Introduction to Java Programming Language

Desenvolvimento de Software e Sistemas Móveis (DSSMV)
Licenciatura em Engenharia de Telecomunicações e Informática
LETI/ISEP

2025/26

Paulo Baltarejo Sousa
pbs@isep.ipp.pt

Instituto Superior de Engenharia do Porto

Disclaimer

Material and Slides

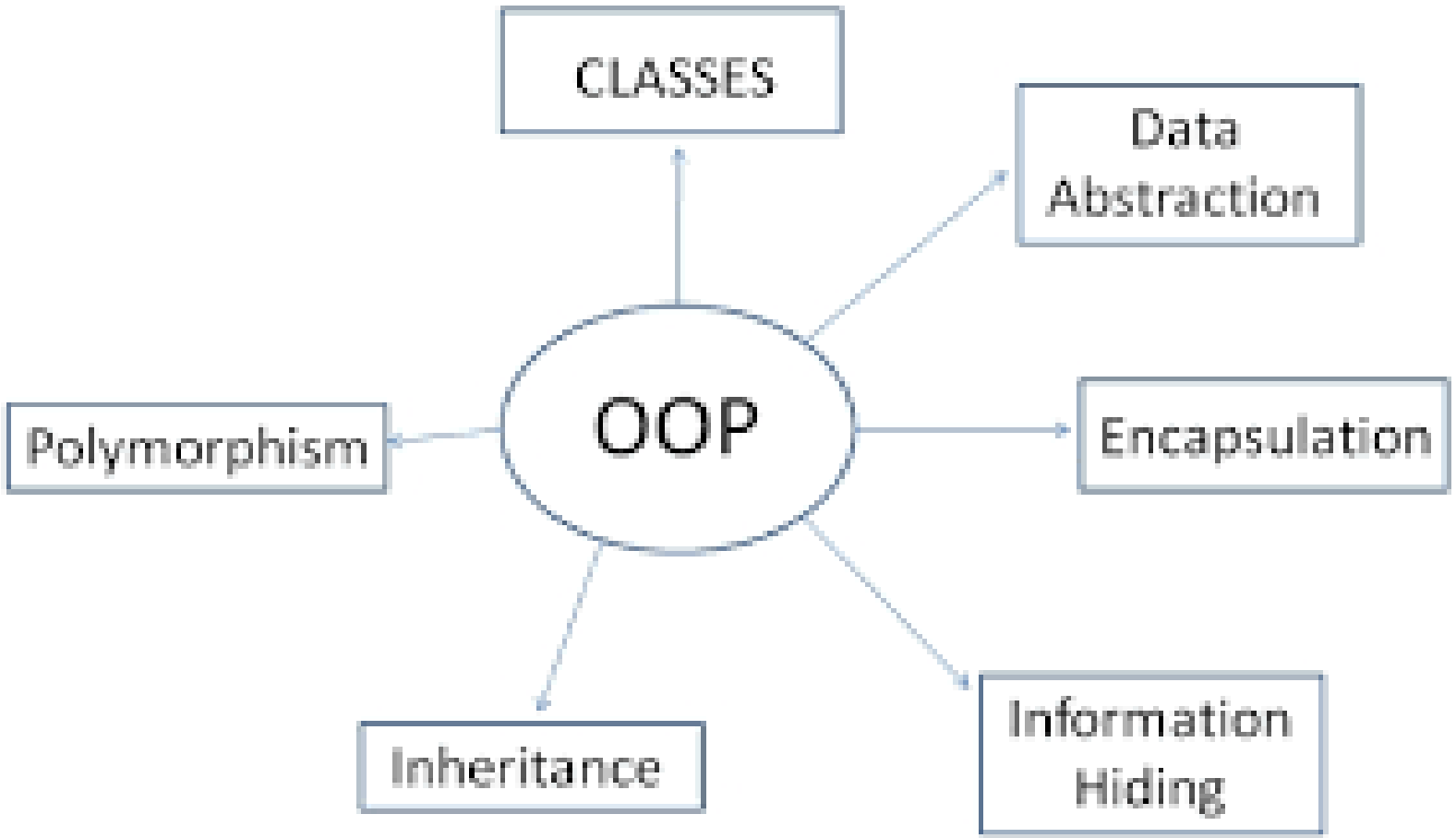
Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

Outline

Object-Oriented Programming Concepts

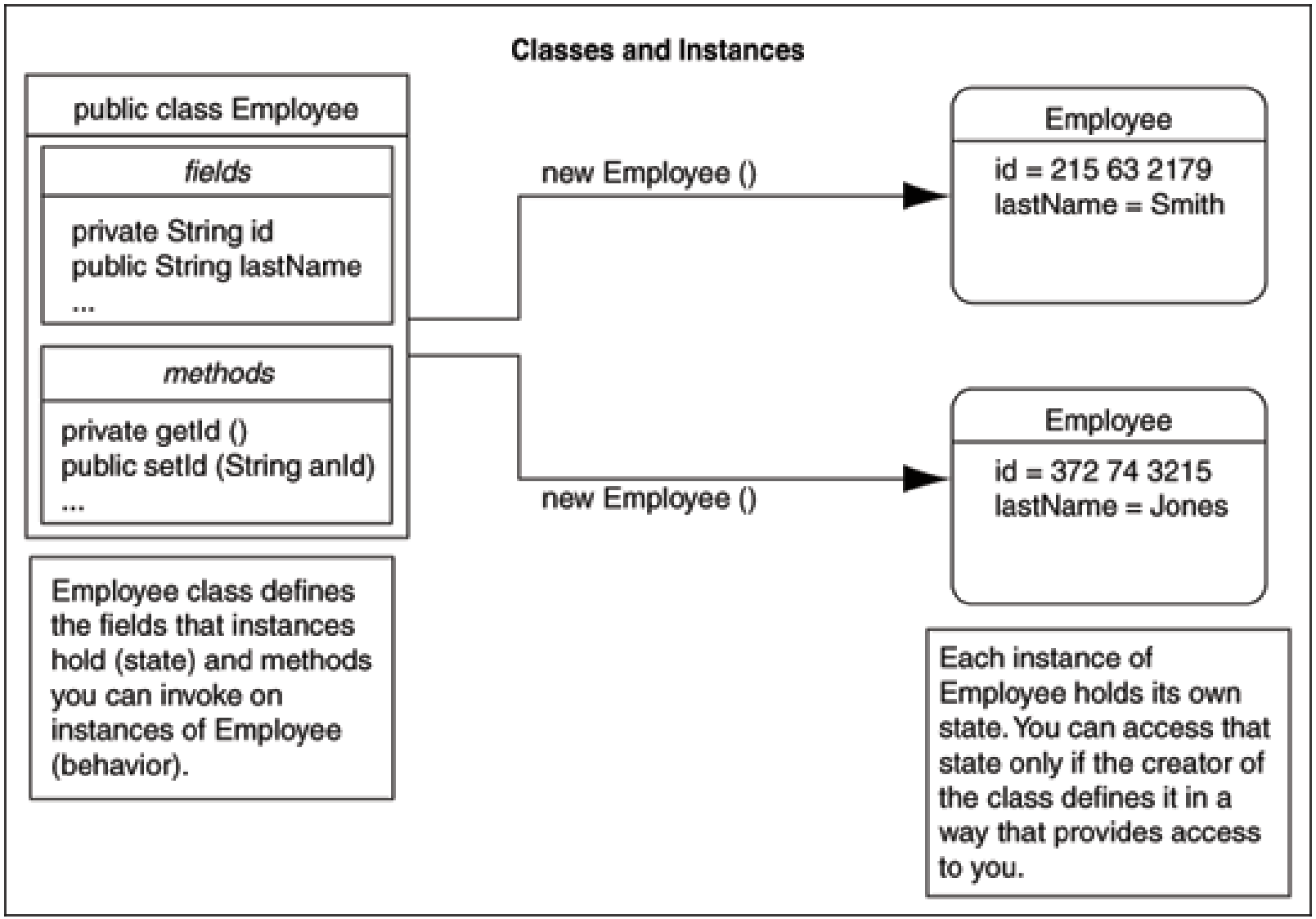
Object-Oriented Programming (OOP) (I)



OOP (II)

- Writing object-oriented programs involves creating **classes**, which are blueprints for **objects**;
- A class definition describes what **attributes** its objects will have and what those objects will be able to do.
- Attributes are the characteristics that define an object; they are **properties** of the object.
- An **object** is a specific, concrete **instance** of a class.
  - The values contained in an object's properties often differentiate instances of the same class from one another.
- A **method** is a self-contained block of program code that carries out some action.

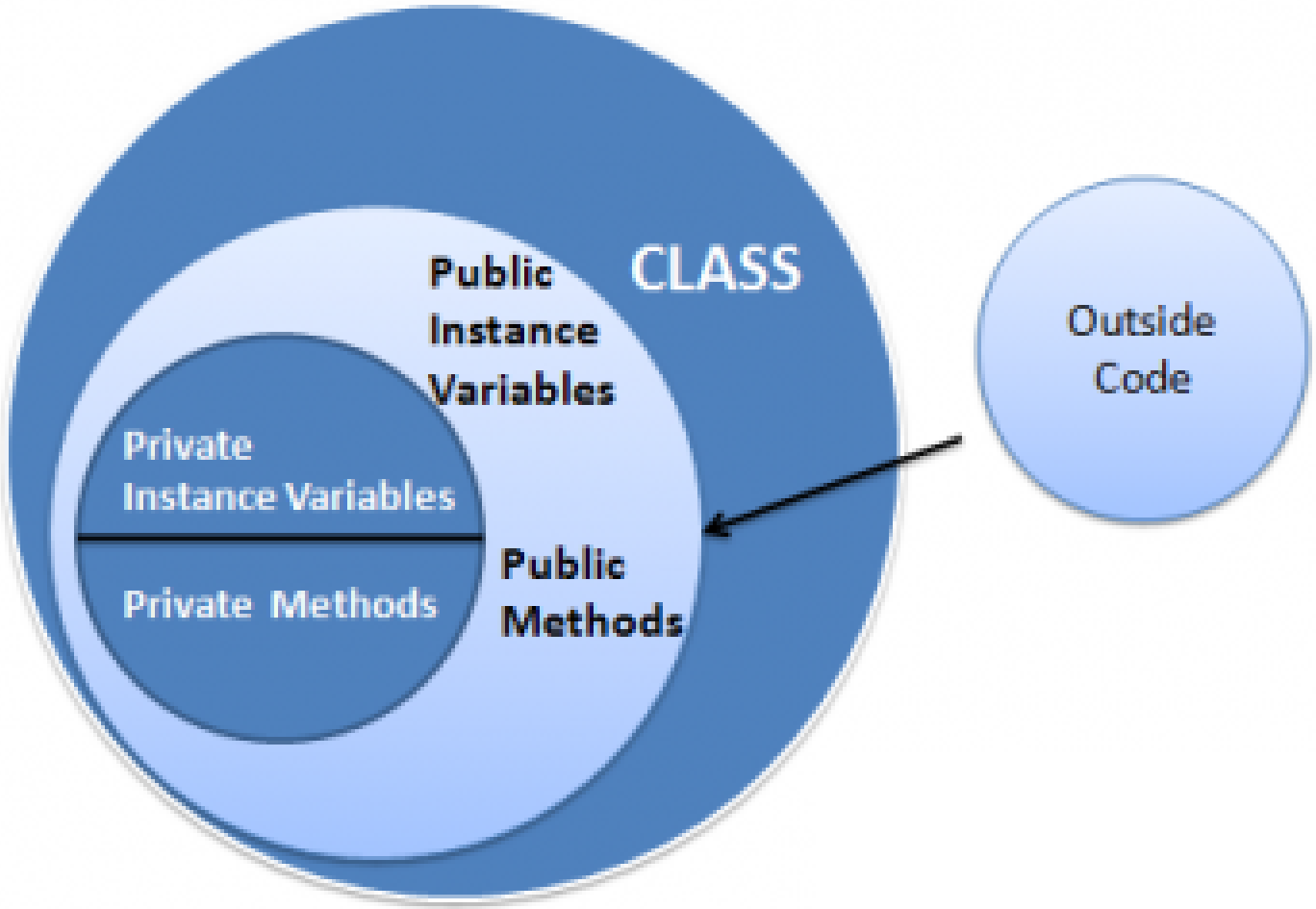
OOP: Classes and Objects



OOP: Encapsulation & Hiding (I)

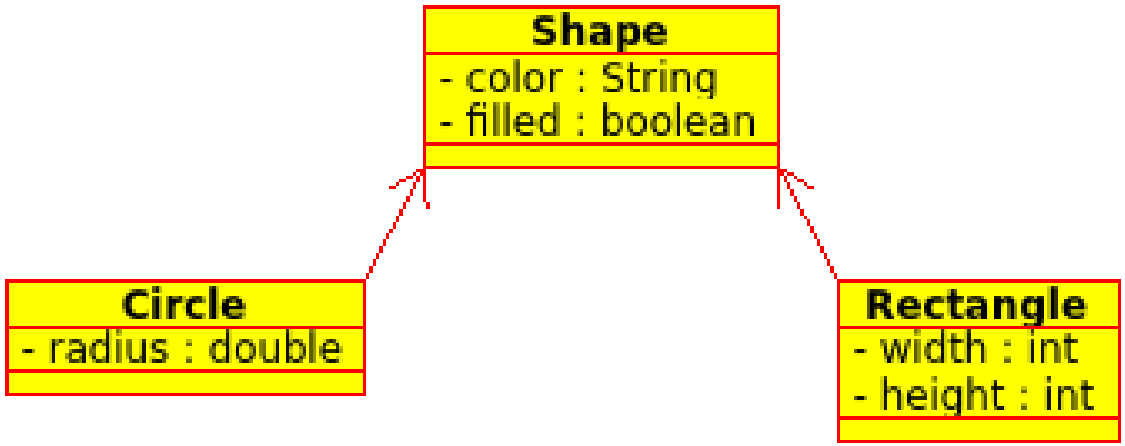
- In object-oriented classes, attributes and methods are **encapsulated** into objects.
  - It is the enclosure of data and methods within an object.
- Encapsulation allows you to treat all of an object's methods and data as a single entity.
- It also refers to the concealment (**hiding**) of an object's data and methods from outside sources.
  - Concealing data is sometimes called information hiding, and concealing how methods work is implementation hiding
  - It lets you hide specific object attributes and methods from outside sources and provides the security that keeps data and methods safe from inadvertent changes.

OOP: Encapsulation & Hiding (II)



OOP: Inheritance

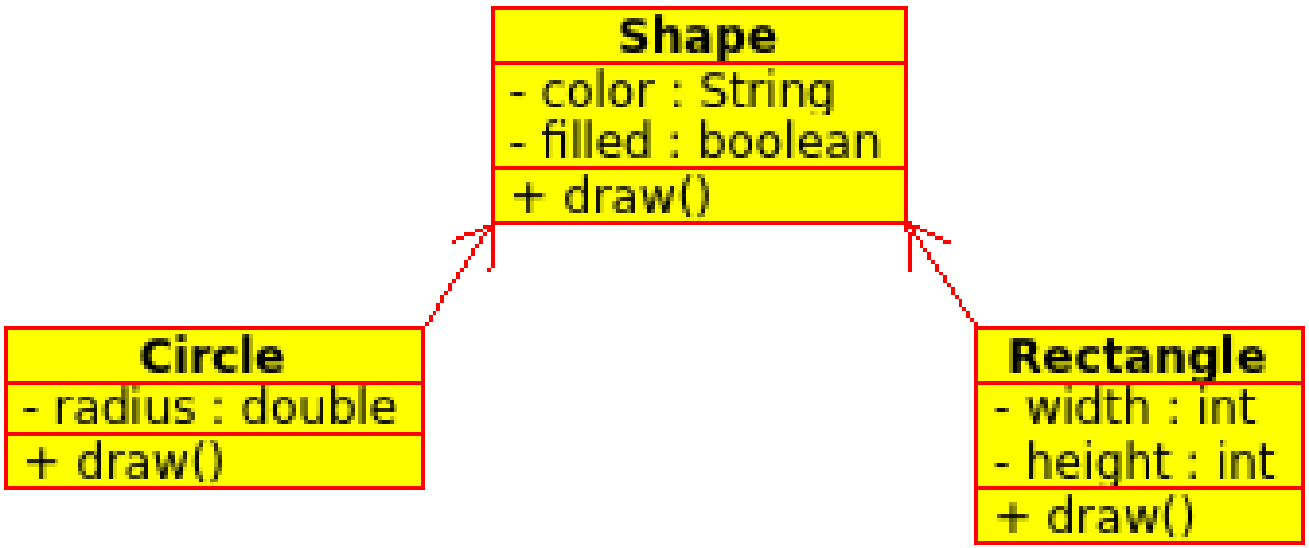
- **Inheritance** is the ability to create classes that share the attributes and methods of existing classes but with more specific features.



- When a **Class** extends another class it inherits all non-private members including fields and methods.
  - It can be best understood in terms of Parent and Child relationship, also known as **Super class** (Parent) and **Sub class** (child) in Java language.

OOP: Polymorphism

- **Polymorphism** means "many forms".
  - It could be static and dynamic both.
- **Overloading** is static polymorphism while, **overriding** is dynamic polymorphism.
  - Overloading in simple words means two methods having same method name but takes different input parameters.
  - Overriding means a derived class is implementing a method of its super class.



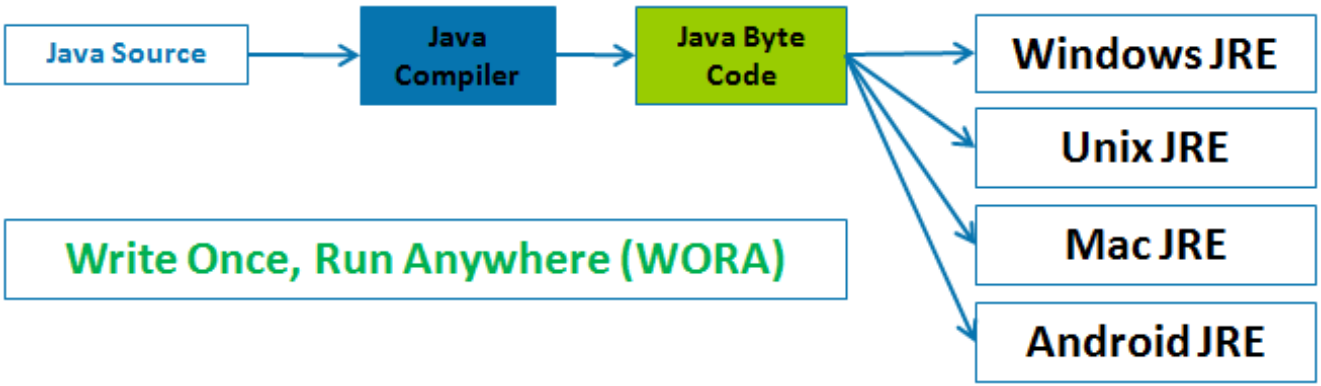
Features of the Java Programming Language

Java (I)

- **Java** was developed by Sun Microsystems as an object-oriented language for general–purpose applications and for interactive, World Wide Web–based Internet applications.
- You can use Java to write a program that runs on any operating system (such as Windows, Mac OS, or Linux) or device (such as PCs, phones, and tablet computers).
  - Java can be run on a wide variety of computers and devices because it does not execute instructions on a computer directly. Instead, Java runs on a hypothetical computer known as the **Java Virtual Machine** (JVM).
- Java is a high-level programming language.
- The Java application source code is saved in a file; then, the Java compiler converts the source code into a binary program of **bytecode**.

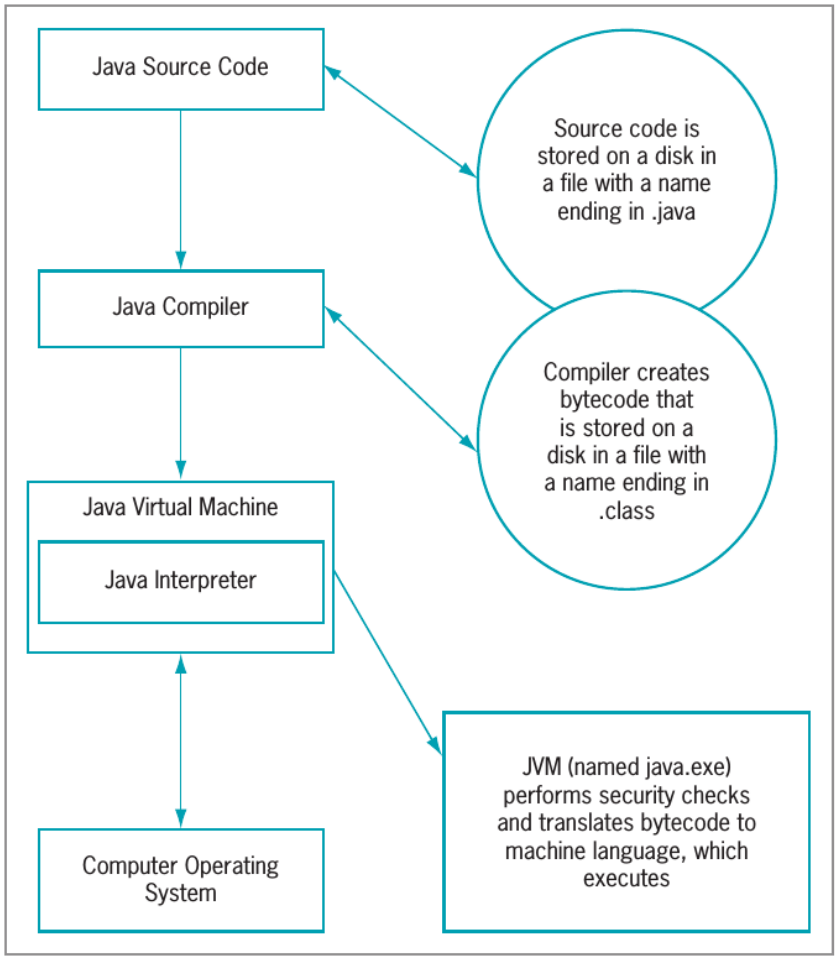
Java (II)

- **Write Once, Run Anywhere (WORA)** is a slogan developed by Sun Microsystems to describe the ability of one Java program version to work correctly on multiple platforms.



Java (III)

- A program called the **Java interpreter** then checks the **bytecode** and communicates with the operating system, executing the bytecode instructions line by line within the **Java Virtual Machine**.
  - Because the Java program is isolated from the operating system, the Java program also is isolated from the particular hardware on which it is run.



Java language (keywords)

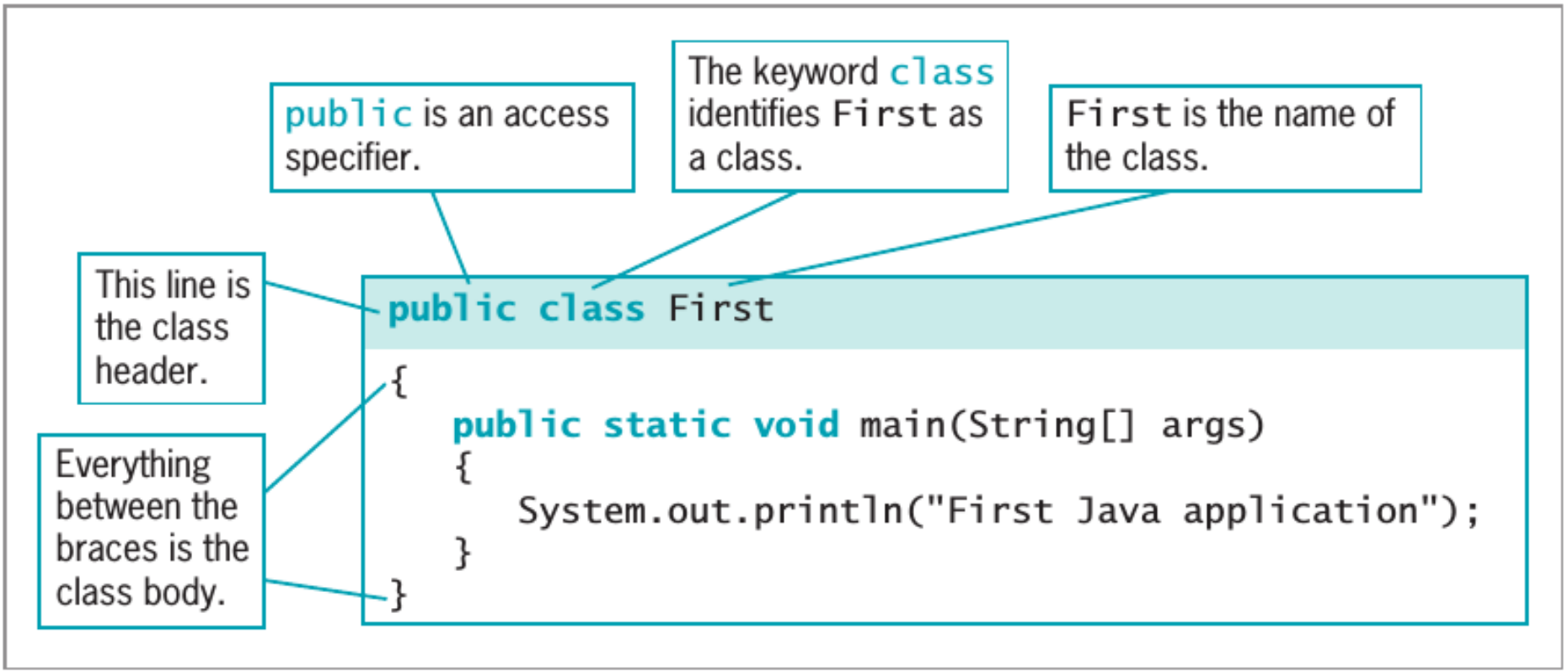
abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Simplest Java application (I)

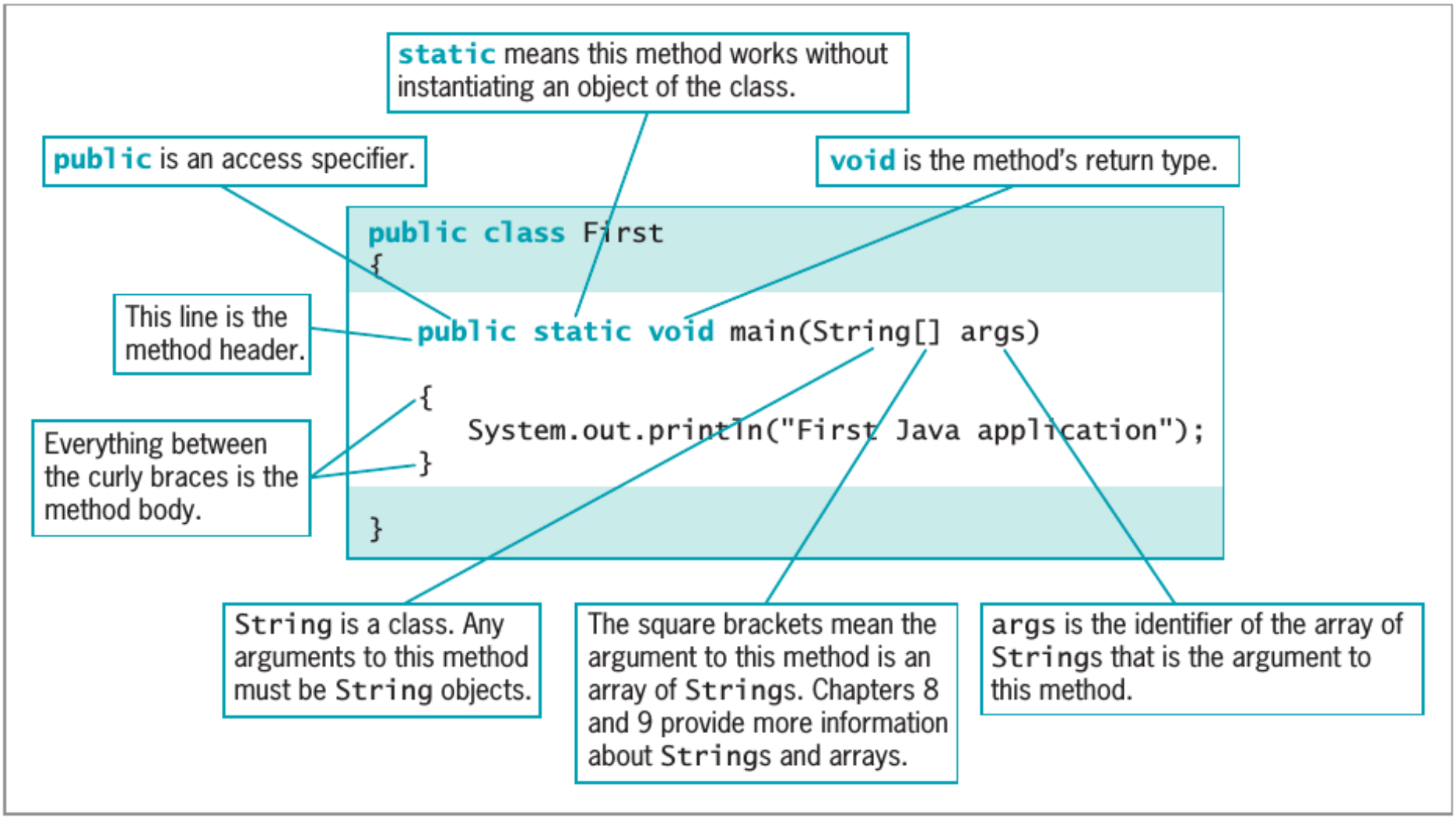
- Every program consists of one or more classes.
- **Every source file can contain at most one public class, and the name of the public class must match the name of the file containing the class**.
  - For example, the class `First` must be contained in the `First.java` file.

```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```

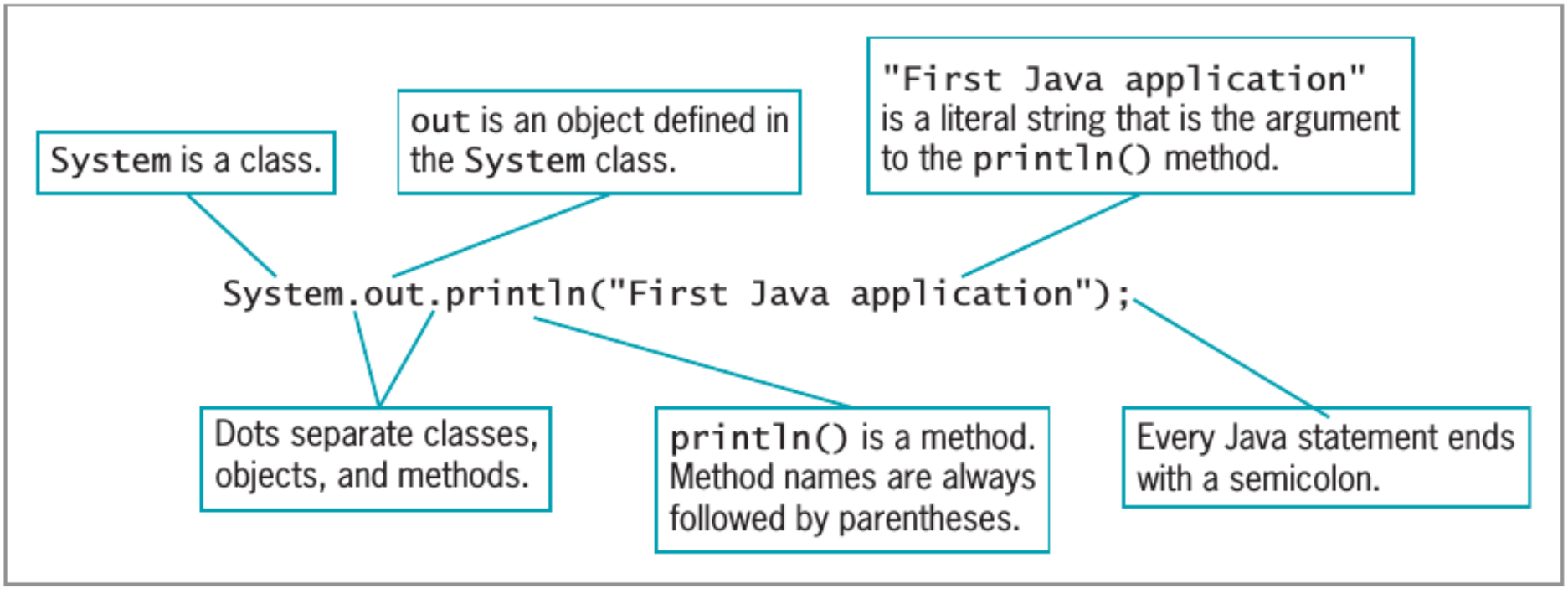
Simplest Java application (II)



Simplest Java application (III)



Simplest Java application (IV)



Compiling a Java Class

- To compile your source code from the command line, your prompt should show the folder or directory where your program file is stored.
- Then, you type `javac` followed by the name of the file that contains the source code.
- For example, to compile a file named `First.java`, you type the following and then press `Enter`:

```
> javac First.java
```

Running a Java Application

- To run the First application from the command line, you type the following:

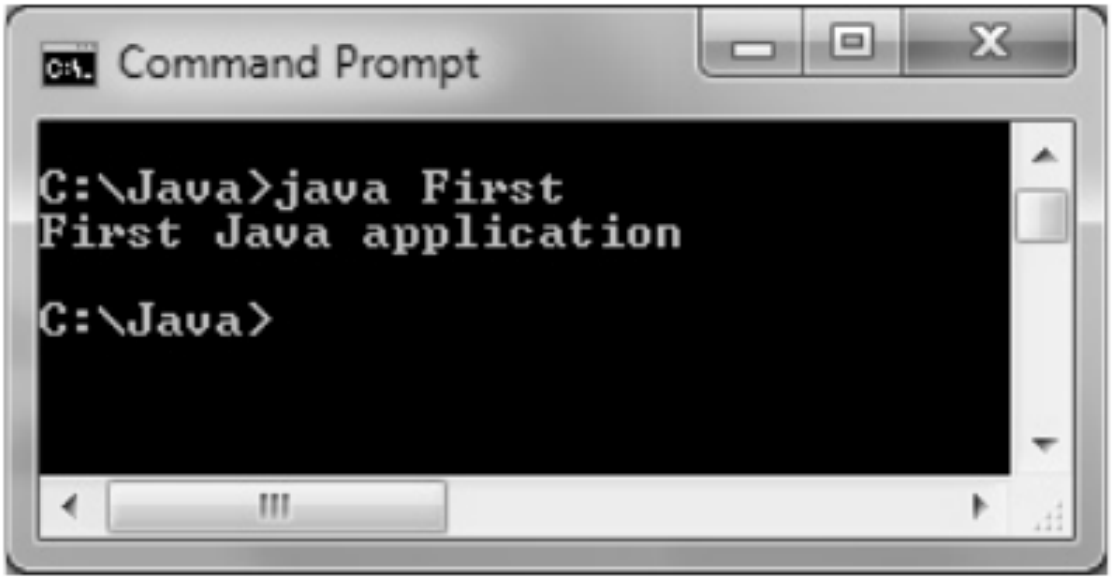
```
> java First
```

or

```
> java -cp .  First
```

or

```
> java -classpath .  First
```



Bibliography

Resources

- "Big Java: Early Objects", 6th Edition by Cay S. Horstmann
- "Java™:The Complete Reference", 7th Edition,Herbert Schildt
- "Java™Programming", 7th Edition, Joyce Farrell
- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html
- http://beginnersbook.com/java-tutorial-for-beginners-with-examples/
- https://www.leepoint.net/index.html
- https://junit.org/junit5/docs/current/user-guide/