

# Linguaggi di Programmazione

## Introduzione a Java

Creare, compilare, eseguire e distribuire un programma  
scritto in Java

# Outline

- Applicazione di esempio: Hello World!
- Ciclo di sviluppo
- Netbeans
  - Compilare
  - Eseguire
  - Distribuire
- Il prompt dei comandi
  - Compilare
  - Eseguire
  - Distribuire

# HelloWorldApp

```
package it.unitn.prog2;

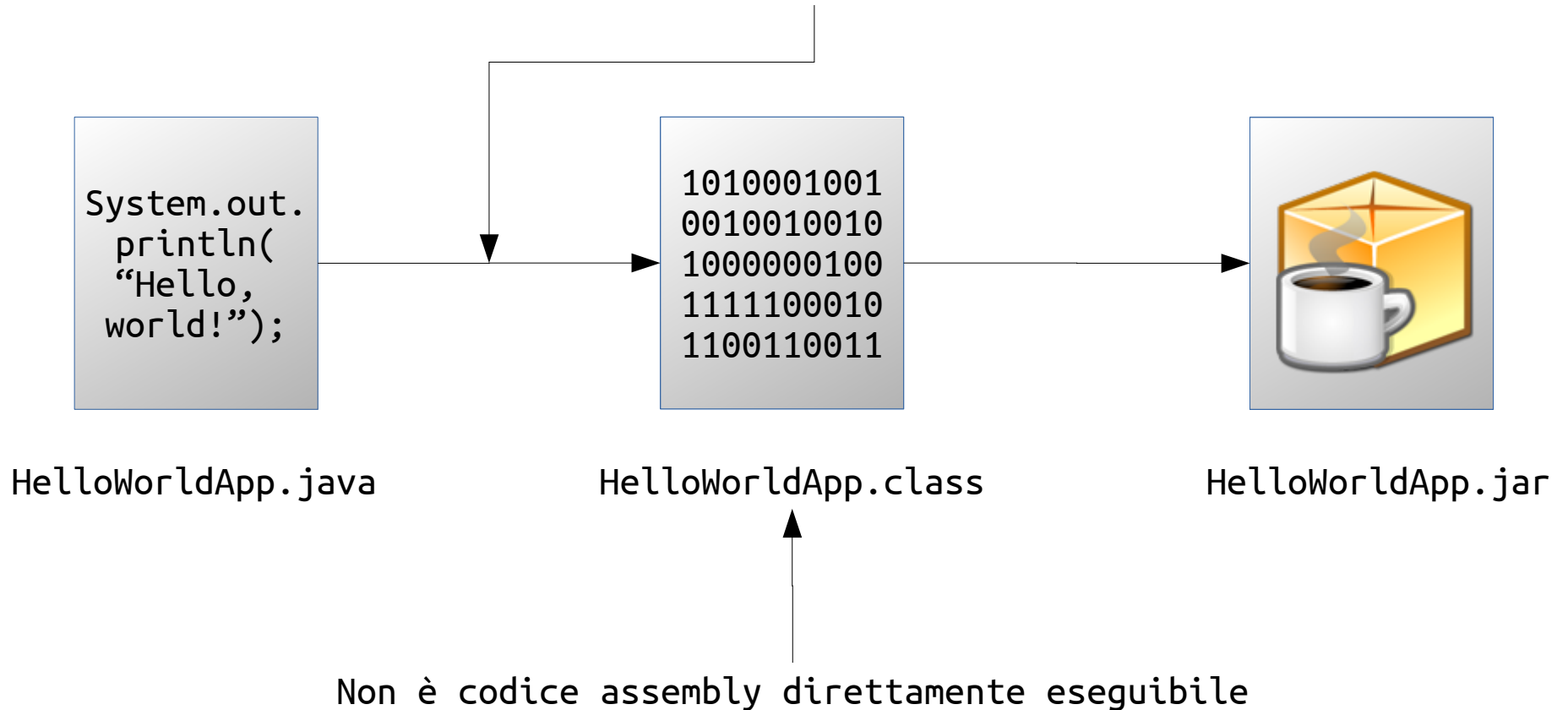
public class HelloWorldApp {

    public HelloWorldApp() {
        System.out.println("Hello, world!");
    }

    public static void main(String[] args) {
        HelloWorldApp app = new HelloWorldApp();
    }
}
```

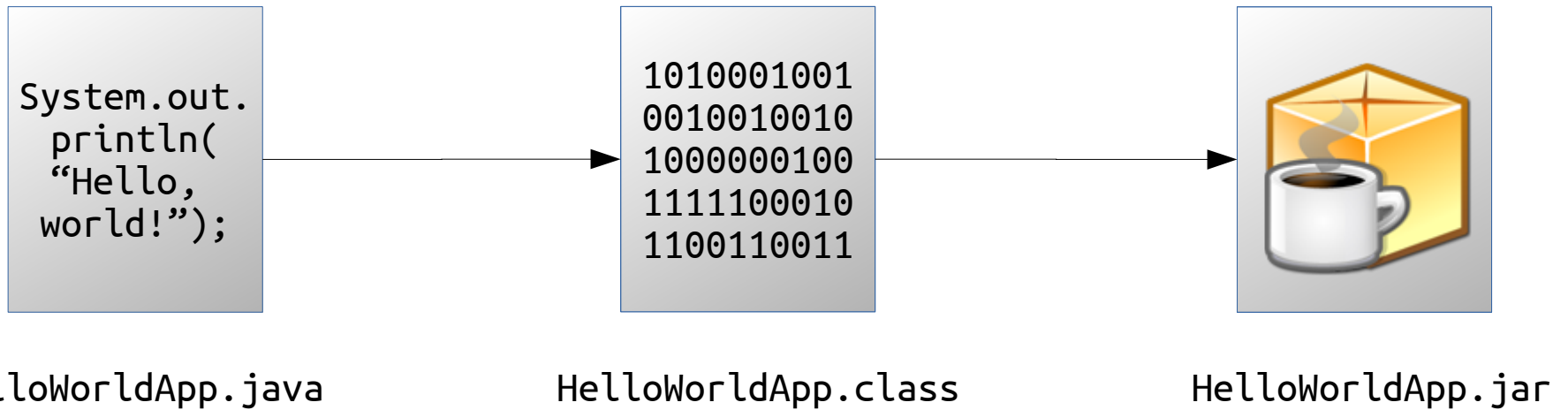
# Ciclo di sviluppo: compilare

Il programma (uno o più file di testo) viene convertito in binario (bytecode)



# Ciclo di sviluppo: eseguire

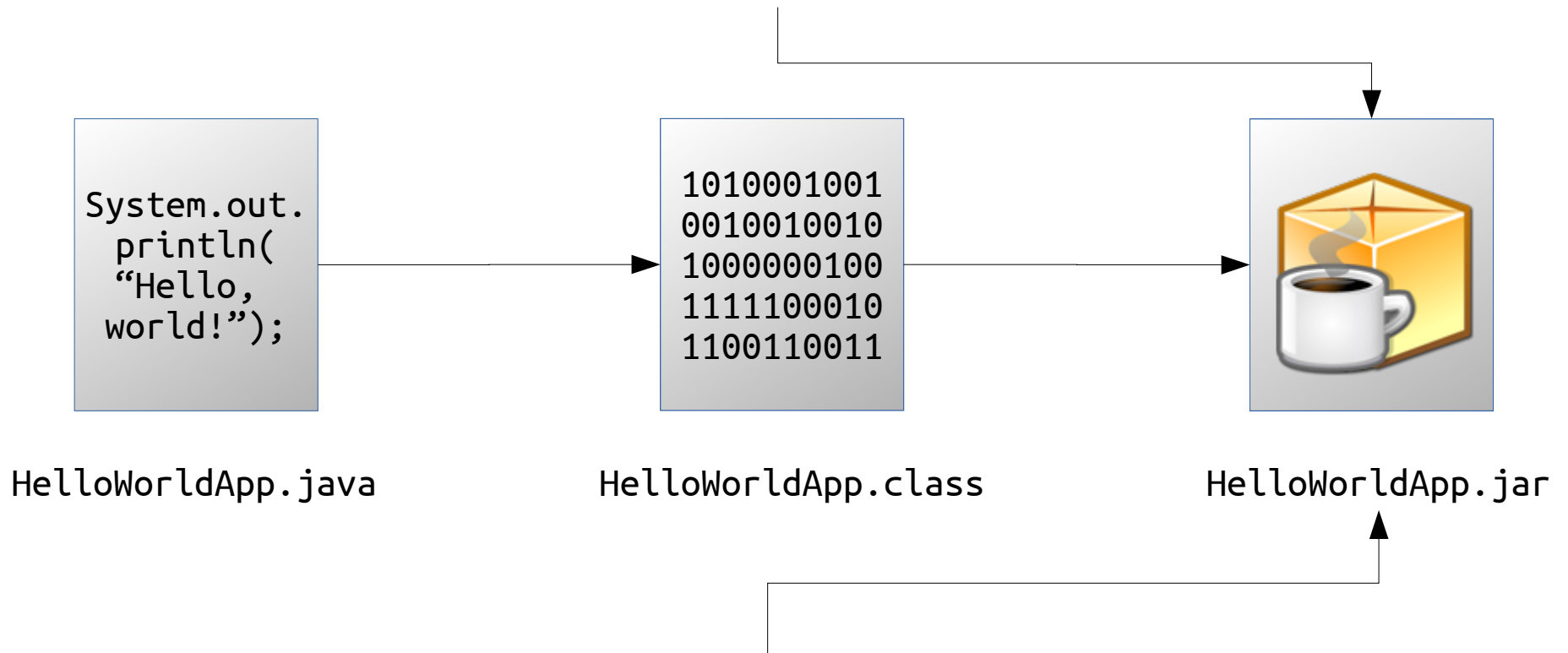
La JVM è disponibile per moltissimi sistemi operativi / architetture



Il codice binario può essere eseguito nella JVM, uno speciale programma interprete

# Ciclo di sviluppo: distribuire

I file dell'applicazione vengono raccolti in un archivio eseguibile

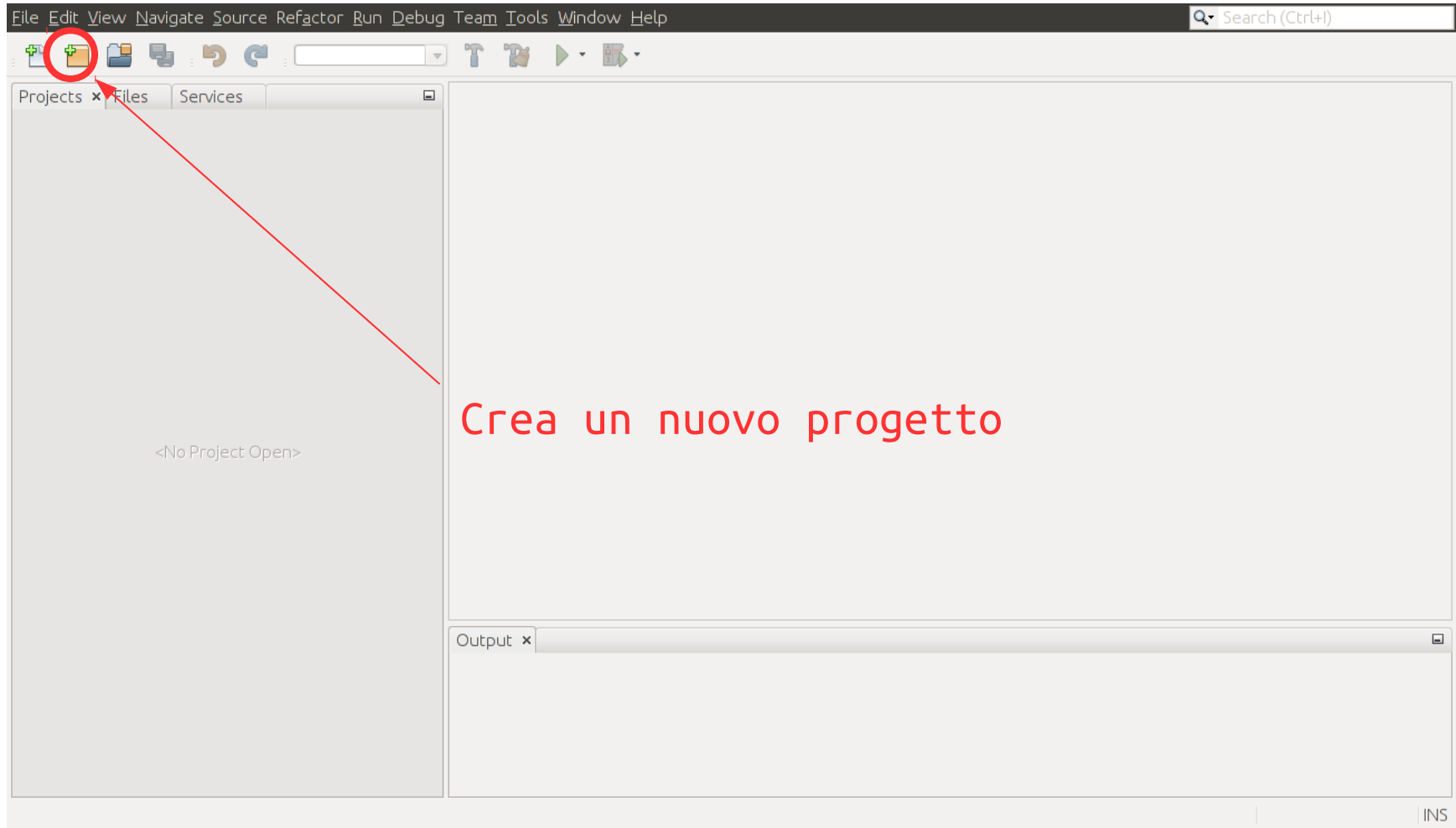


Non è altro che un archivio ZIP con una struttura interna speciale

# Netbeans

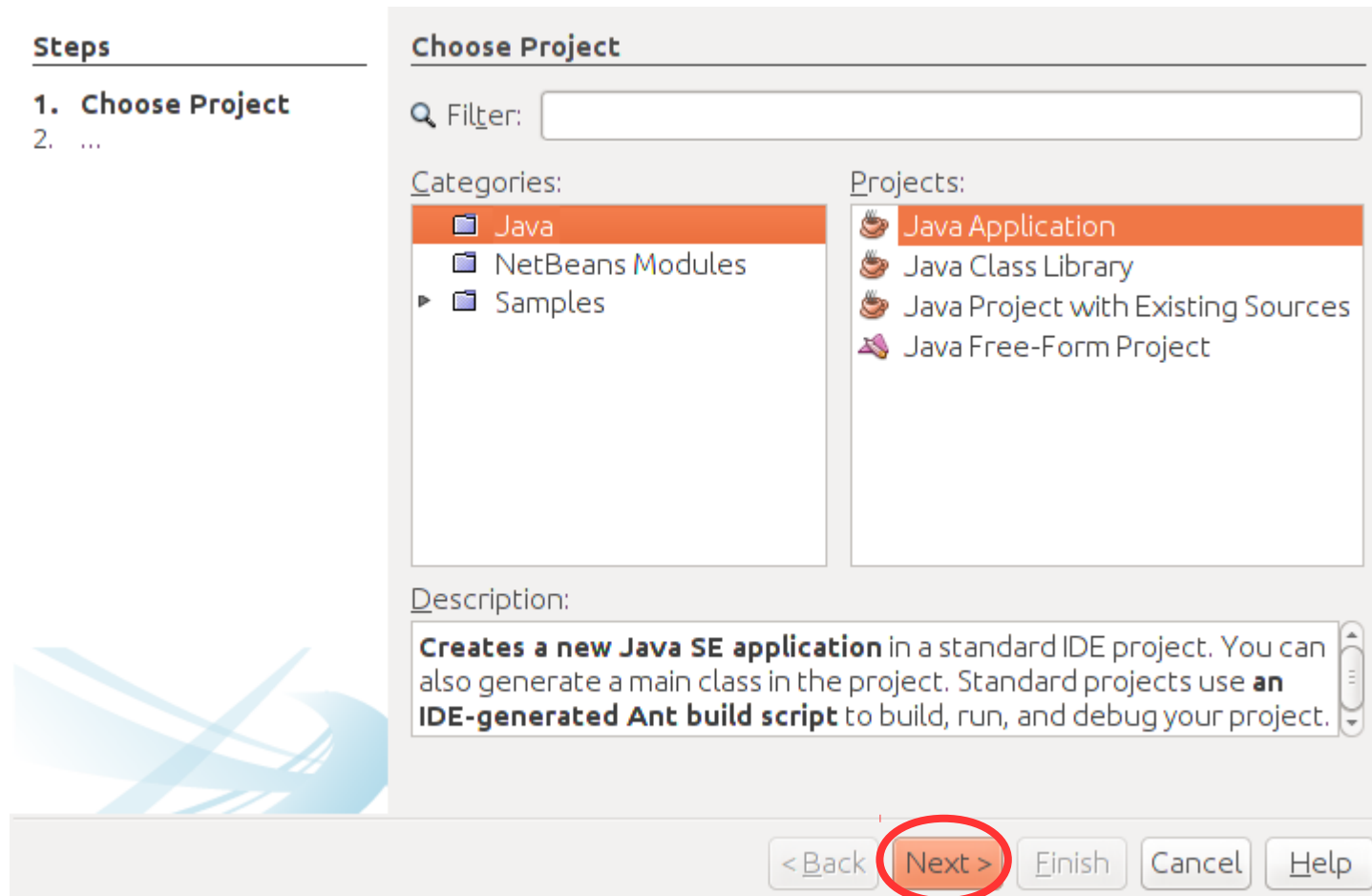
- Un IDE Java, scritto in Java
- Opensource
- Non solo codice Java
- Era Sun/Oracle, a breve sarà Apache
- <https://netbeans.org/>

# Netbeans: creare un progetto





# Netbeans: creare un progetto



# Netbeans: creare un progetto

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

Use Dedicated Folder for Storing Libraries

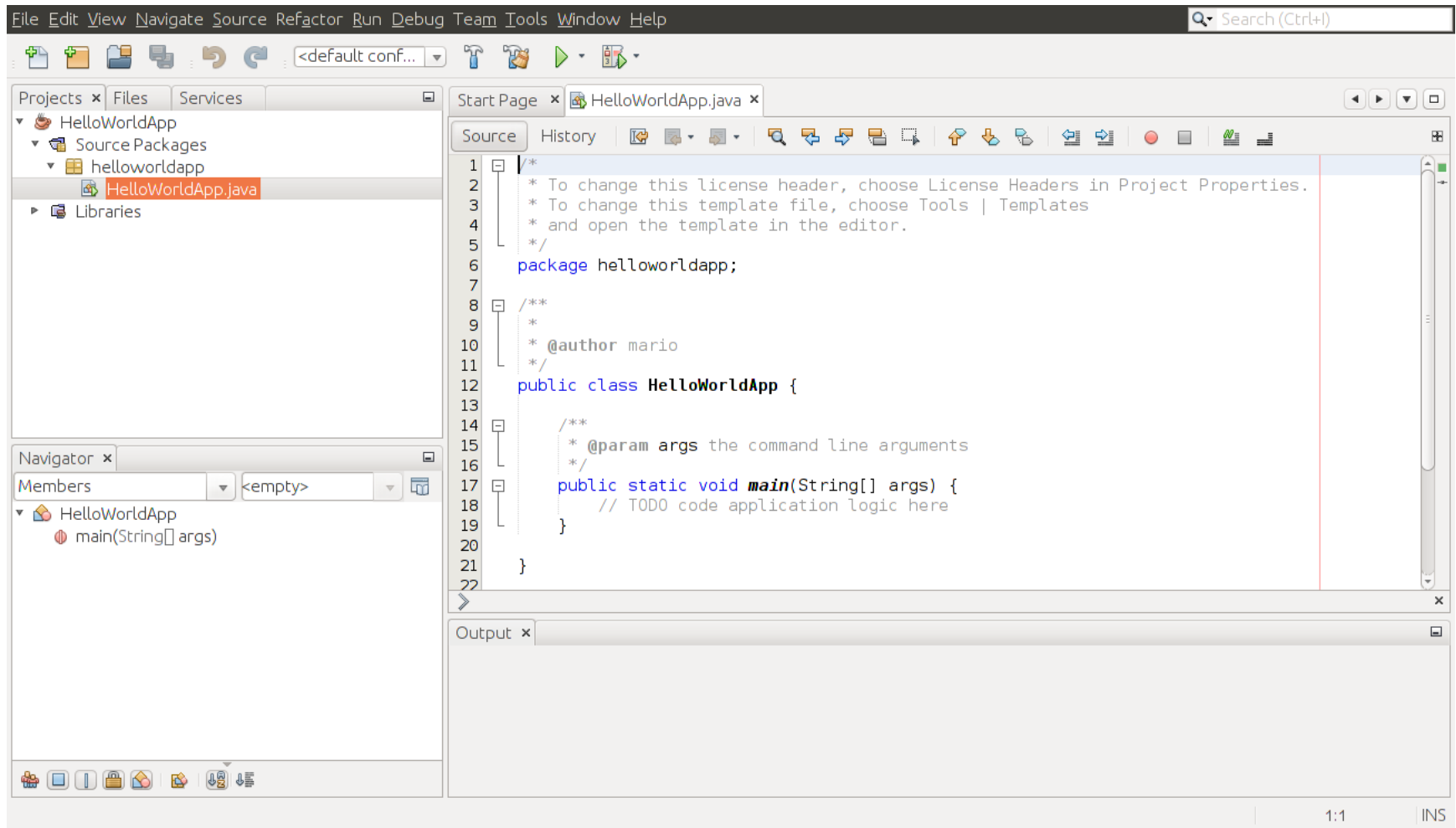
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

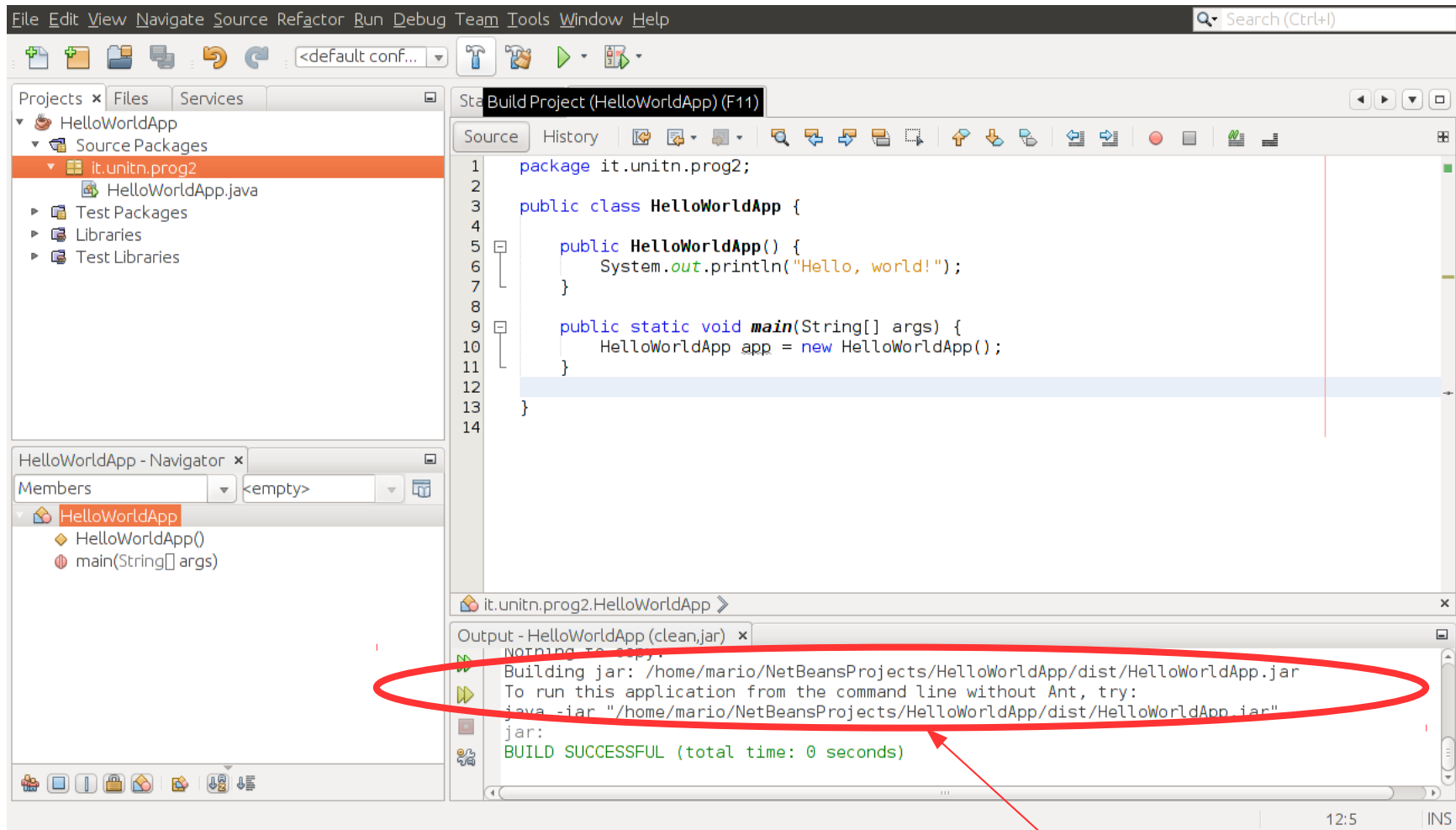
Create Main Class

< Back   Next >   **Finish**   Cancel   Help

# Netbeans: creare un progetto

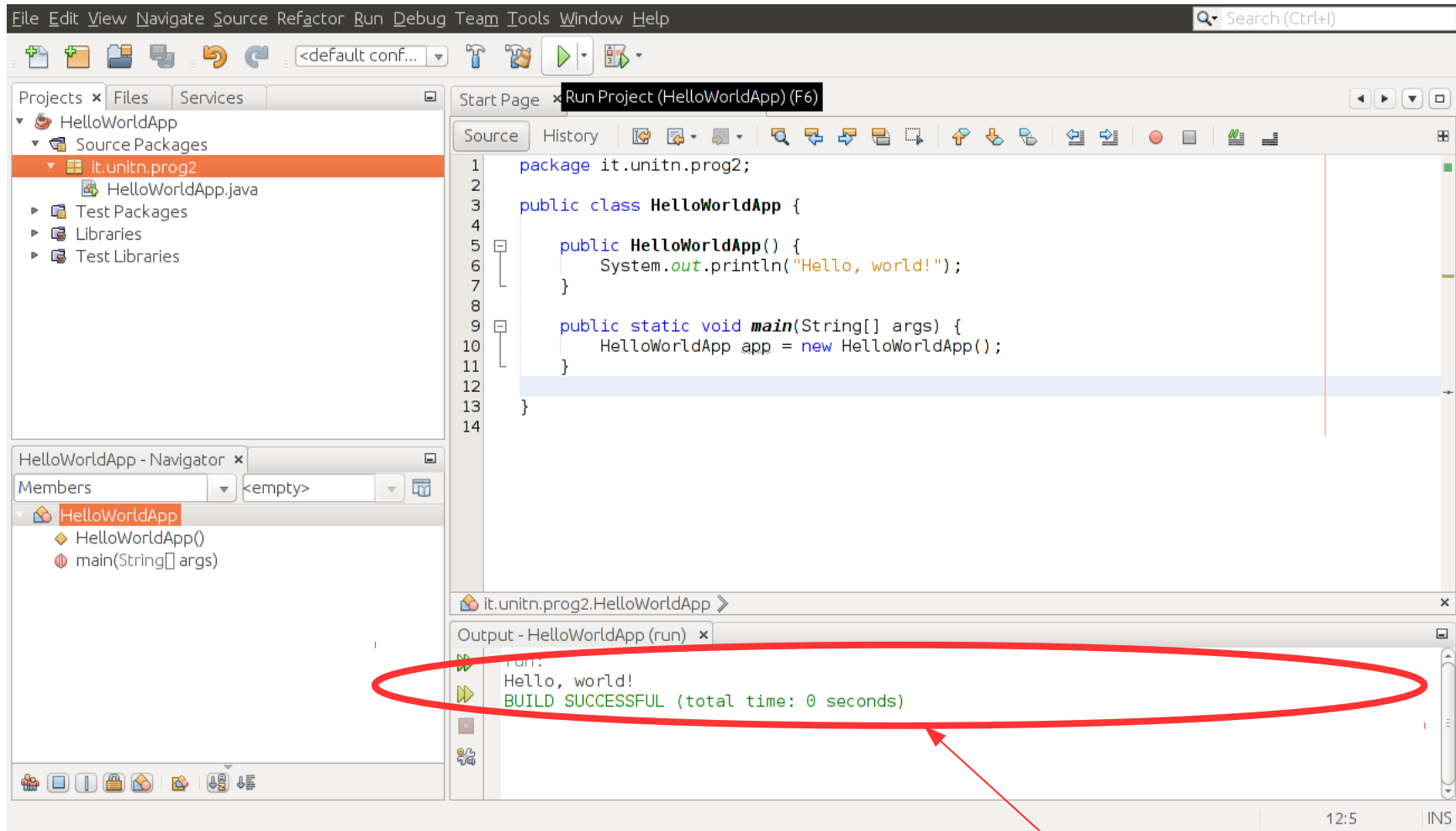


# Netbeans: compilare (build)



Questo passaggio crea anche un file .jar

# Netbeans: eseguire



Risultati esecuzione e output della console del programma

# Il prompt dei comandi: preparare l'ambiente di sviluppo

- L'ambiente di sviluppo (Java SDK) deve essere installato correttamente
  - I comandi java, javac e jar devono essere eseguibili
  - Devono essere nel path degli eseguibili
- Verifica (in Linux):
  - java -version
  - whereis java javac jar
  - echo \$PATH

# Il prompt dei comandi: preparare l'ambiente di sviluppo

- Verifica (in Windows):

```
java -version  
echo %PATH%
```

- Variabili di ambiente:

```
https://stackoverflow.com/questions/1672281/environment-variables-for-java-installation#26640589
```

- Verifica (in MacOS):

- ~ Linux

# Il prompt dei comandi

- Spostarsi nella cartella:

```
cd ~/NetbeansProject/HelloWorldApp
```

- Listarne il contenuto:

```
ls
```

- Output:

```
build build.xml dist manifest.mf nbproject src test
```

↑  
compilati

↑  
eseguibili

↑  
sorgenti



# Package e struttura cartelle

- In Java, package = struttura cartelle

```
ls -R src
```

- Output:

```
src:
```

```
it
```

```
src/it:
```

```
unitn
```

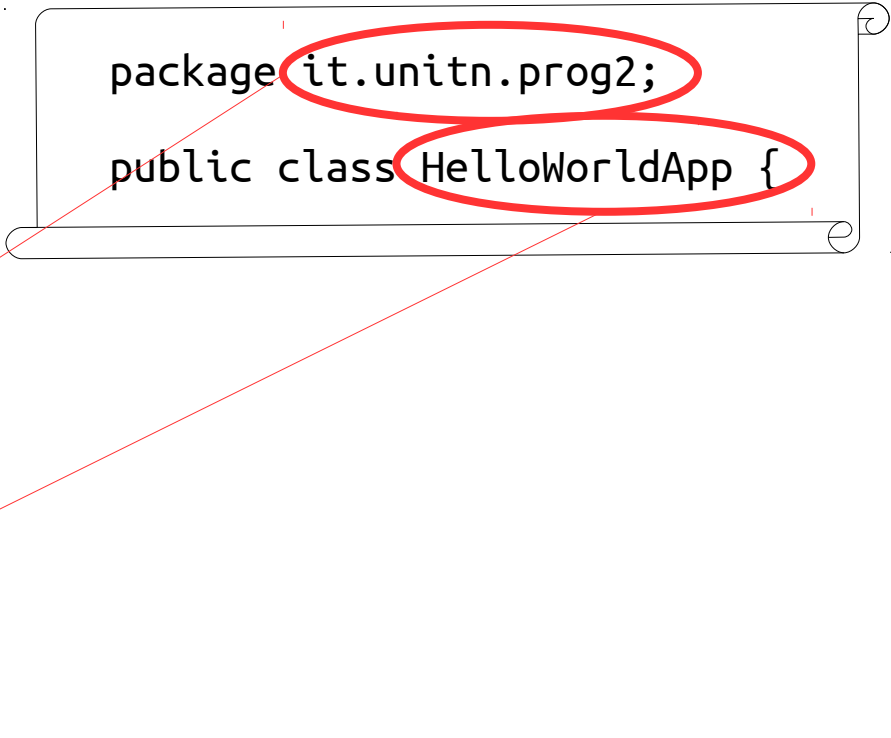
```
src/it/unitn:
```

```
prog2
```

```
src/it/unitn/prog2:
```

```
HelloWorldApp.java
```

```
package it.unitn.prog2;  
public class HelloWorldApp {
```



# Il prompt dei comandi: compilare

- Si utilizza il tool javac:

```
javac nomefile.java
```

- Nel nostro caso:

```
javac -sourcepath src/  
      -d build/classes/  
      src/it/unitn/prog2/HelloWorldApp.java
```

- Per sapere cosa succede effettivamente (utile in caso di errori):

```
javac -verbose  
      -sourcepath src/  
      -d build/classes/  
      src/it/unitn/prog2/HelloWorldApp.java
```

# Il prompt dei comandi: eseguire

- Si utilizza l'interprete java:

```
java package.nomeClasse
```

- Nel nostro caso:

```
java -classpath build/classes  
    it.unitn.prog2.HelloWorldApp
```

- Per sapere cosa succede effettivamente (utile in caso di errori):

```
java -verbose  
    -classpath build/classes  
    it.unitn.prog2.HelloWorldApp
```

# Il prompt dei comandi: distribuire

- Si utilizza il tool jar:

```
jar cfe nomefile.jar package.ClasseConMain nomefile.class
```

- Nel nostro caso:

```
jar cfe HelloWorldApp.jar  
    it.unitn.prog2.HelloWorldApp  
    -C build/classes/  
    it/unitn/prog2/HelloWorldApp.class
```

- Analizziamo il file (è semplicemente un archivio zippato):

```
unzip -l HelloWorldApp.jar
```

# Il prompt dei comandi: distribuire

- Contenuto file jar:

`META-INF`

`META-INF/MANIFEST.MF`

`it/unitn/prog2/HelloWorldApp.class`

- Il file “MANIFEST.MF” spiega quale classe con la main eseguire:

`unzip -ap HelloWorldApp.jar META-INF/MANIFEST.MF`

- Contiene la riga:

`Main-Class: it.unitn.prog2.HelloWorldApp`

# Eseguire il file Jar

- Semplicemente:

```
java -jar HelloWorldApp.jar
```

- Netbeans mette I suoi jar in un'altra directory:

```
ls -l dist
```

- E' del tutto simile al file creato al prompt dei comandi:

```
cd dist
```

```
unzip -l HelloWorldApp.jar
```

```
unzip -ap HelloWorldApp.jar META-INF/MANIFEST.MF
```

```
java -jar HelloWorldApp.jar
```

# Credits where credits are due

- Jar icon by David Vignoni / ICON KING -  
<http://icon-king.com>, LGPL,  
<https://commons.wikimedia.org/w/index.php?curid=326931>