GETTING STARTED WITH JAVA

Dr. Krishnendu Guha

Lecturer/ Assistant Professor

School of Computer Science and Information Technology

University College Cork

INTRODUCTION

Java is object-oriented programming language, developed at **Sun Microsystems**

It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more!

JAVA V/S PYTHON

- Python is **simpler** to learn.
- Python may have **less lines of code** when compared to Java.
- **Java** is **statically typed** (expects its variables to be declared before they can be assigned values), **Python** is **dynamically typed** (the declaration is not required).
- Java language uses **curly braces** to define the **beginning** and **end** of ***each function and class*** definition, whereas Python uses ***indentation to separate code*** *into separate blocks*.
- Java **does not support multi-inheritance**. This is mostly achieved by using interfaces.
- They **support different Machine Learning libraries** (Python: Tensorflow, pytorch, etc.; Java: Weka, Mallet, etc.)
- Java is **more portable** when compared to Python, because Java is available to more systems.
- Python **syntax is easier** to remember - it depends!

BENEFITS OF JAVA

- Java works on **different platforms** (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the **most popular** programming language in the world
- It has a large demand in the current **job market**
- It is **easy** to learn and simple to use
- It is **open-source** and free
- It is **secure, fast and powerful**
- Java is an object oriented language which gives a **clear structure to programs and allows code to be reused, lowering development costs**
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

FIRST JAVA PROGRAM

- In Java, every application begins with a class name, and that class must match the filename.
- Let's create our first Java file, called Main.java, which can be done in any text editor (like Notepad).
- The file should contain a "Hello World" message, which is written with the following code:

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

```
public class Exercise {
    public static void main(String[] args) {
        // Write your code here
    }
}
```

- Unlike Python, Java variables should be explicitly declared along with their type.

Python

```
movie = 'Kill Bill'
ticket_price = 60
std_discount = 0.1
total_cost = ticket_price * std_discount

print(movie, 'costs', total_cost)
```

Java

```
String movie;
int ticket_price = 60;
double std_discount;
int total_cost;

movie = "Kill Bill";
std_discount = 0.1;
total_cost = ticket_price * std_discount;

System.out.println(movie + " costs " + totalcost);
```

Java syntax for:

- declaring variables: type name;

```
int age;
```

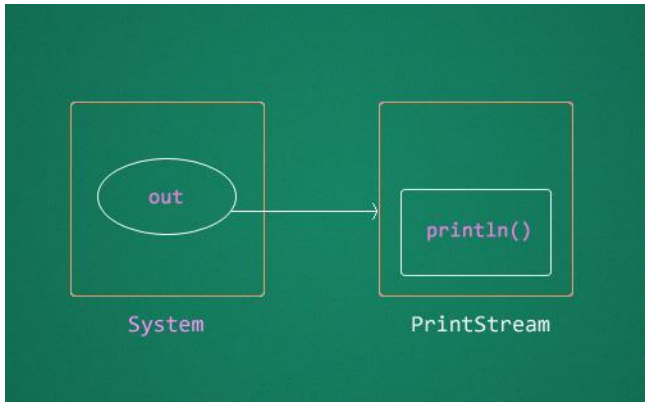
- declaring constants: final type name = value; - we'll see this later.

```
final float PI = 3.142;
```

PRINTING

There is a '**predefined' object** which knows how to '**print**' to the screen.

- Its 'name' is java.lang.System.out
- You can use its short 'name': System.out

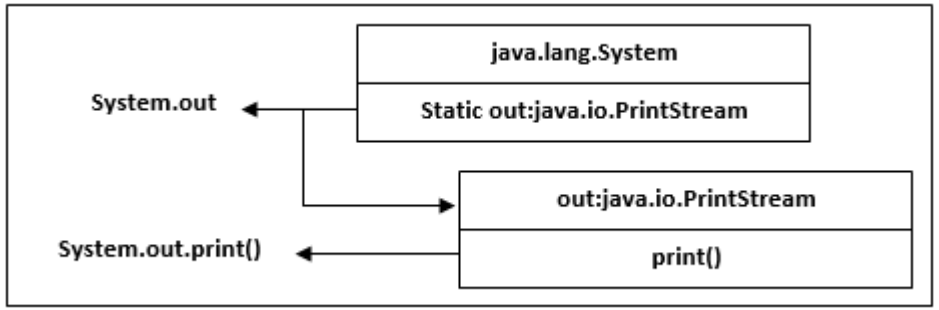


- It has **several functions (methods)** including:
- **println** for printing some data and a new line character
- **print** for printing some data (and remaining on the same line)

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

HelloWorld.java

```
1  /**
2   * HelloWorld
3   */
4  public class HelloWorld {
5
6      public static void main(String[] args) {
7          System.out.println("Hello World!");
8      }
9  }
```



READING INPUT

- There is a **'predefined' object** which knows how to **read input from the keyboard**
- Its 'name' is java.lang.System.in
- You can use its short 'name': System.in
- It has several functions (methods) including: read for reading one byte

==========

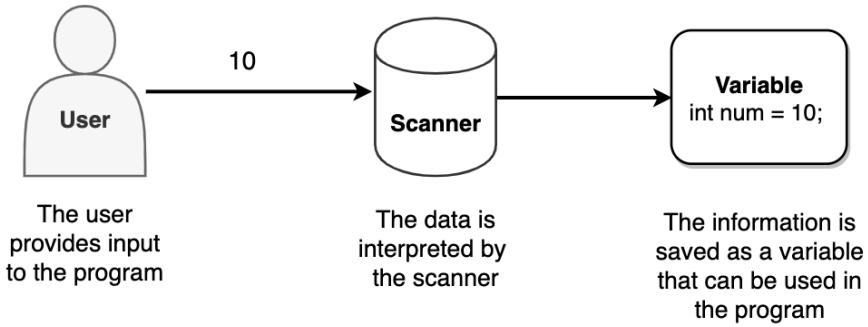
However, this one is more complicated. When there's no suitable 'predefined' object, we need to create one using new.

E.g. the following creates an object whose 'name' is sc:

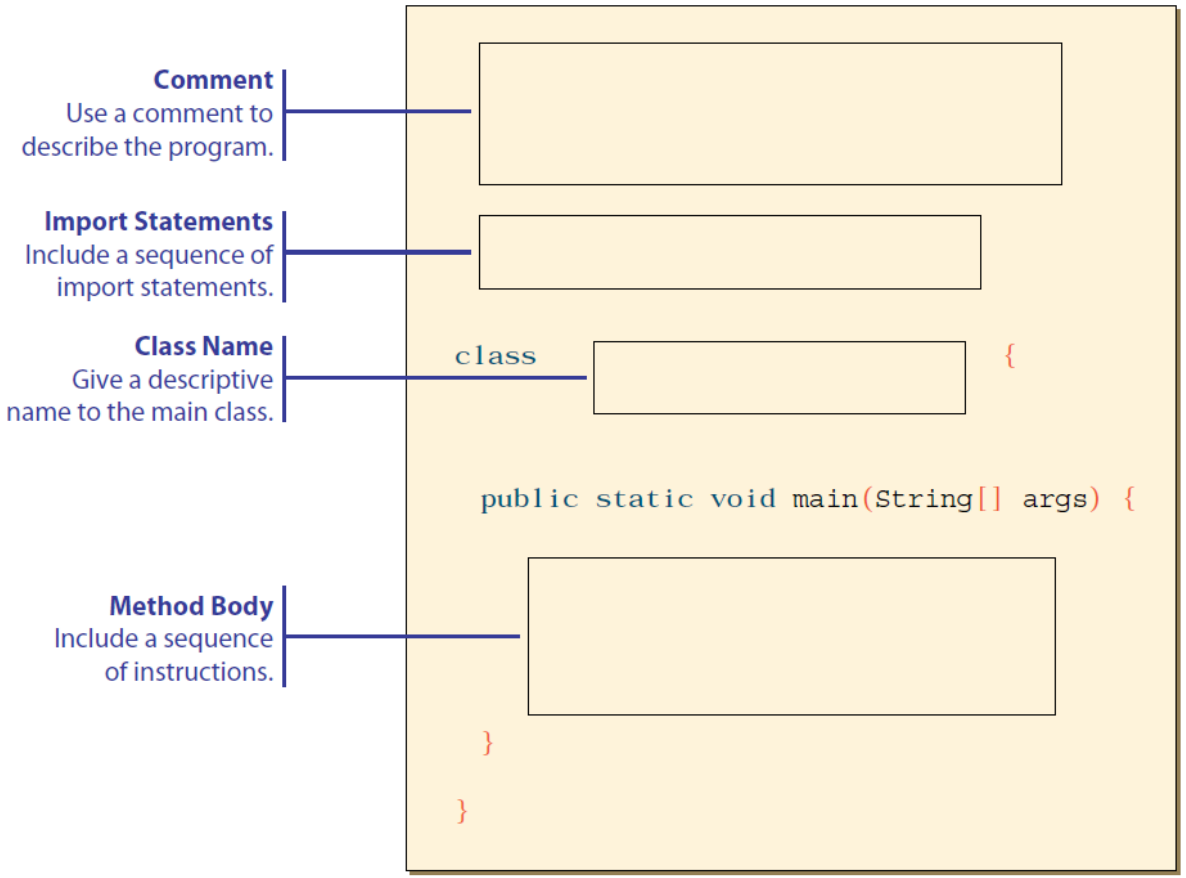
java.util.Scanner sc = new java.util.Scanner(System.in);

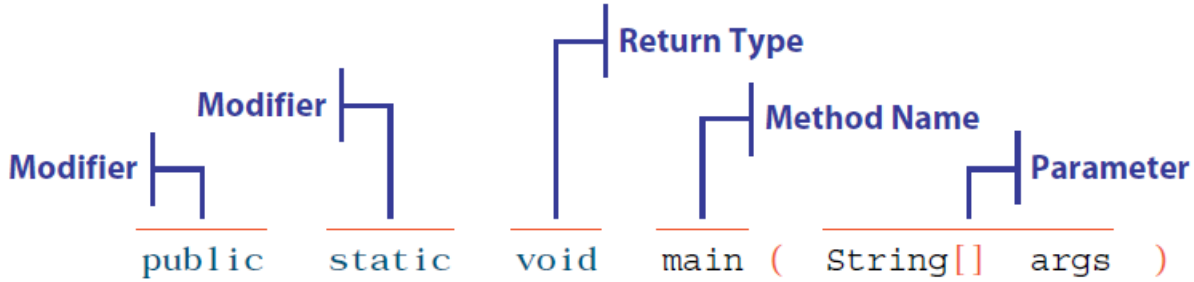
sc has several functions (methods) including:

- nextInt for reading an integer
- nextDouble for reading a floating-point number
- nextLine for reading a String



IN A NUTSHELL





Overall Flow

Editor

```
/*
   Chapter 2 Sample Program: Displaying a Window
   File: Ch2Sample.java
*/

import javax.swing.*;

class Ch2Sample1 {

    public static void main( String[] args ) {

        JFrame    myWindow;

        myWindow = new JFrame();
```

(source file)

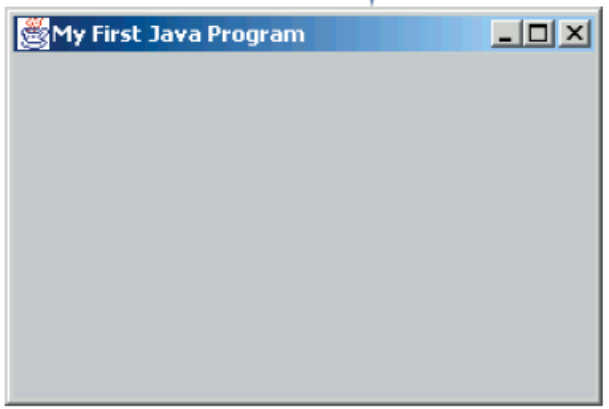
Compiler

```
be 00 03 00 2d 00 1f 08 00 12 07 00 0c
..ʃ....-........
000010  07 00 15 07 00 13 0a 00 04 00 08
0a 00 03 00 07  ................
000020  0c 00 19 00 1c 0c 00 17 00 14 01
00 04 74 68 69  ............thi
000030  73 01 00 0d 43 6f 6e 73 74 61 6e
74 56 61 6c 75  s...ConstantValu
000040  65 01 00 12 4c 6f 63 61 6c 56 61
72 69 61 62 6c  e...LocalVariabl
000050  65 54 61 62 6c 65 01 00 0e 6d 79
46 69 72 73 74  eTable...myFirst
000060  50 72 6f 67 72 61 6d 01 00 0a 45
78 63 65 70 74  Program...Except
000070  69 6f 6e 73 01 00 0f 4c 69 6e 65
4e 75 6d 62 65  ions...LineNumbe
000080  72 54 61 62 6c 65 01 00 0a 53 6f
75 72 63 65 46   rTable...SourceF
```

(bytecode file)

Interpreter

Running Program



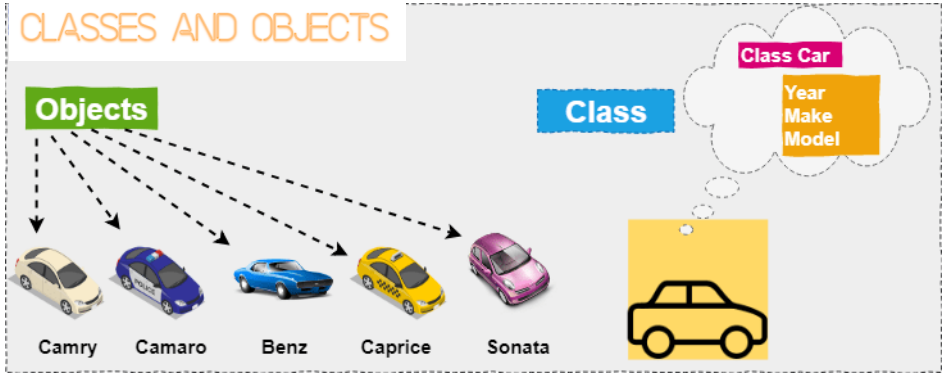
CLASSES AND OBJECTS

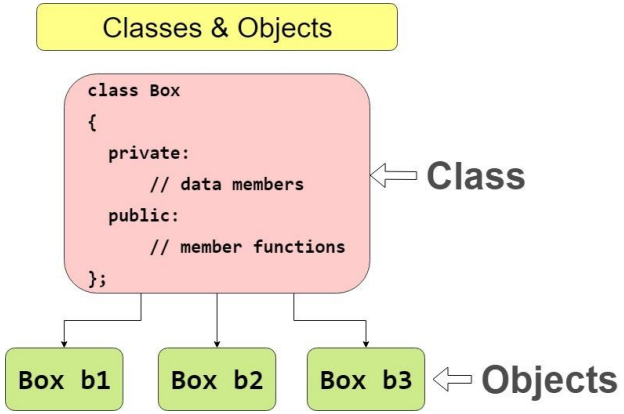
The two most important concepts in object-oriented programming are the **class** and the **object**.

In the broadest term, an ***object*** is a **thing, both tangible and intangible**, that we can imagine

A **program written in object-oriented style will consist of interacting objects**

An **object** is comprised of **data and *operations that manipulate these data***





Inside a program we write instructions to create objects

For the computer to be able **to create an object, we must provide a definition, called a *class***

A class is a kind of mold or **template** that dictates what objects can and cannot do

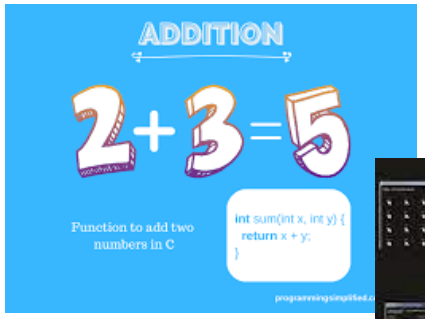
An object is called an *instance* of a class.

An object is an instance of exactly one class.
An instance of a class *belongs to* the class.
Once a class is defined, we can create as many instances of the class as a program requires.

TASKS

In writing object-oriented programs we must **first define classes**, and **while the program is running, we use the classes and objects from these classes** to accomplish **tasks**

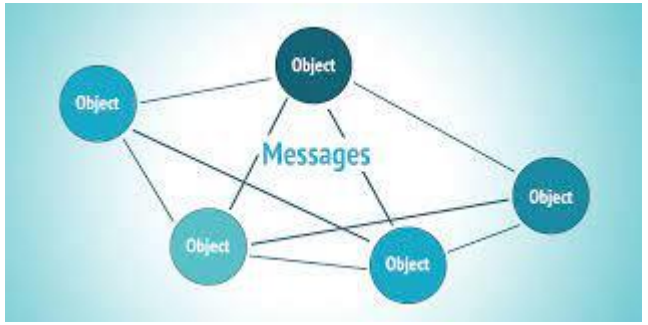
A **task** can range from adding two numbers, to managing satellites



Message

To **instruct a class or an object to perform a task**, we send a ***message*** to it

For a class or **an object to process the message**, it must be **programmed accordingly**



You cannot just send a message to any class or object.
You can send a message only to the classes and objects that understand the message you send

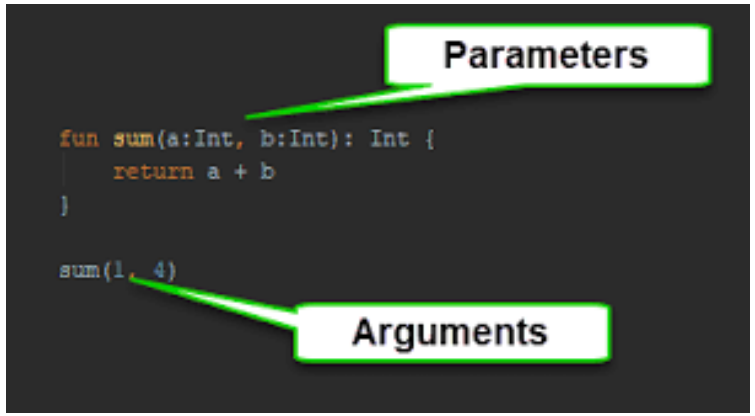
Methods and Arguments

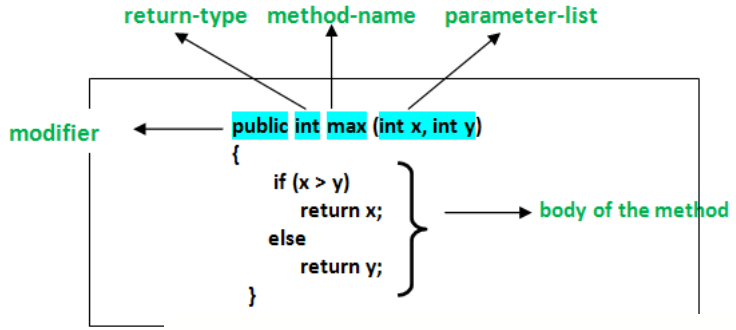
For a **class or an object to process the message** it receives, it must possess a **matching** ***method***

A Method is a **sequence of instructions that a class or an object follows to perform a task.**

A method defined for a class is called a ***class method***, and a method defined for an object is an ***instance method***.

A value we pass to an object is called an ***argument*** of a message.





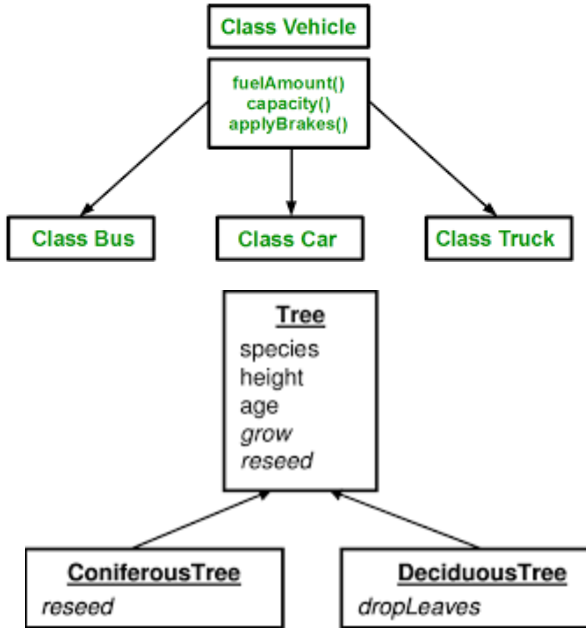

```
class ClassLesson
  def self.class_method
    "This is a class method!"
  end

  def instance_method
    "This is an instance method!"
  end
end

ClassLesson.class_method
```

The name of the message we send to an object or a class must be the same as the method's name

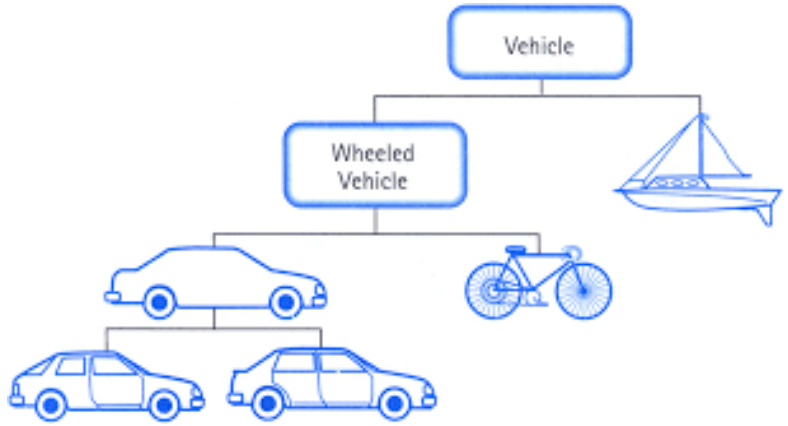
INHERITANCE

In object-oriented programming, we use a mechanism called ***inheritance*** to **design two or more entities that are different but share many common features**

First we **define a class that contains the common features of the entities**.
Then we **define classes as an extension of the common class inheriting everything from the common class**



We call the common class the ***superclass*** and all classes that inherit from it ***subclasses***.
We also call the superclass an ***ancestor*** and the subclass a ***descendant***
Other names for superclass and subclass are ***base class*** and ***derived class***, respectively



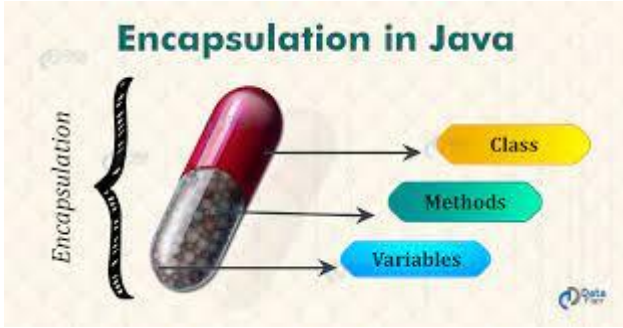
Inheritance is not limited to one level
A subclass can be a superclass of other classes, forming an inheritance hierarchy

ENCAPSULATION

- Encapsulation is the **process of hiding information** within an **object** so that **it cannot be accessed directly from outside the object.**

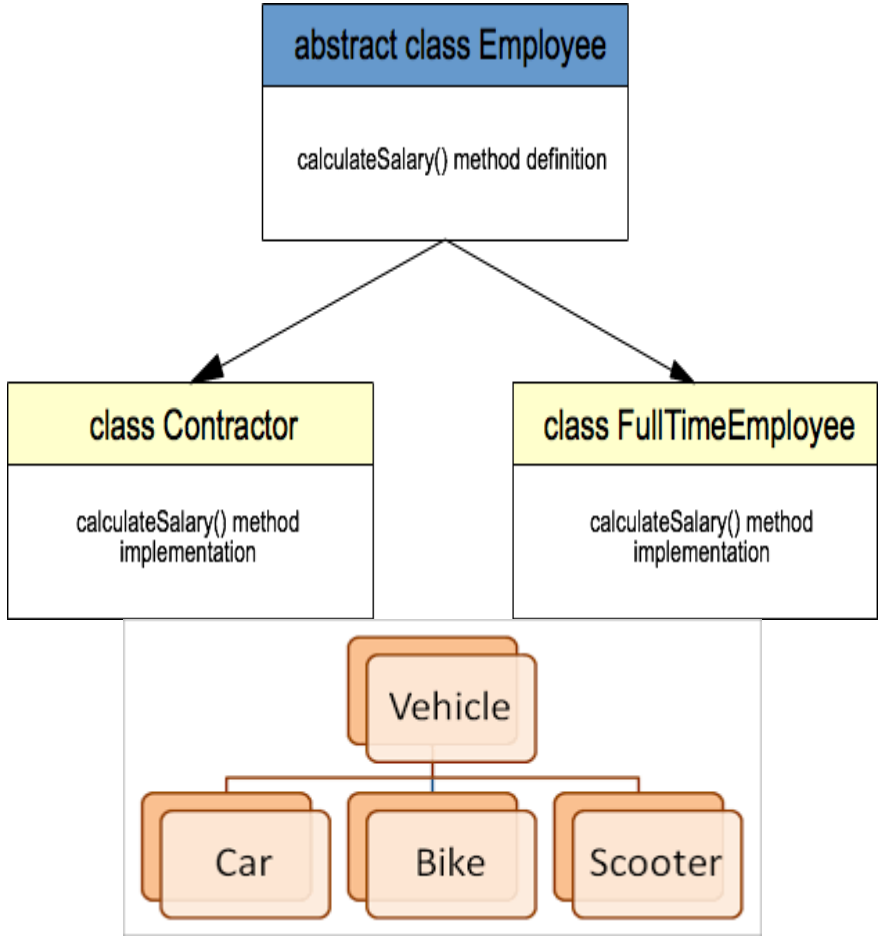
- This *allows you to control how data is used* and *prevents accidental modification of data*

- Encapsulation is important because it helps to **keep your data safe and secure**
- It also *allows you to change the implementation of your code without affecting the rest of your codebase.*



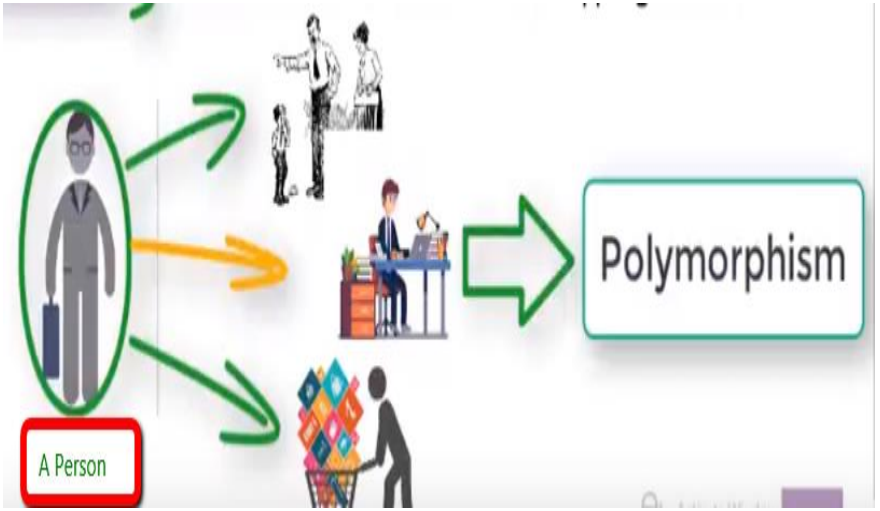
ABSTRACTION

- Abstraction is the **process of hiding the implementation details of an object** so that ***it can be used without understanding how it works***
- This *allows you to create code that is easy to use and maintain*
- Abstraction is important because it allows you to *create code that is easy to use and understand*
- Abstraction allows the user to *use your code without needing to know the details of how it works*



POLYMORPHISM

- Polymorphism is the **ability of an object to take on multiple forms**
- This is useful because it *allows you to create code that is more flexible and adaptable*
- Polymorphism allows you to *write code that can be used with multiple types of objects*




ANY
QUESTIONS


Thanks!