

Introduzione a JAVA

Metodi Avanzati di Programmazione
Laurea Triennale in Informatica
Università degli Studi di Bari Aldo Moro
Docente: Pierpaolo Basile

Roadmap

- Introduzione alla programmazione ad oggetti (RECAP)
- Introduzione al linguaggio JAVA
- Gestione degli errori e collezioni di dati
- Generics
- Il sistema di I/O
- La programmazione concorrente
- Programmazione in rete
- Accesso a DBMS (JDBC)
- Sviluppo di interfacce grafiche con SWING
- Programmazione funzionale

JAVA un po' di storia...

- Creato a partire da ricerche effettuate alla Stanford University
- Nel 1992 Sun Microsystems pubblica la prima versione che si chiama Oak (quercia)
 - cambia il nome in JAVA per motivi di copyright
- Sintassi molto simile a C e C++ per facilitare la migrazione dei «vecchi» sviluppatori
- La storia di JAVA è legata allo sviluppo di Internet...

JAVA un po' di storia...

- Netscape integra nel suo browser la JVM (Java Virtual Machine)
 - per eseguire le applet
- Nel 1996 la Sun annuncia ufficialmente JAVA
- Nel 2006 viene pubblicato sotto licenza GPL
 - la sua versione completamente open-source è la IcedTea
 - nel 2007 anche le librerie diventano GPL
- Nel gennaio 2010 Sun viene acquisita dalla Oracle Corporation

JAVA caratteristiche principali

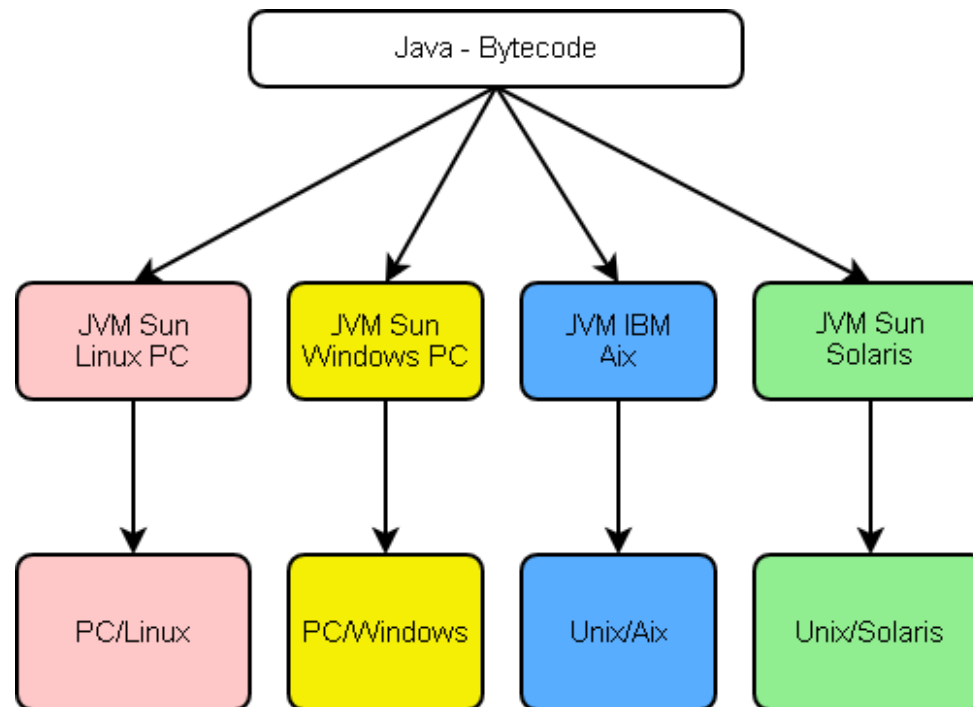
- JAVA è
 - un linguaggio **orientato agli oggetti** (object-oriented)
 - indipendente dalla piattaforma di esecuzione (**multiplatforma**)

Orientato agli oggetti

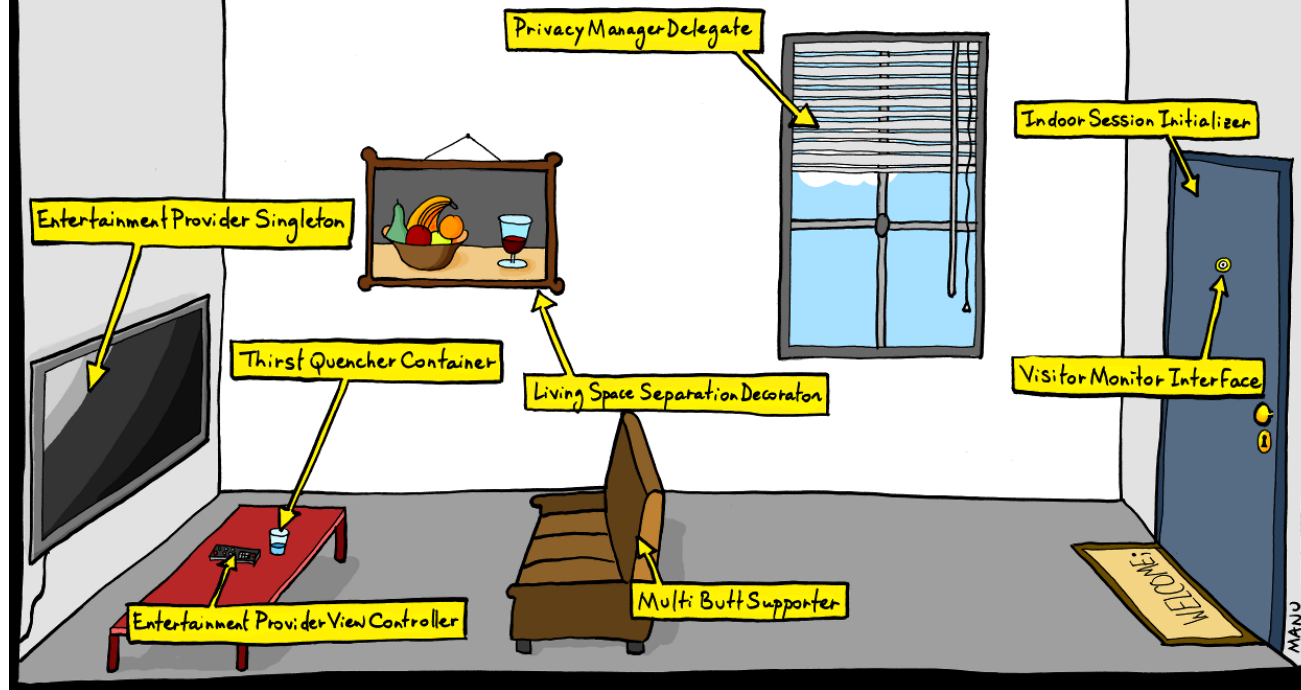
- Il mondo procedurale
 - un sistema di processi
 - descritto tramite un flow-chart
 - usa procedure, funzioni, strutture
 - *C, Basic, Pascal, Fortran, Cobol*
- Il mondo ad oggetti
 - un sistema di cose
 - descritto come gerarchie e dipendenze tra classi
 - usa dichiarazioni di classi e di metodi
 - *Simula, Smalltalk, C++, JAVA*

Indipendente dalla piattaforma

- Non dipende dalla macchina fisica ne dal sistema operativo
- Il codice compilato (**bytecode**) può essere eseguito su qualunque Virtual Machine

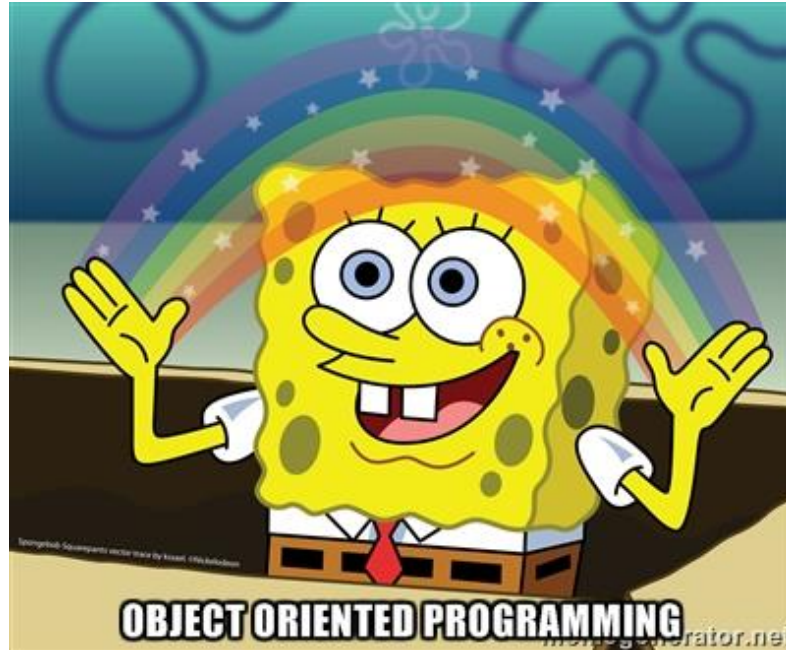


THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.



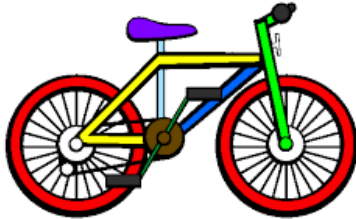
IL MONDO AD OGGETTI

Cosa sono gli oggetti?



- Ogni cosa nel mondo reale è un oggetto
- Ogni oggetto ha uno **stato** e un suo **funzionamento**

Oggetti



Stato

- Marcia corrente
- Velocità di marcia

Funzionamento

- Cambia marcia
- Frena



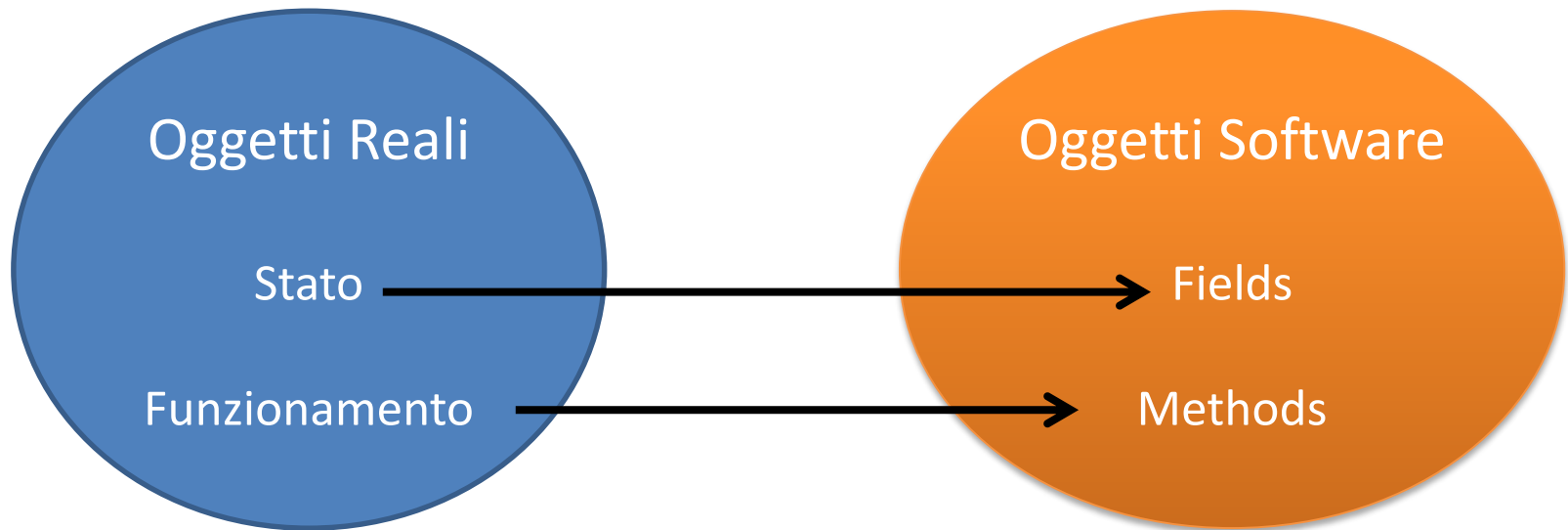
Stato

- Accesa
- Spenta

Funzionamento

- Accendi
- Spegni

Oggetti Software



- Gli oggetti software memorizzano il loro stato in **Fields** (variabili/attributi)
- Gli oggetti software espongono il loro funzionamento attraverso **Methods** (funzioni/metodi)

Oggetti software

- Lo **stato** è dato da i valori che assumono i suoi attributi:
 - Bicicletta -> marcia corrente=5, velocità=30 km/h
- Il funzionamento è dato dai suoi metodi
 - Bicicletta -> cambia marcia
 - I metodi **modificano** lo stato dell'oggetto agendo sui suoi attributi

Oggetti: vantaggi

- **Modularità:** ogni oggetto è indipendente dagli altri, il suo codice può essere gestito separatamente
- **Information-hiding:** i dettagli implementativi di ogni oggetto sono nascosti agli altri oggetti che interagiscono solo con i suoi metodi
- **Riutilizzo** del codice: oggetti scritti da altri possono facilmente essere ri-utilizzati o estesi
- Ogni oggetto può facilmente essere sostituito

Cosa è una classe?

- Nel mondo reale molti oggetti condividono delle caratteristiche
- Raggruppiamo gli oggetti per caratteristiche e funzionamento simile
 - ad esempio tutte le biciclette che abbiamo posseduto sono delle **biciclette**

Cosa è una classe?



← classe

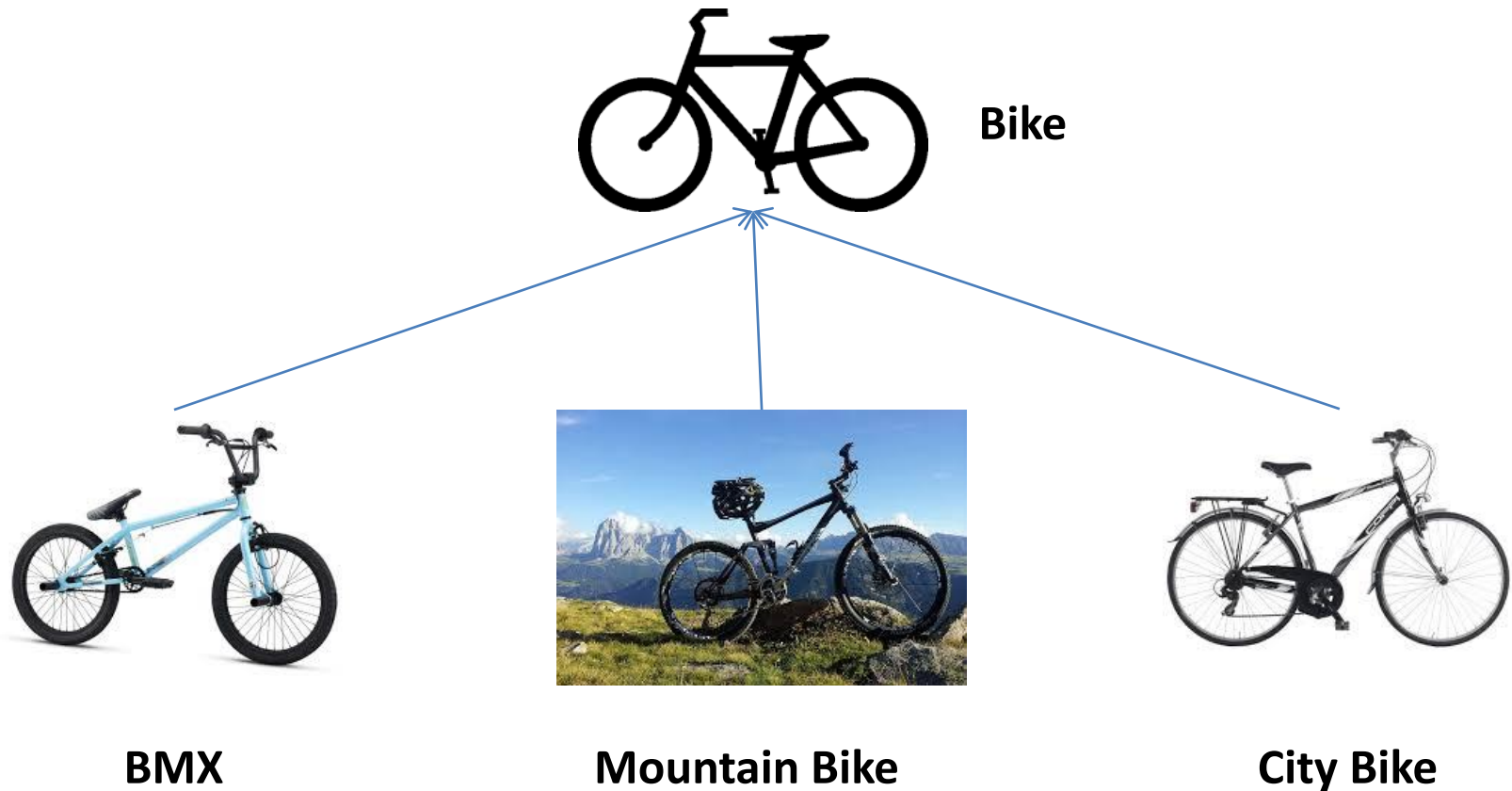
← istanze



Ereditarietà

- Alcune classi di oggetti spesso hanno delle caratteristiche in comune
 - diversi tipi di **bici**: BMX, Mountain Bike, City Bike
- Tutti i tipi di bici hanno caratteristiche in comune ad esempio la velocità corrente o la capacità di frenare
- L'ereditarietà permette ad una classe di ereditare attributi e metodi di un'altra classe

Ereditarietà



BMX, Mountain Bike e City Bike ereditano tutti gli attributi e i metodi di Bike, in più avranno altri metodi e attributi specifici

Ereditarietà

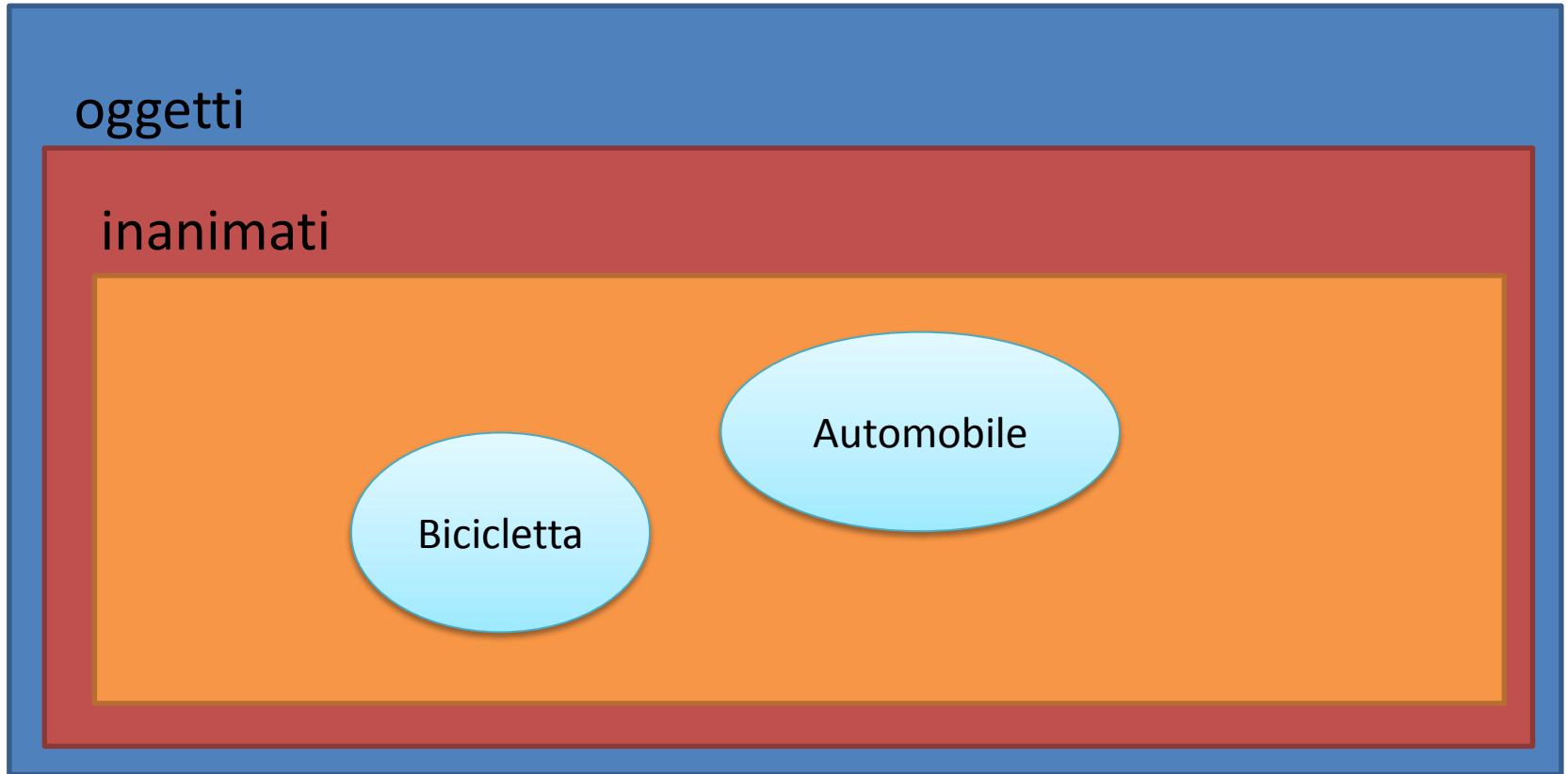
- Le istanze di BMX saranno tutte le biciclette del mondo reale che sono delle BMX
 - tutte le BMX saranno anche delle Biciclette
- L'ereditarietà è un meccanismo potente per rappresentare una gerarchia tra classi

I package

- JAVA permette di organizzare le classi in cartelle separate
- Ogni cartella si chiama package
- Anche i package hanno una gerarchia
 - sottocartelle
- Il percorso completo dei package in cui si trova una classe si chiama **namespace**

I package

mondo



Il **namespace** di Bicicletta è: *mondo.oggetti.inanimati*

```
>HELLO WORLD
```

```
>_
```

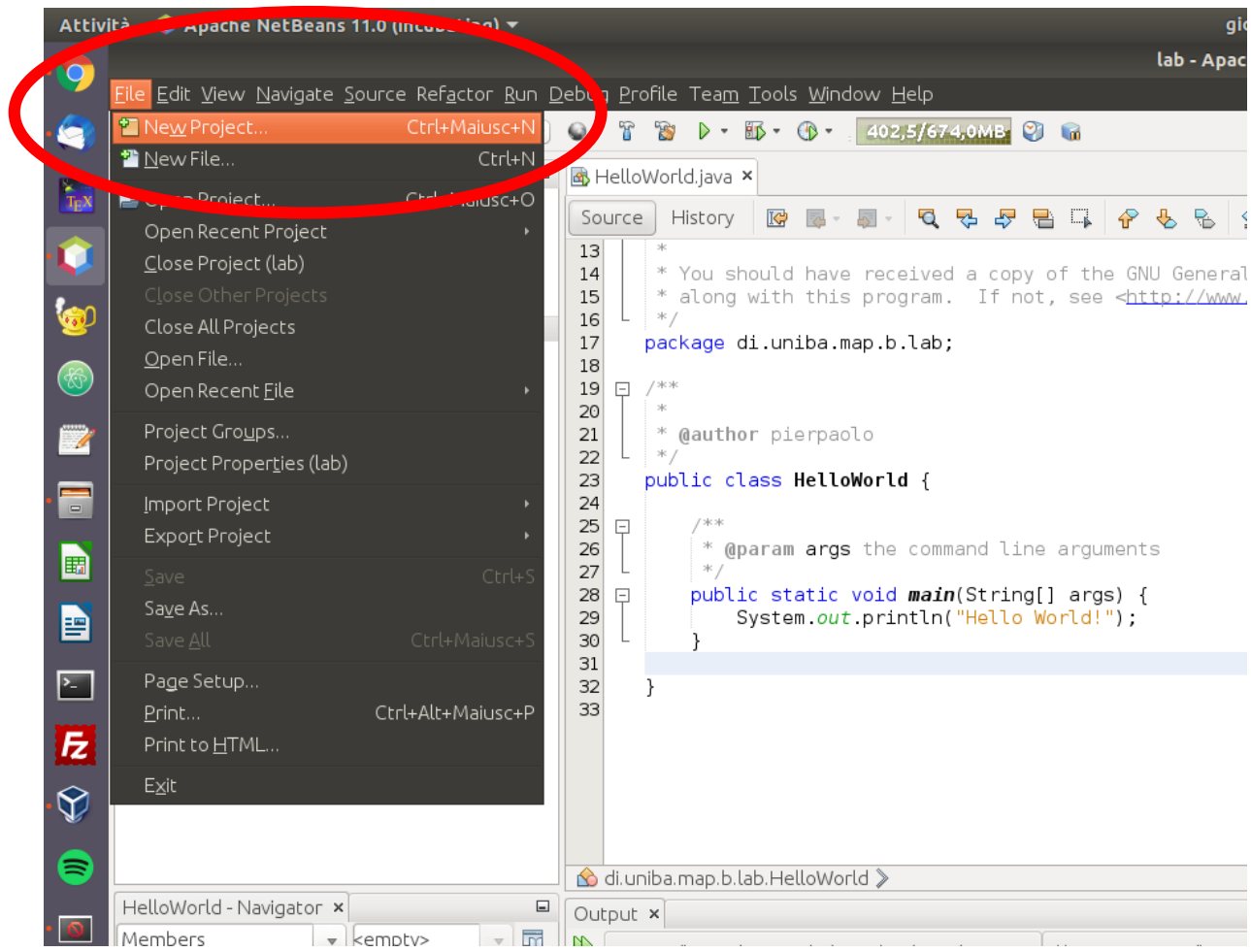
HELLO WORLD

Installazione ambiente

- Installare Java JDK 8 o 11
 - <https://www.oracle.com/technetwork/java/javase/downloads/index.html>
 - scegliere la versione per il proprio sistema operativo
- Installare NetBeans 11 LTS
 - <https://netbeans.apache.org/download/nb110/nb110.html>

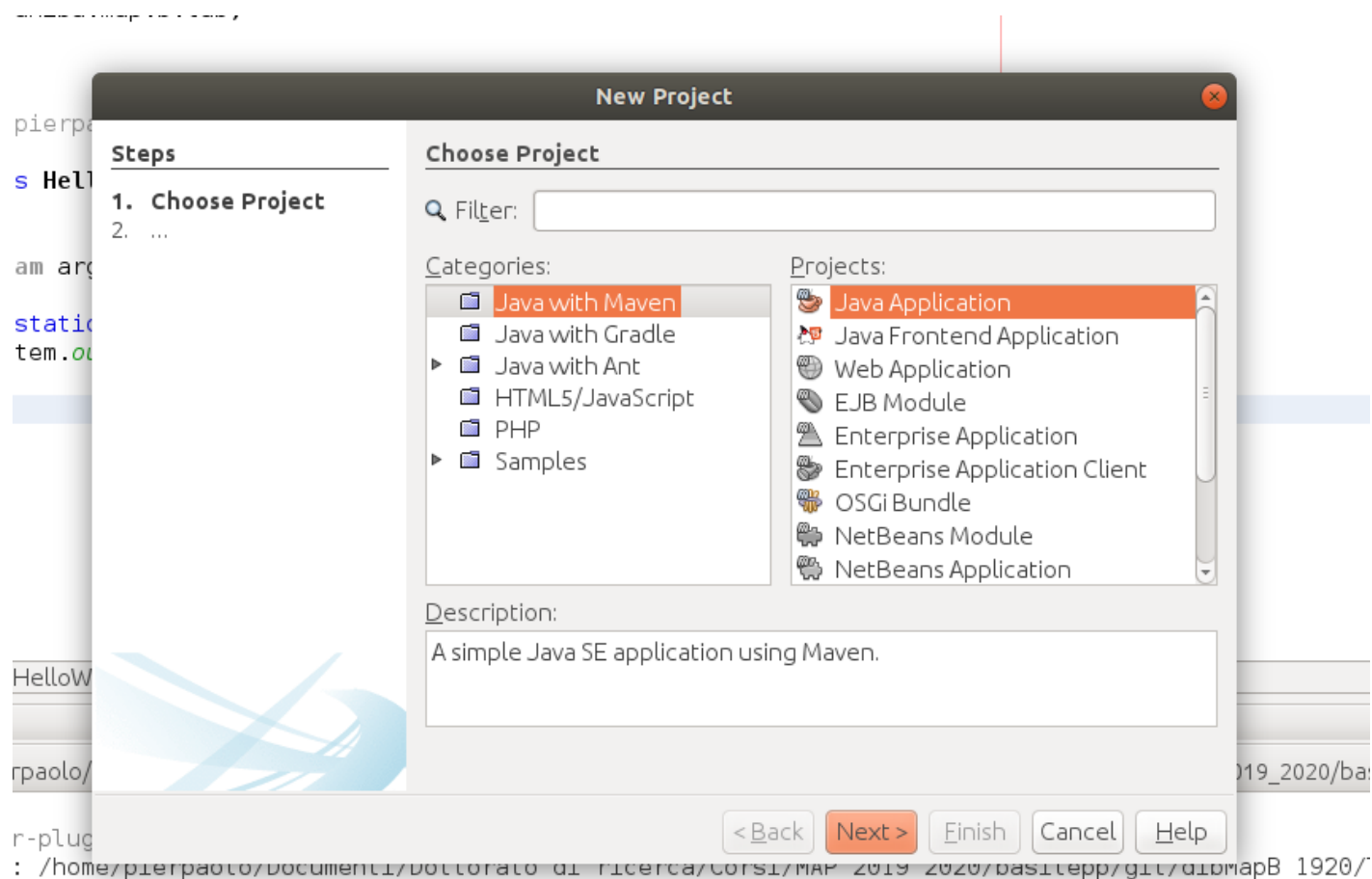
Creare il primo progetto

- Creare un nuovo progetto



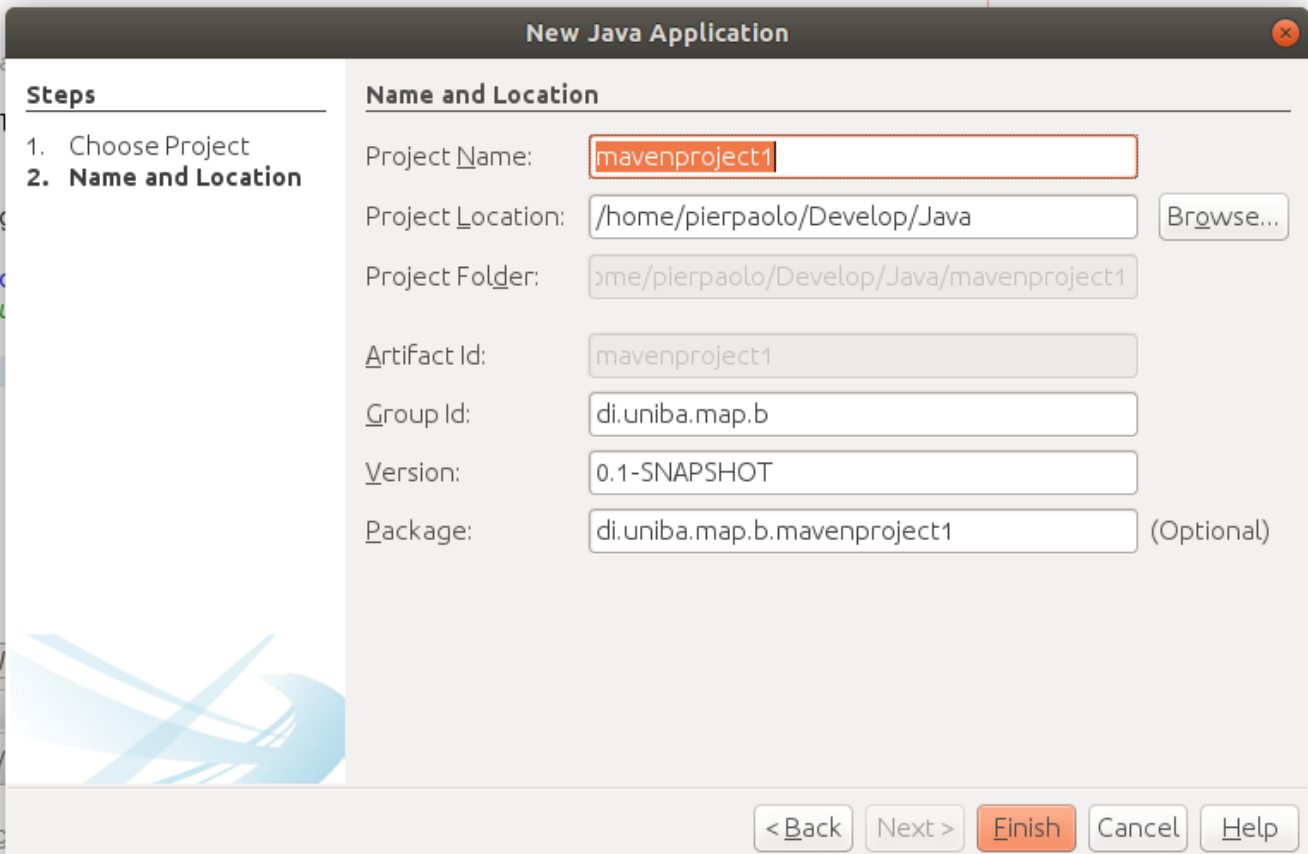
Creare il primo progetto

- Selezionare **Java with Maven** -> **Java Application**



Creare il primo progetto

- Compila i campi richiesti (Project Name e Location)



New Java Application

Steps

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

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

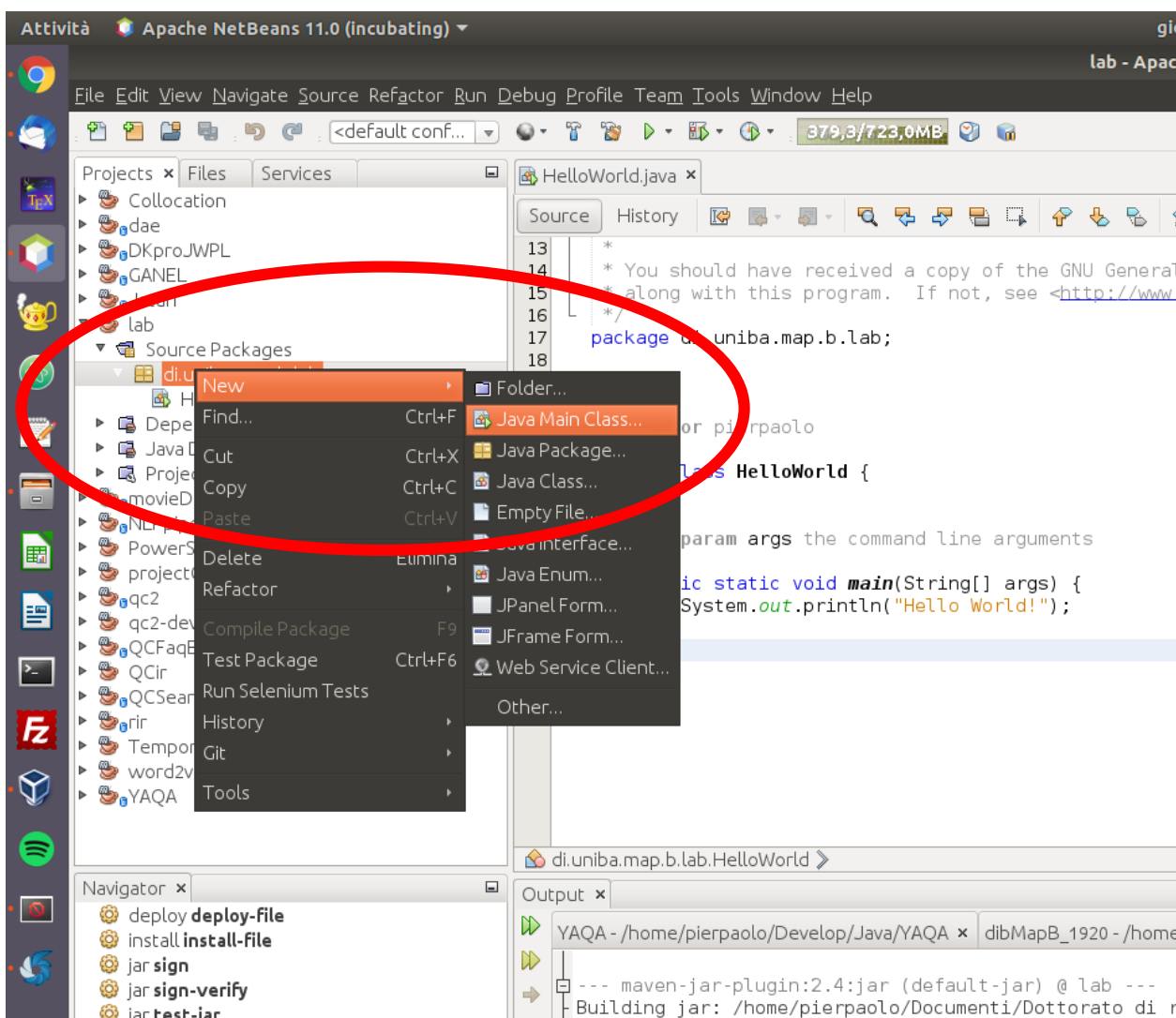
Group Id:

Version:

Package: (Optional)

< Back Next > **Finish** Cancel Help

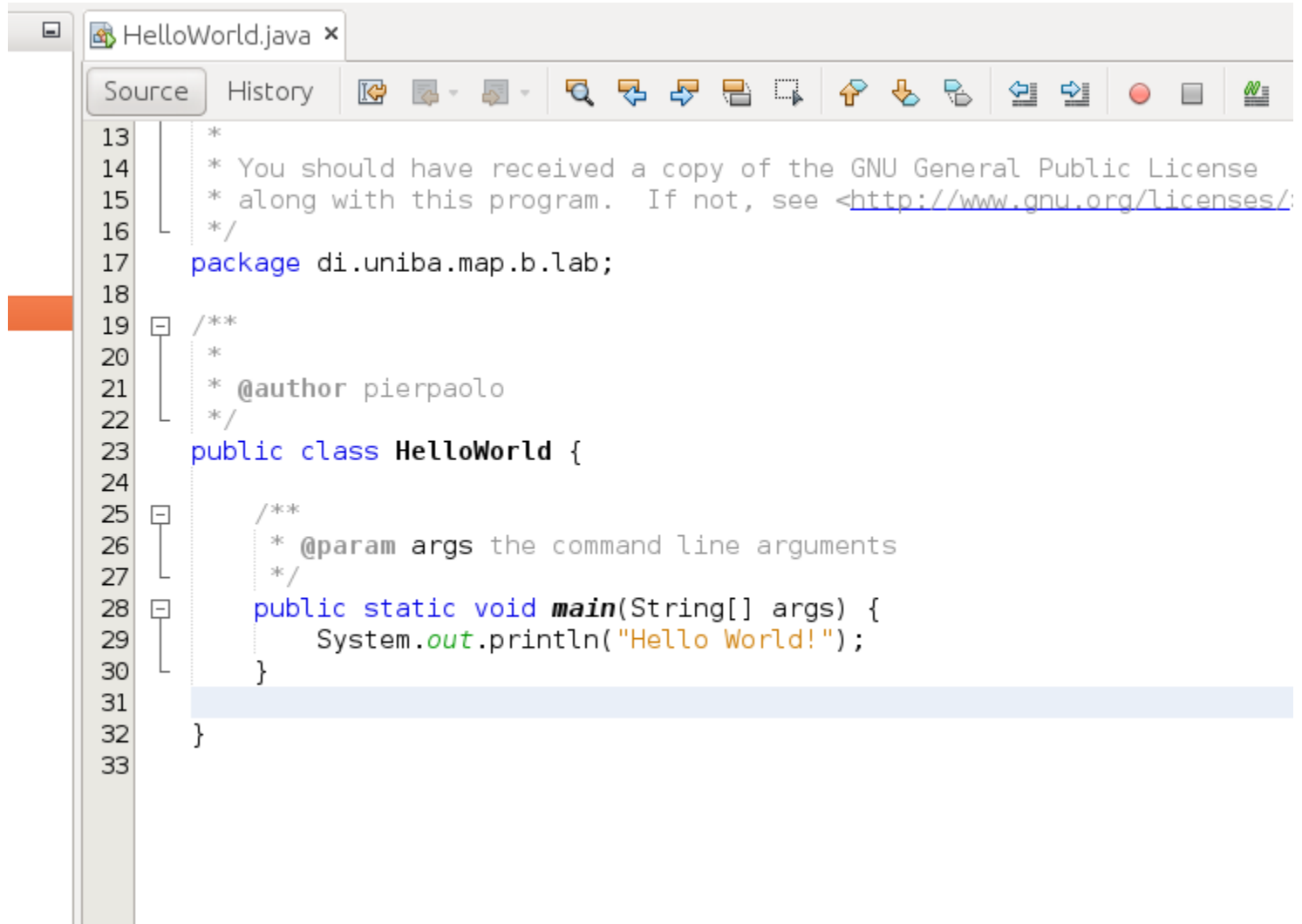
Creare la prima classe



L'albero del progetto contiene il nodo «Source Packages» che contiene i relativi Package del progetto con i vari sorgenti.

- Creare un Package
- Click destro sul Package
- Selezionare New -> Java Main Class
- Nella finestra di dialogo inserire il nome che si vuole dare alla classe

Scrivere il primo programma



```
13  *
14  * You should have received a copy of the GNU General Public License
15  * along with this program.  If not, see <http://www.gnu.org/licenses/>
16  */
17  package di.uniba.map.b.lab;
18
19  /**
20   *
21   * @author pierpaolo
22   */
23  public class HelloWorld {
24
25      /**
26       * @param args the command line arguments
27       */
28      public static void main(String[] args) {
29          System.out.println("Hello World!");
30      }
31
32  }
33
```

Hello World...in JAVA

```
/*  
 * To change this template, choose Tools | Templates  
 * and open the template in the editor.  
 */  
package helloworldapp;
```

Comments

```
/**  
 * The HelloWorldApp class implements an application that  
 * simply prints "Hello World!" to standard output.  
 */  
public class HelloWorldApp {
```

```
/**  
 * @param args the command line arguments  
 */  
public static void main(String[] args) {  
    System.out.println("Hello World!"); // Display the string.  
}  
  
}
```

Commenti

/ text */*

- Il compilatore ignora ogni cosa da */** a **/*.

*/** documentation */*

- Commenti per la documentazione. Il compilatore ignora questi commenti ma sono utilizzati dal tool javadoc per creare automaticamente la documentazione del codice

// text

- Il compilatore ignora ogni cosa da *//* alla fine della linea

Definizione di una classe

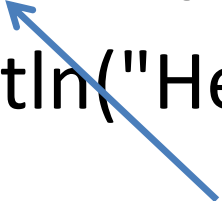
```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```



```
class <name> {  
  
}
```

Il metodo main...


```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello  
        Display the string.  
    }  
}
```



- Ogni applicazione deve avere almeno una classe con un metodo *main*
- Il metodo *main* è l'entry point di ogni applicazione
- Dal *main* possiamo richiamare altre classi e metodi

Il metodo main

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!"); //  
        Display the string.  
    }  
}
```



String[] args → sono gli argomenti che possiamo passare al *main* dalla riga di comando quando invochiamo un'applicazione JAVA

La prima istruzione

```
System.out.println("Hello World!");
```

- *System* è una classe delle librerie «core» di JAVA
- *out* è un suo attributo (un oggetto *PrintStream*)
- *println* è un metodo della classe *PrintStream* che permette di scrivere su un dispositivo di output