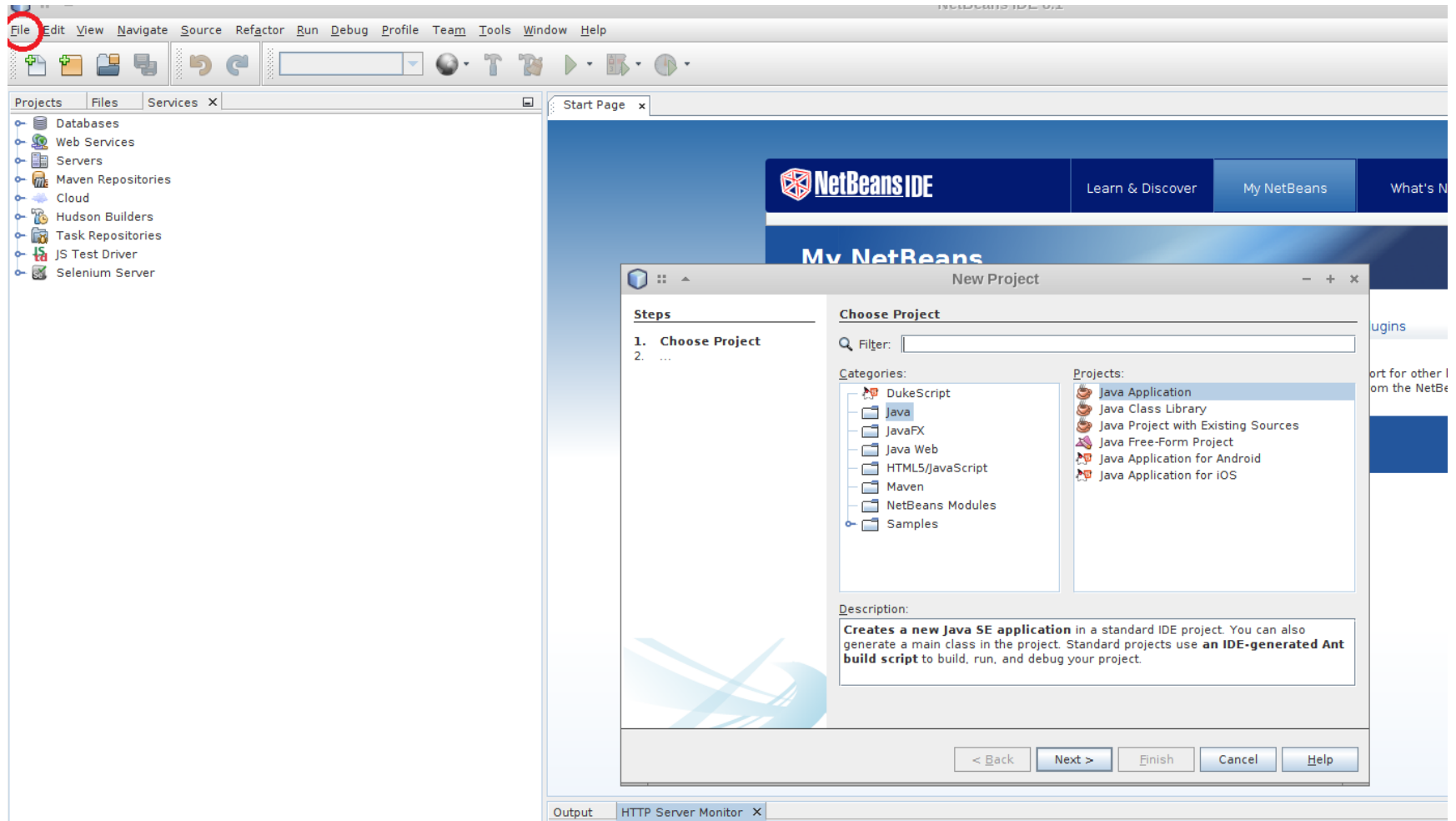


IDE Esempio: Eclipse

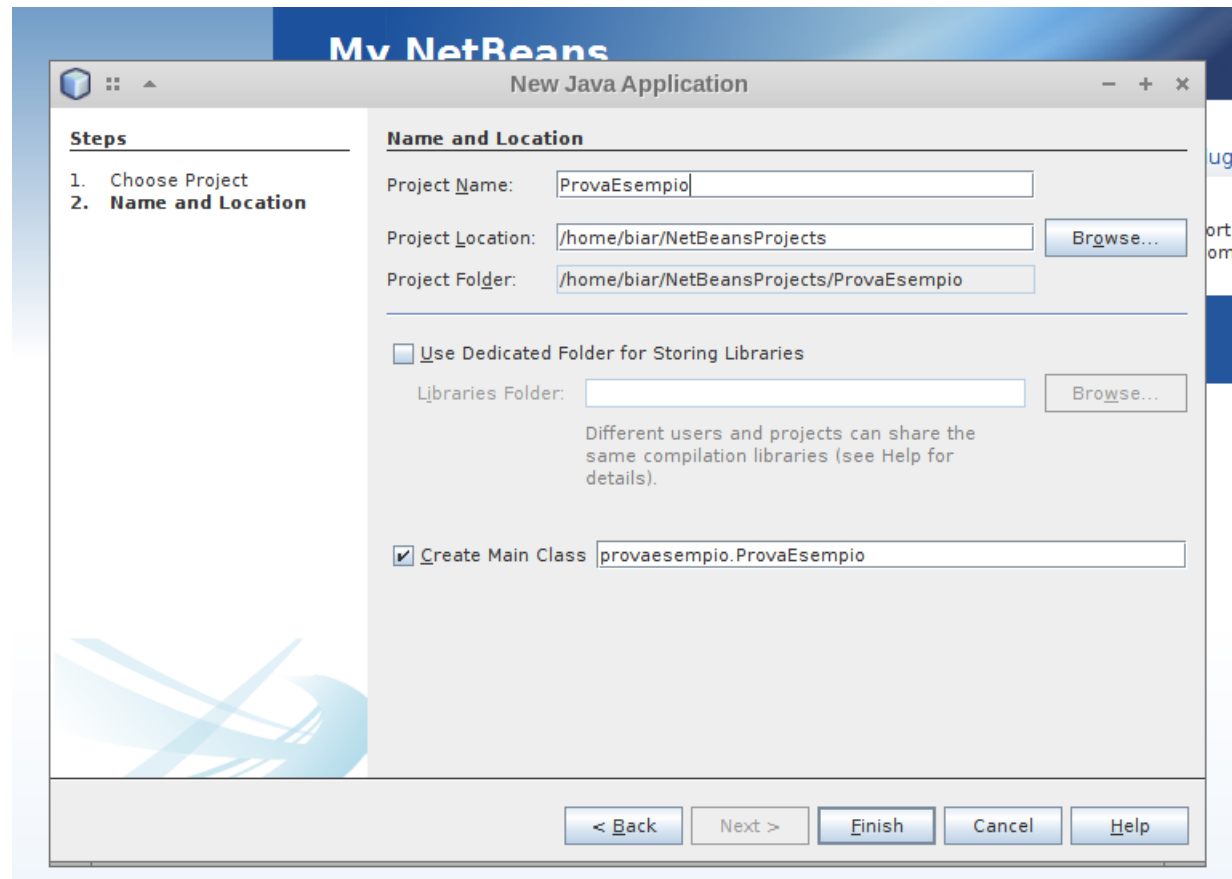


IDE Esempio: NetBeans

Ora riproponiamo lo stesso esempio utilizzando NetBeans:

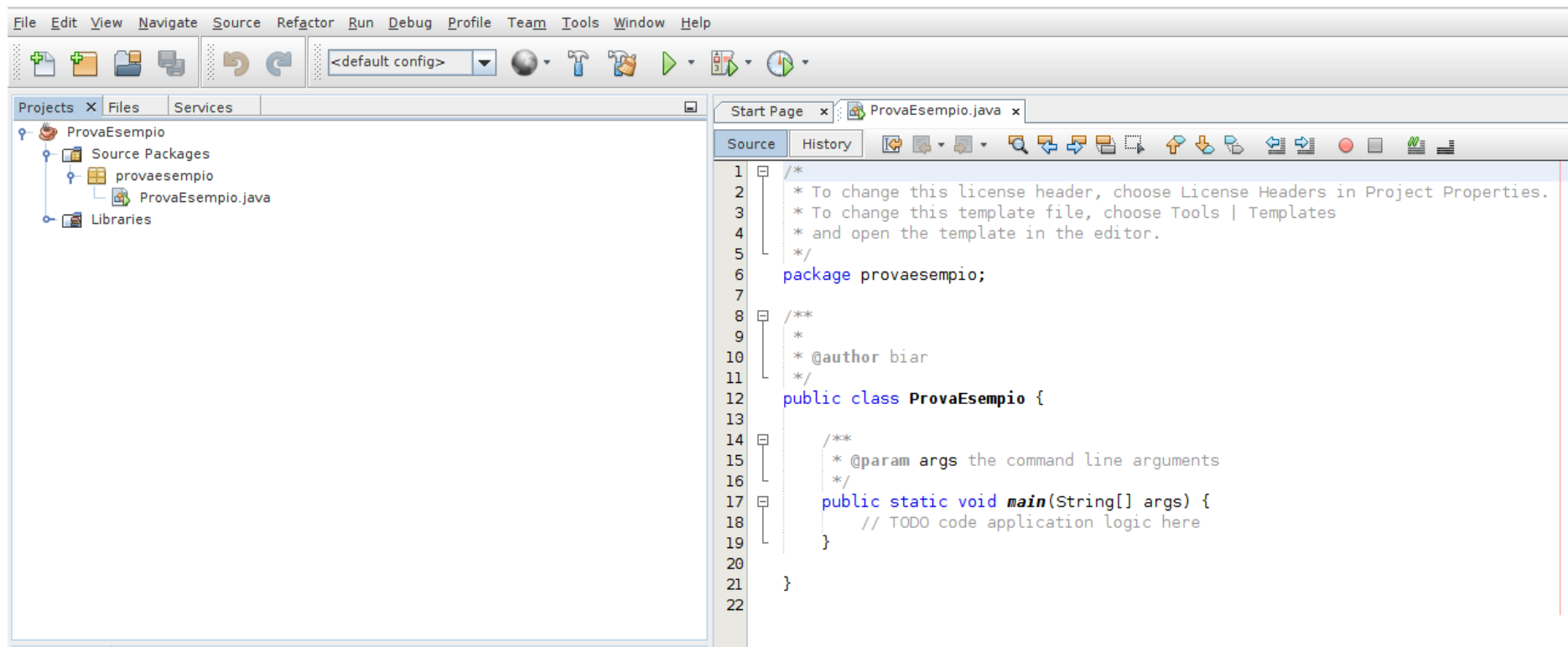


IDE Esempio: NetBeans

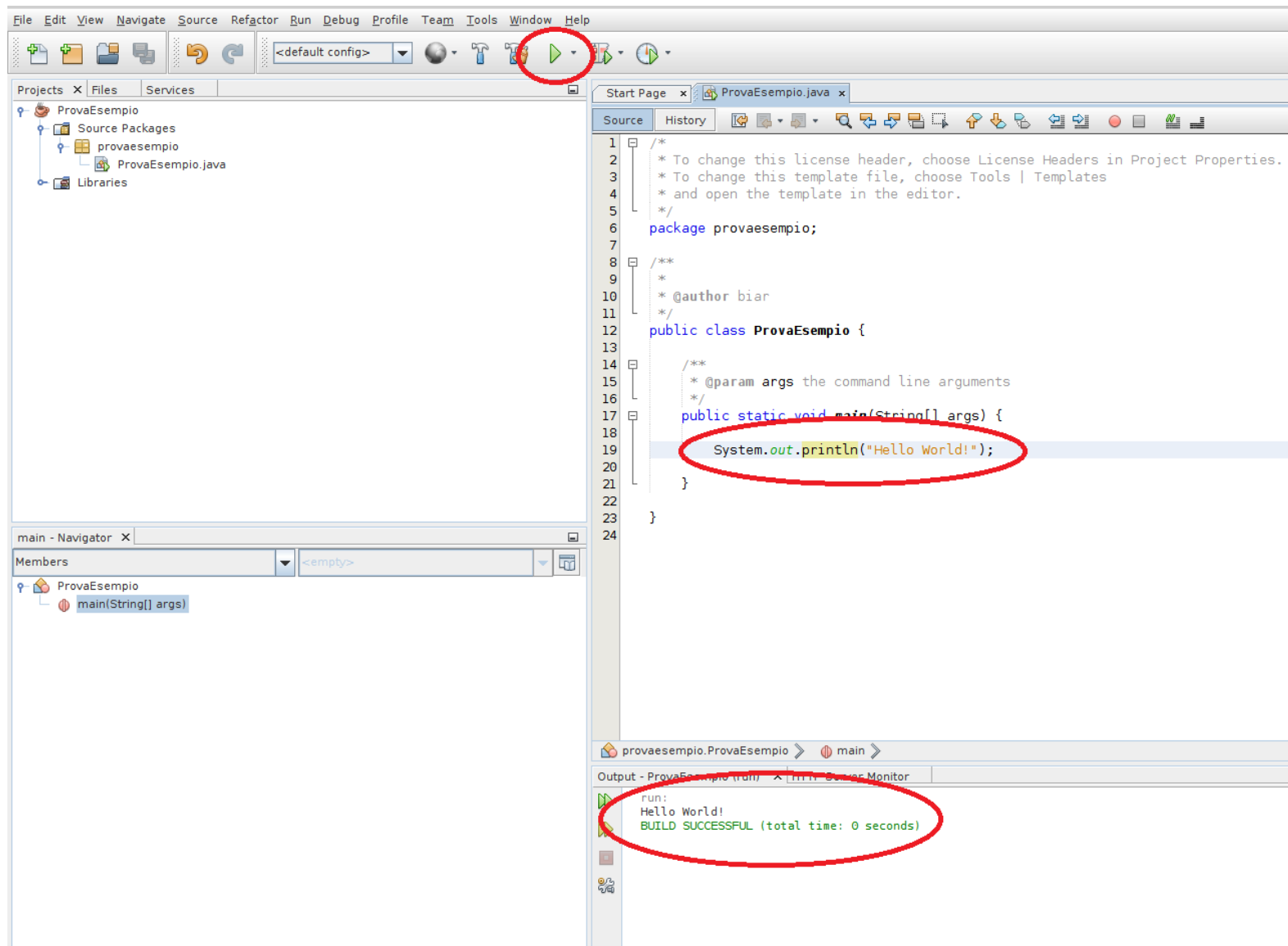


IDE Esempio: NetBeans

NetBeans a differenza di Eclipse crea direttamente la classe main in un package diverso da quello di default



IDE Esempio: NetBeans



API Java

- Con API (Application Programming Interface) si indicano tutta una serie di procedure che sono a disposizione del programmatore.
- In particolare le API Java mostrano tutte le classi offerte dal linguaggio e descrivono i metodi a disposizione in ogni classe.
- Le API per Java 7 sono disponibili all'URL <https://docs.oracle.com/javase/7/docs/api/>



API Java

- Con API (Application Programming Interface) si indicano tutta una serie di procedure che sono a disposizione del programmatore.
- In particolare le API Java mostrano tutte le classi offerte dal linguaggio e descrivono i metodi a disposizione in ogni classe.
- Le API per Java 7 sono disponibili all'URL <https://docs.oracle.com/javase/7/docs/api/>



API Java

- Per ogni Classe le API riportano e descrivono in dettaglio superclassi, sottoclassi, funzionalità ad alto livello della classe, campi, costruttori, metodi.
- In questa prima esercitazione ci concentreremo soprattutto sui metodi



API Java

https://docs.oracle.com/javase/7/docs/api/ api java

Java™ Platform Standard Ed. 7

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing <i>beans</i> -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.



API Java

Esempio: la classe String

The screenshot shows the Oracle Java API documentation for the `String` class. The browser address bar shows `https://docs.oracle.com/javase/7/docs/api/` and the search bar contains `api java`. The left sidebar lists various Java packages, with `String` highlighted under the `java.lang` package. The main content area displays the `String` class details, including its inheritance hierarchy, implemented interfaces, and source code.

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class String

java.lang.Object
java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2,3);
String d = cde.substring(1, 2);
```

The class `String` includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the `Character` class.

The Java language provides special support for the string concatenation operator (`+`), and for conversion of other objects to strings. String concatenation is implemented through the `StringBuilder` (or `StringBuffer`) class and its `append` method. String conversions are implemented through the method `toString`, defined by `Object` and inherited by all classes in Java. For additional information on string concatenation and conversion, see Gosling, Joy, and Steele, *The Java Language Specification*.



API Java

Esempio: la classe String

https://docs.oracle.com/javase/7/docs/api/

java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
StatementEvent
StatementEventListener
StAXResult
StAXSource
Streamable
StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder
StringCharacterIterator
StringContent
StringHolder
StringIndexOutOfBoundsException
StringMonitor
StringMonitorMBean
StringNameHelper
StringReader
StringRefAddr
StringSelection
StringSeqHelper
StringSeqHolder
StringTokenizer
StringValueExp

Field Summary

Fields	
Modifier and Type	Field and Description
static Comparator<String>	CASE_INSENSITIVE_ORDER A Comparator that orders String objects as by compareToIgnoreCase.

Constructor Summary

Constructors	
Constructor and Description	
String()	Initializes a newly created String object so that it represents an empty character sequence.
String(byte[] bytes)	Constructs a new String by decoding the specified array of bytes using the platform's default charset.
String(byte[] bytes, Charset charset)	Constructs a new String by decoding the specified array of bytes using the specified charset.
String(byte[] ascii, int hibyte)	Deprecated. This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a Charset, charset name, or that use the platform's default charset.
String(byte[] bytes, int offset, int length)	Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.
String(byte[] bytes, int offset, int length, Charset charset)	Constructs a new String by decoding the specified subarray of bytes using the specified charset.
String(byte[] ascii, int hibyte, int offset, int count)	Deprecated. This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a Charset, charset name, or that use the platform's default charset.
String(byte[] bytes, int offset, int length, String charsetName)	Constructs a new String by decoding the specified subarray of bytes using the specified charset.
String(byte[] bytes, String charsetName)	Constructs a new String by decoding the specified array of bytes using the specified charset.
String(char[] value)	Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.
String(char[] value, int offset, int count)	



API Java

Esempio: la classe String

← ⓘ 🔒 https://docs.oracle.com/javase/7/docs/api/ 🔍 api java ☆ 📁 📄 ⬇ 🏠 🚫

java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
StatementEvent
StatementEventListener
StAXResult
StAXSource
Streamable
StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder
StringCharacterIterator
StringContent
StringHolder
StringIndexOutOfBoundsException
StringMonitor
StringMonitorMBean
StringNameHelper
StringReader
StringRefAddr
StringSelection
StringSeqHelper
StringSeqHolder
StringTokenizer
StringValueExp

Method Summary

Methods	
Modifier and Type	Method and Description
char	<code>charAt(int index)</code> Returns the char value at the specified index.
int	<code>codePointAt(int index)</code> Returns the character (Unicode code point) at the specified index.
int	<code>codePointBefore(int index)</code> Returns the character (Unicode code point) before the specified index.
int	<code>codePointCount(int beginIndex, int endIndex)</code> Returns the number of Unicode code points in the specified text range of this String.
int	<code>compareTo(String anotherString)</code> Compares two strings lexicographically.
int	<code>compareToIgnoreCase(String str)</code> Compares two strings lexicographically, ignoring case differences.
String	<code>concat(String str)</code> Concatenates the specified string to the end of this string.
boolean	<code>contains(CharSequence s)</code> Returns true if and only if this string contains the specified sequence of char values.
boolean	<code>contentEquals(CharSequence cs)</code> Compares this string to the specified CharSequence.
boolean	<code>contentEquals(StringBuffer sb)</code> Compares this string to the specified StringBuffer.
static String	<code>copyValueOf(char[] data)</code> Returns a String that represents the character sequence in the array specified.
static String	<code>copyValueOf(char[] data, int offset, int count)</code> Returns a String that represents the character sequence in the array specified.
boolean	<code>endsWith(String suffix)</code> Tests if this string ends with the specified suffix.
boolean	<code>equals(Object anObject)</code> Compares this string to the specified object.
boolean	<code>equalsIgnoreCase(String anotherString)</code> Compares this String to another String, ignoring case considerations.
static String	<code>format(Locale l, String format, Object... args)</code> Returns a formatted string using the specified locale, format string, and arguments.
static String	<code>format(String format, Object... args)</code> Returns a formatted string using the specified format string and arguments.

