Introduction To OOP

- Administration
- Languages overview
- Java
- Eclipse

Course Requirements

- 7 Exercises:
  - 1-3: 6pts
  - 4-7: 8pts
- Final Exam: 50pts

Exercises

- 90% of tests are given to you
- Late submissions: 1pt per day, up to a week
- Extensions: Sickness or miluim only
- Appeals: appeals forum
- Individual work! (talk is ok, but don’t show code)

Contacting the staff

- via forums
- via course mail:

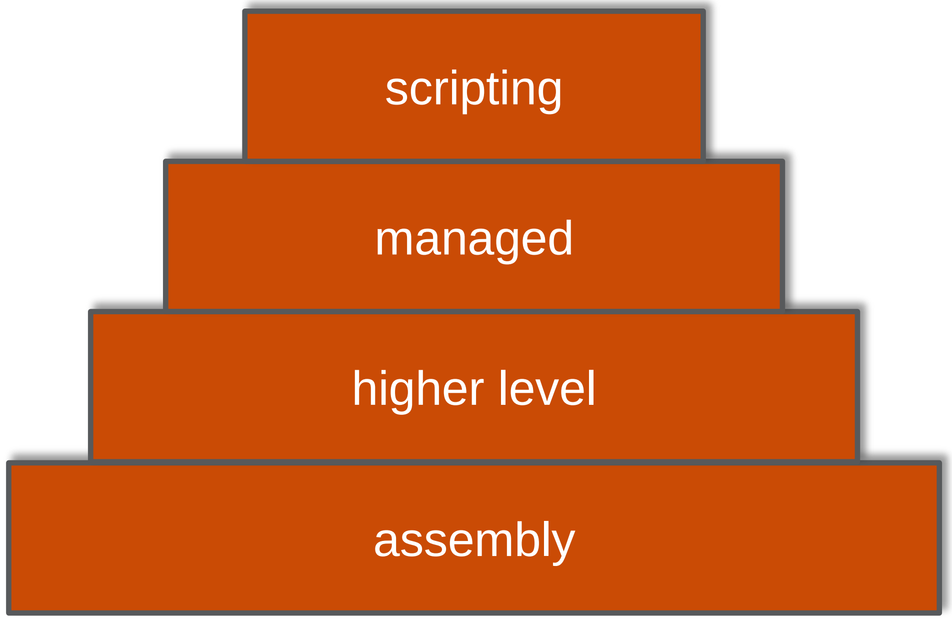
  oophuji@gmail.com

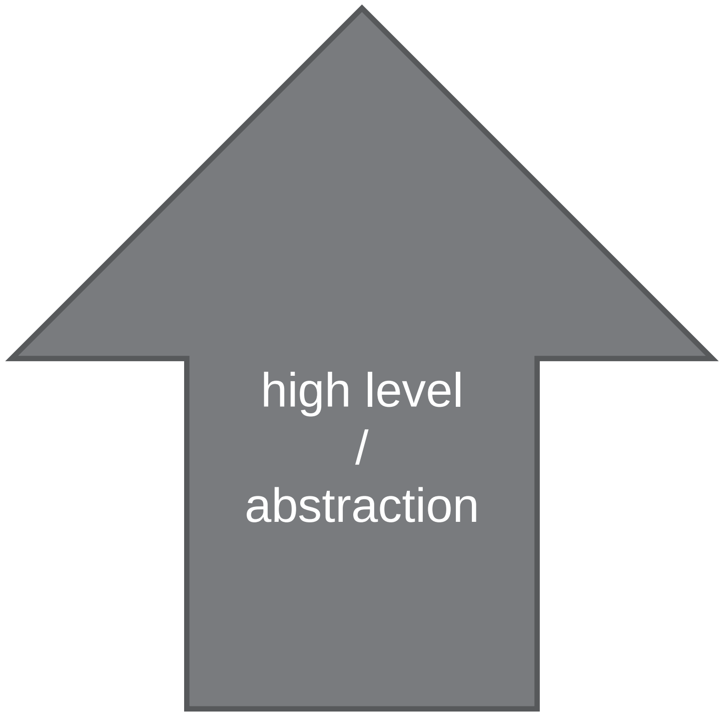
Regulations

- Exercise Preparation & Submission Guidelines
- Coding Style
- Working in Pairs

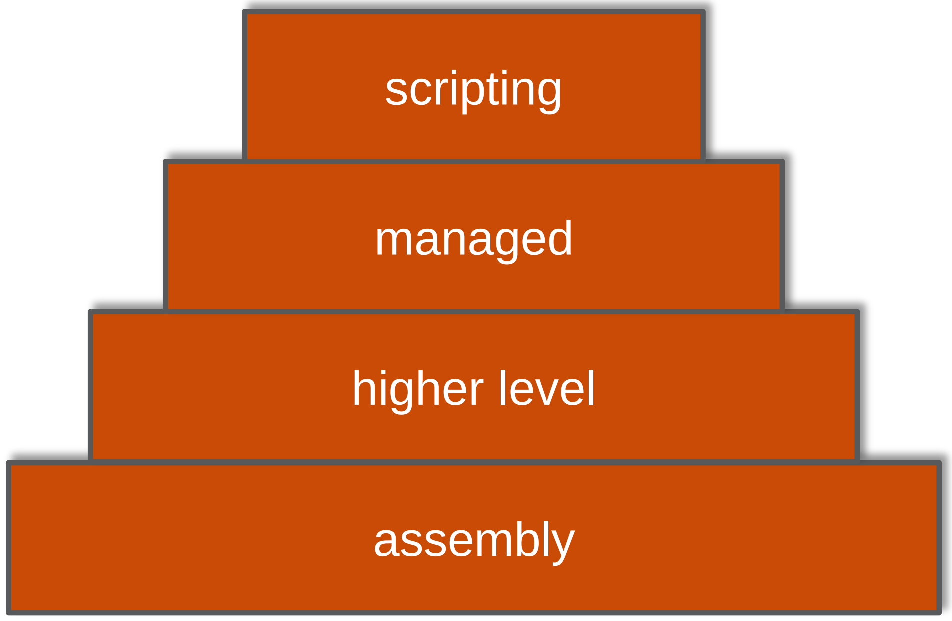
- Must follow **NEWS FORUM**
  - (make sure it reaches your mail)

CLASSIC HIERARCHY OF LANGUAGES



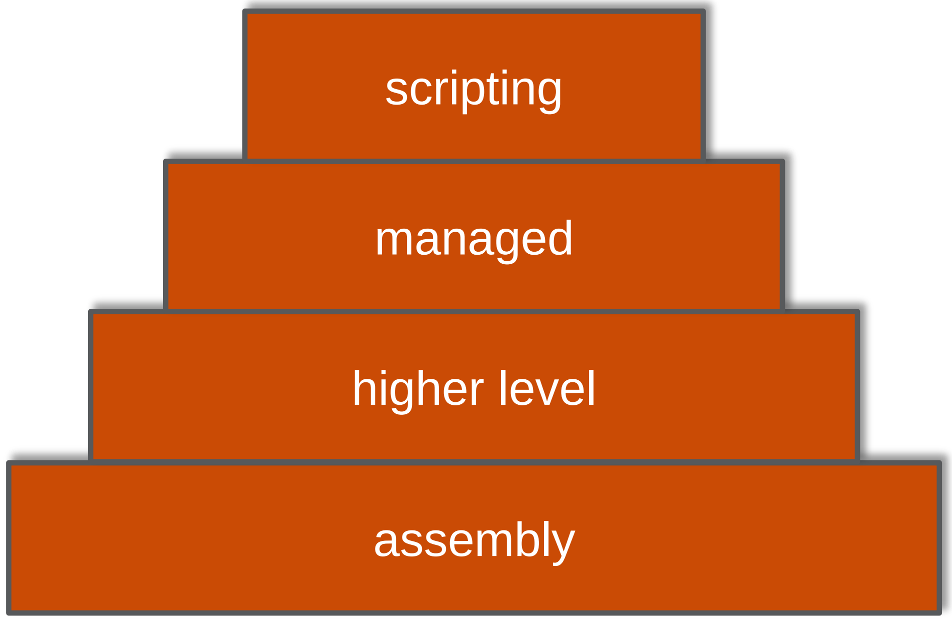


CLASSIC HIERARCHY OF LANGUAGES



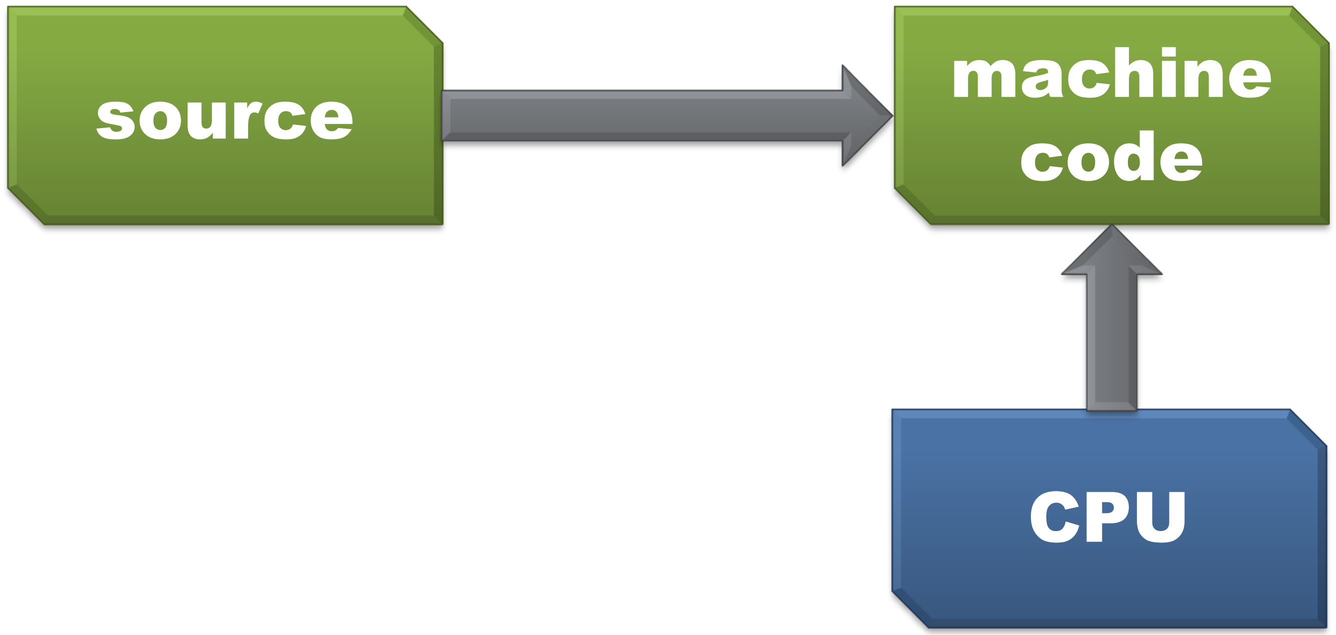
- Python, Perl, Ruby
- JAVA, C#
- C, C++, basic
- assembly languages

MAIN LANGUAGE PROPERTIES

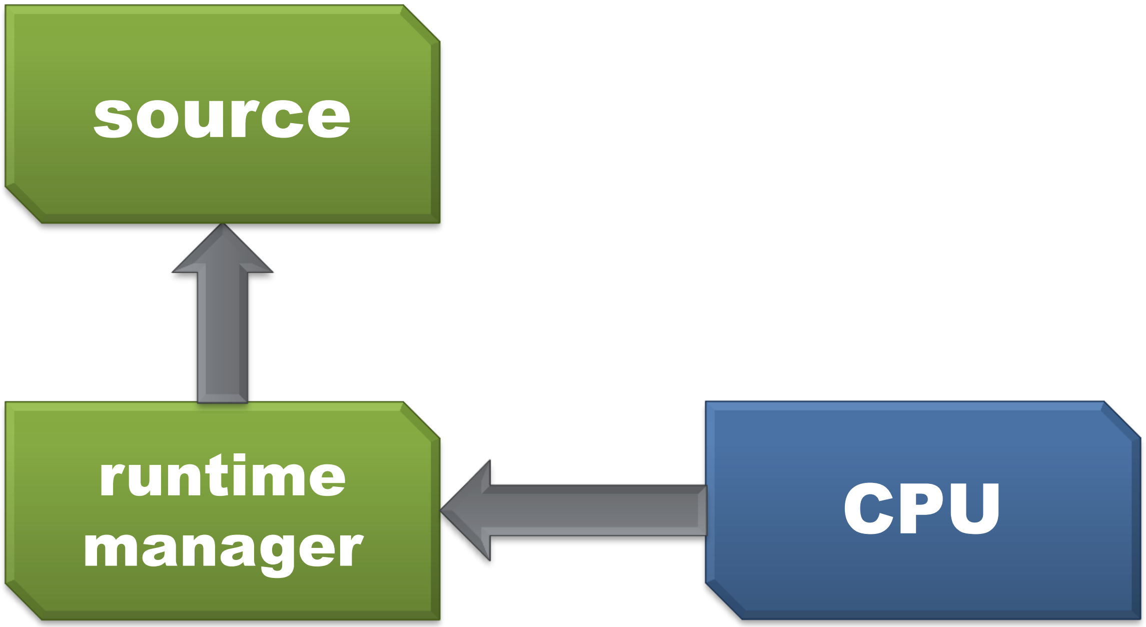


- compilation vs interpretation
- managed vs native

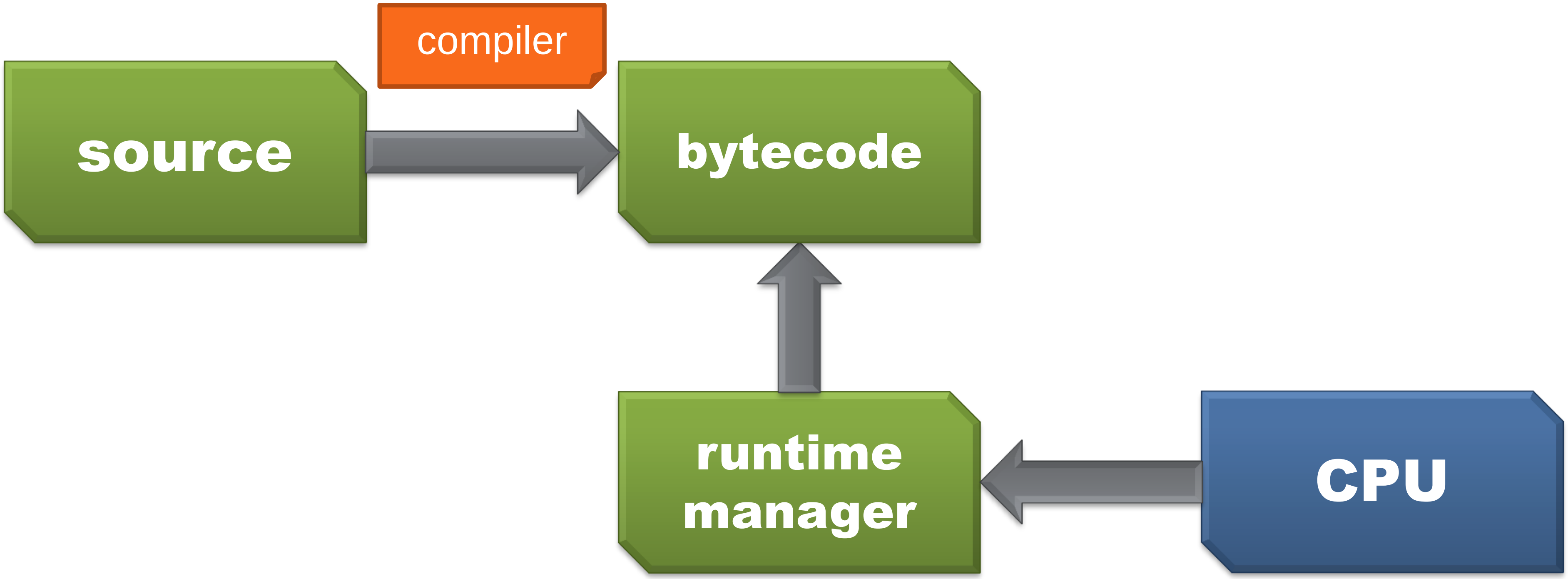
NATIVE CODE



INTERPRETED & MANAGED



COMPILED & MANAGED

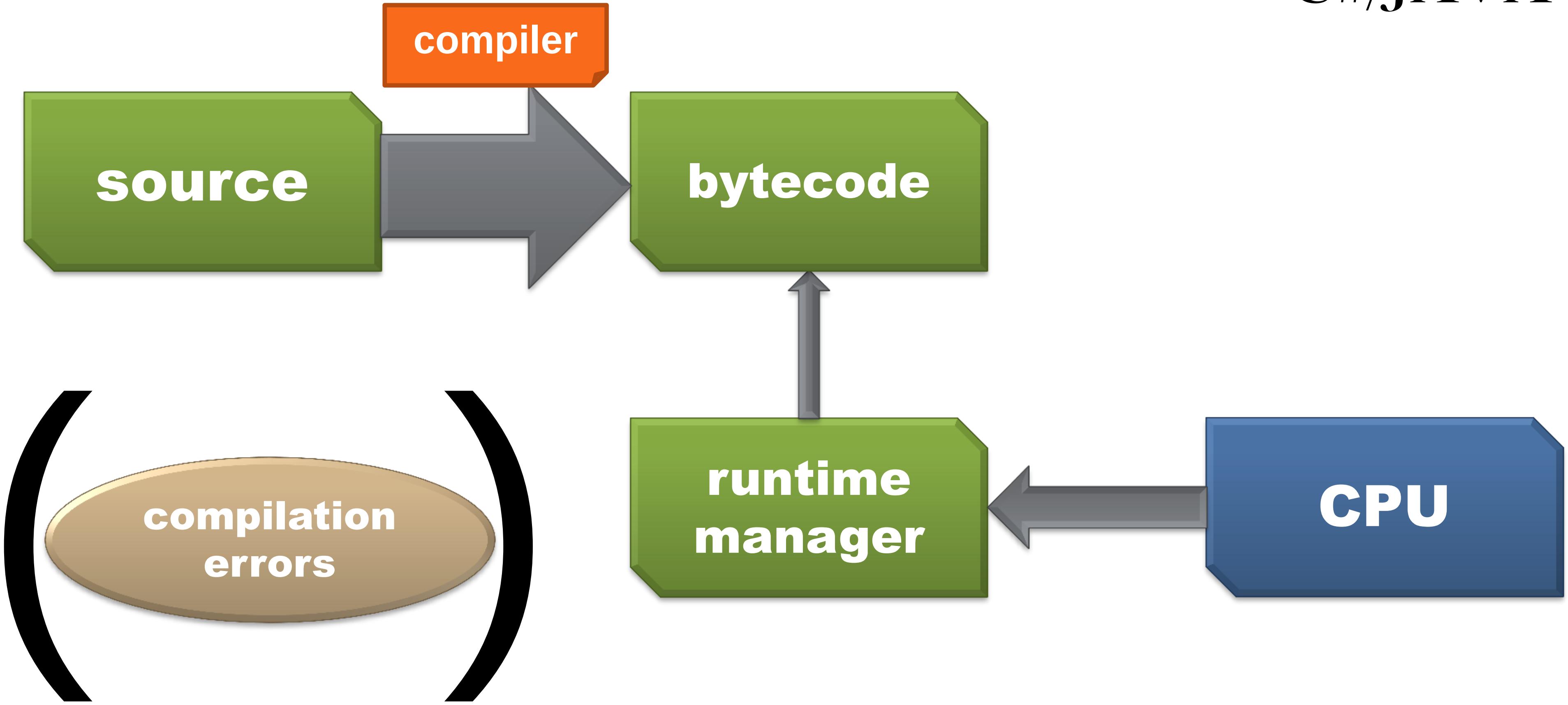


C#, JAVA (and Python?)

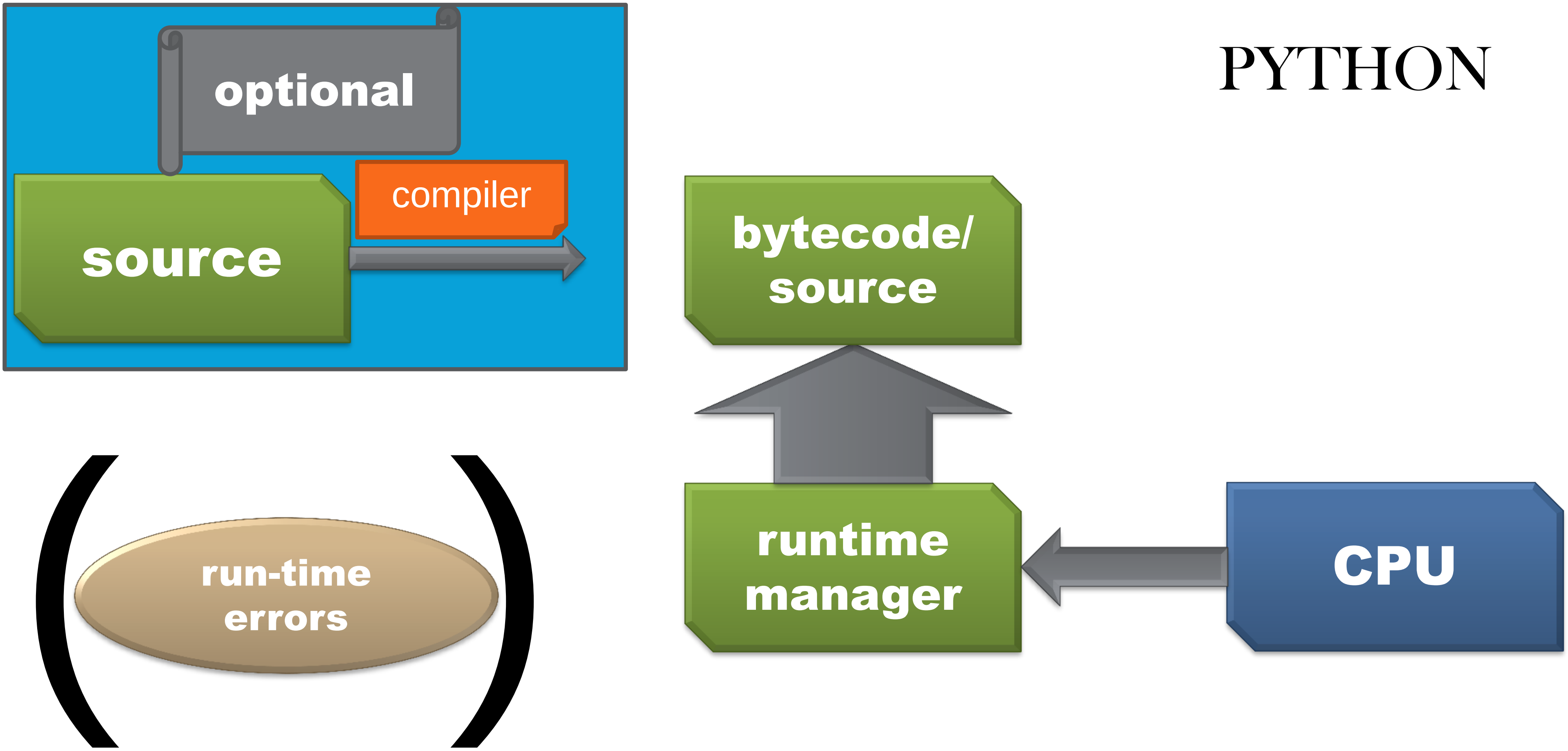
NATIVE VS MANAGED

- speed
- portability
- core updates
- abstraction (memory)

C#/JAVA


compiler
source
bytecode
compilation errors
runtime manager
CPU

PYTHON


optional
compiler
source
bytecode/ source
runtime manager
CPU
run-time errors

ERRORS, BEST TO WORST

- Compile time errors
- Runtime errors
- Logic errors (bugs)
  - That we know of
  - That we don’t know of

Compiled vs. Interpreted

```
var = 1
if (var ==2):
        var = blaBlaBla
else:
        print(‘program ended successfully');
```

- Python: Very basic pass (indentation check)
- Lines are parsed **when reached**

“program ended successfully”

very nice program you got there

Compiled vs. Interpreted

```
var = 1
if (var ==2):
        var = blaBlaBla
else:
        print('program ended successfully');
```

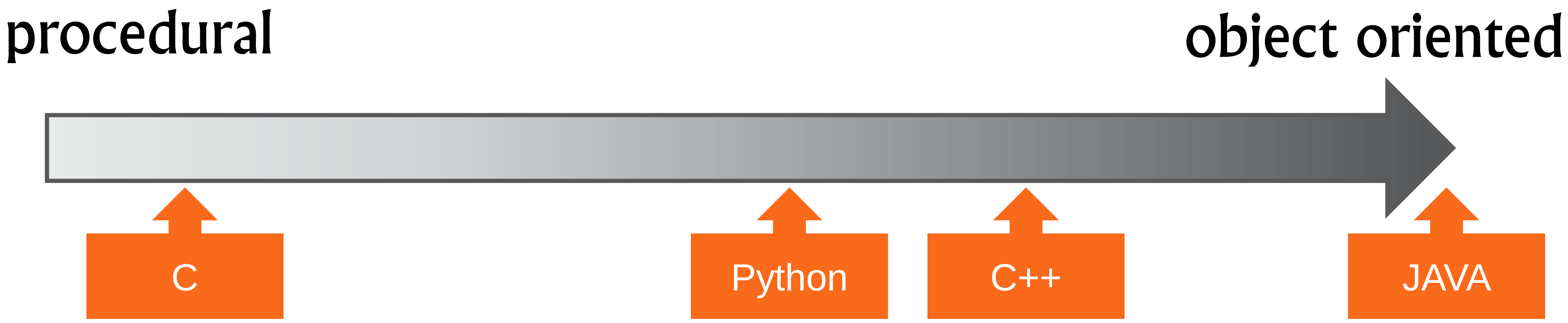
- Java:

WTF undefined variable

NO

fix fix FIX
no run, no run

OBJECT ORIENTATION



Development Tools Evolution

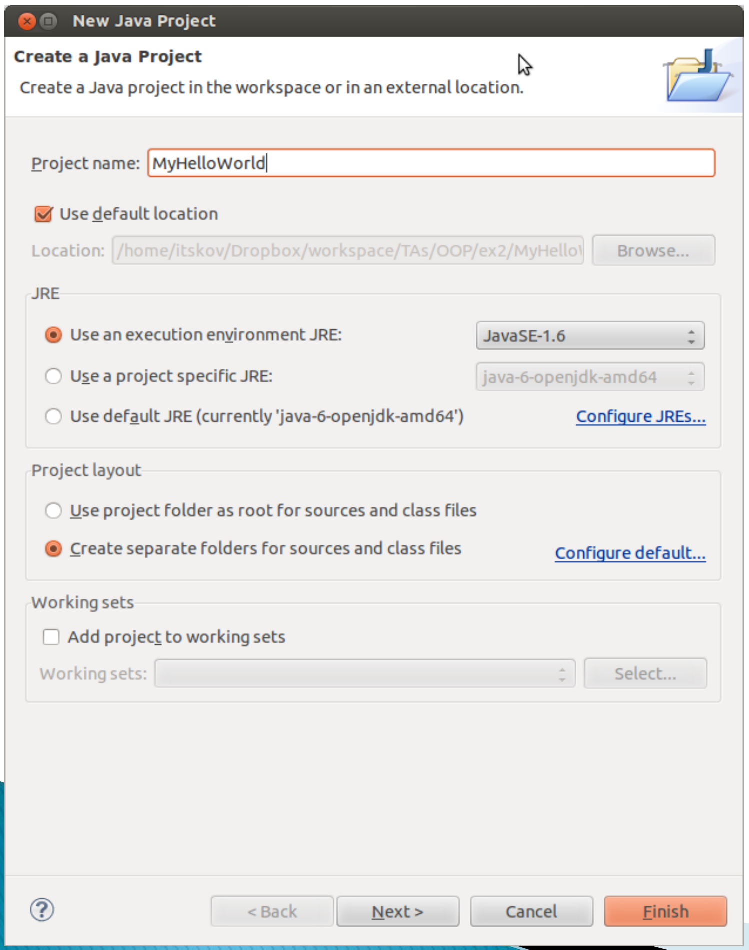
- Editor: Notepad++, gedit, vim, emacs…
- Compiler (javac, gcc…)
- **Debugger**

- **IDE:** Integrated Development Environment
  - Visual Studio
  - Komodo
  - **Eclipse**

Eclipse Demo

Glimpse Of Eclipse

- Open Source (GNU License)



New Java Project

Create a Java Project

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

Project name: MyHelloWorld

[✓] Use default location

Location: /home/itskov/Dropbox/workspace/TAs/OOP/ex2/MyHello\ Browse...

JRE

[✓ Use an execution environment JRE:] JavaSE-1.6

[ ] Use a project specific JRE: java-6-openjdk-amd64

[ ] Use default JRE (currently 'java-6-openjdk-amd64') Configure JREs...

Project layout

[ ] Use project folder as root for sources and class files

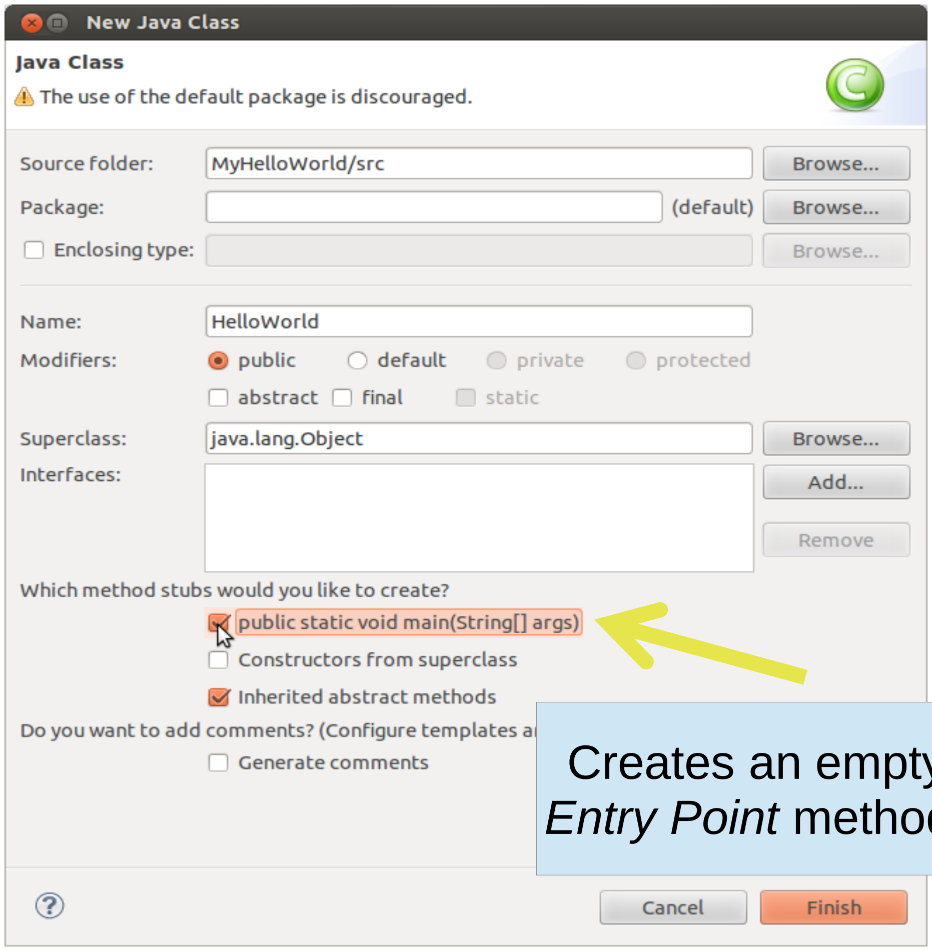
[✓ Create separate folders for sources and class files] Configure default...

Working sets

[ ] Add project to working sets

Working sets: Select...

< Back Next > Cancel Finish



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: MyHelloWorld/src Browse...

Package: (default) Browse...

[ ] Enclosing type: Browse...

Name: HelloWorld

Modifiers: [✓ public] [ ] default [ ] private [ ] protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

[✓] public static void main(String[] args)

[ ] Constructors from superclass

[✓] Inherited abstract methods

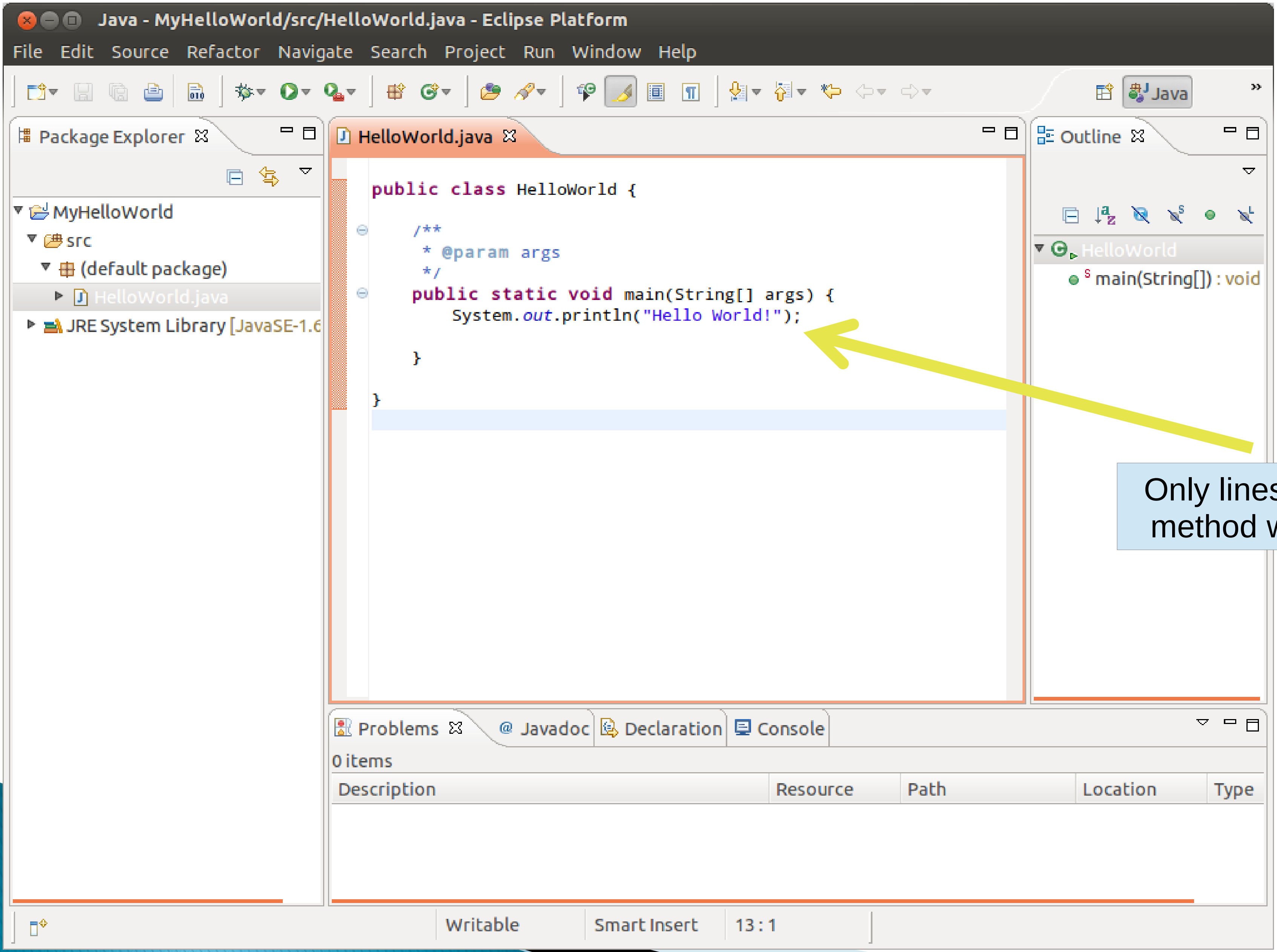
Do you want to add comments? (Configure templates a

[ ] Generate comments

Cancel Finish

Creates an empty Main *Entry Point* method for us

Hello World !



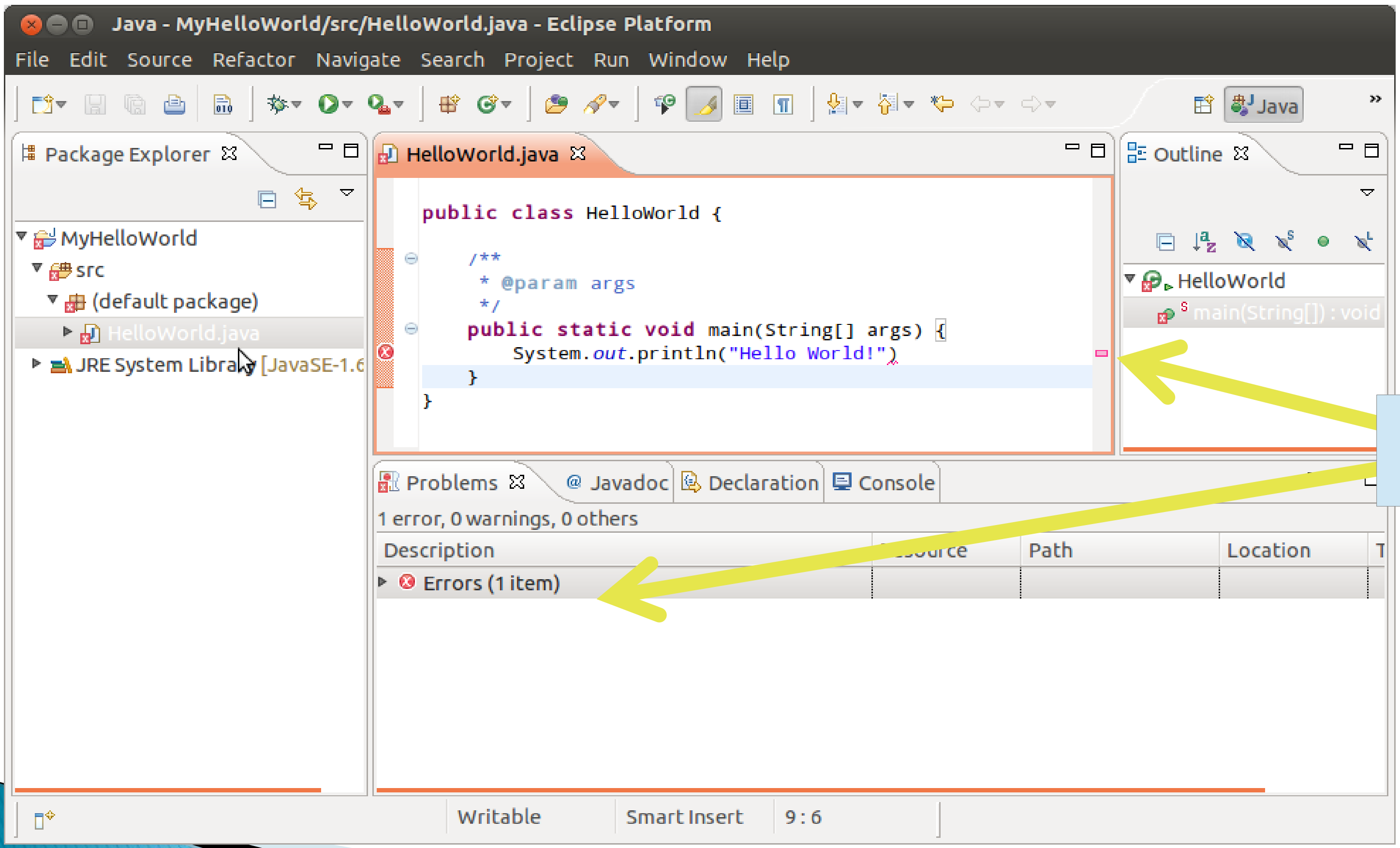
```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");

    }

}
```

Description	Resource	Path	Location	Type

Only lines inside the main method will be executed.

Compiling and Running

- Ctrl + F11: Compiles and Runs the code.



Java - MyHelloWorld/src/HelloWorld.java - Eclipse Platform

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World!")
    }
}
```

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	T
Errors (1 item)				

Code won't compile - compilation error

Java Primitive Types

- Java: statically-typed
- **all variables must be declared before they can be used.**

Name	Content	Size
int	integers	4 bytes
char	single characters	2 bytes
float	real numbers	4 bytes
double	real numbers	8 bytes
boolean	true/false	Undefined
byte	raw data	1 byte

Primitive Types Implicit Cast

- Arithmetic result type:
  type of “more complicated” operand
- **casting** is used for:
  - Adding precision
  - Explicitly losing precision
  - Sometimes just to make java happy

Primitive Types Implicit Cast

- Adding precision:
  - `float res = 5 / 2;                //res is 2.0`
  - `float res = (float) 5 / 2;        //res is 2.5`
  - `or: float res = 5f / 2;           //res is 2.5`
- Losing precision:
  - `int res = (int) 5.7f;`
- Making Java happy:
  - `int res = (int) Math.round(5.7);`

What's the actual reason?

/ / ?