Tutorial 4

- Class declaration
- Package & Module
- Composition
- Inheritance
- Interface
- Access Modifiers

Class declaration

```java
public class A_class {

    /*Here I declare my class attributes
        Attribute vs Variables
             - Attribute ->access  into class
             - Variable -> access into specific method
    */
    private String name;

    public A_class() {
        System.out.println("HELLO i am a constructor ");
        //I know this is a constructor because has the same name with class
    }

    public void setName(String name){
        this.name=name;
        //this indicates -> in current object, attribute
            //==current_object.name
    }
    public String getName(){
        return this.name;
    }
}
```

Where I run main ?

Compiler is searching for `main` method

Preferable :

to have a Main.java with the psvm function

** psvm == public static void main()

Tip: You can also include other methods there

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

Can i class have as attribute another class?

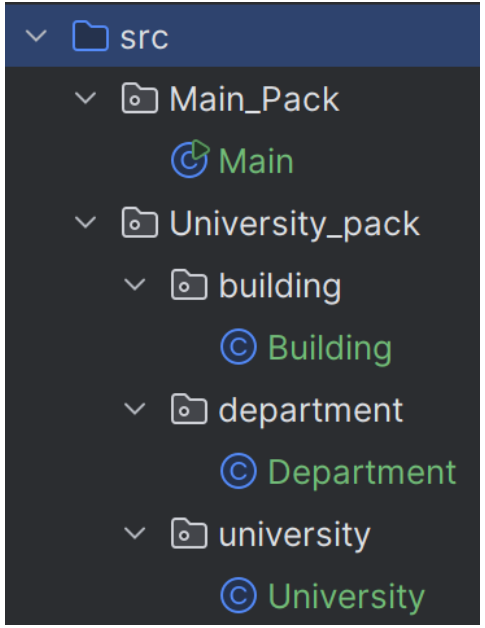
Called : Composition

```java
class Engine {
    private String type;
    private int horsepower;
}

class Wheels {
    private int size;
    private String type;
}

class Body {
    private String color;
    private String style; }

class Car {
    private Engine engine;
    private Wheels wheels;
    private Body body;

    public Car(Engine engine, Wheels wheels, Body body) {
        this.engine = engine;
        this.wheels = wheels;
        this.body = body;
    }

}
```

Packages and modules

if I have this structure and this beginning of a .java:

where am I ?




```java
package Main_Pack; // where am I ?

import University_pack.building.Building; // which class I need?
import University_pack.department.Department; // which class I need?
import University_pack.university.University; // which class I need?
```

Example of Composition

University , Department and Buildings

```java
public class Main {
    public static void main(String[] args) {
        /*
        This University (with name 'University of ECE')
            has
              -> only a department (with name 'Electrical and Computer Engineering')

         This department has 3 building with names :
             1)  Labs_Building
             2)  Teaching_Building_1
             3)  Teaching_Building_2

        */
```

Example of Composition

University , Department and Buildings

```java
package Main_Pack;
// making it accessible within the class.

import University_pack.building.Building;
import University_pack.department.Department;
import University_pack.university.University;

public class Main {
    public static void main(String[] args) {
        University university1 = new University("University of ECE");

        Department ece = new Department("Electrical and Computer Engineering");
        university1.addDepartment(ece);

        Building labs = new Building("Labs_Building");
        Building teaching_building_1 = new Building("Teaching_Building_1");
        Building teaching_building_2 = new Building("Teaching_Building_2");

        ece.addBuildings(labs);
        ece.addBuildings(teaching_building_1);
        ece.addBuildings(teaching_building_2);

        System.out.println("Buildings in the " + ece.getName() + " department:");
        for(Building building : ece.buildings){
            System.out.println(building.getName());
        }
    }
}
```

Example of Composition

University , Department and Buildings

```java
package University_pack.department;

import University_pack.building.Building;
import java.util.ArrayList;

public class Department {
    private String name;
    public ArrayList<Building> buildings;

    public Department(String name) {
        this.name = name;
        this.buildings = new ArrayList<>();
    }

    public String getName() {
        return name;
    }
    public void addBuildings(Building building) {
        this.buildings.add(building);
    }
}
```

```java
package University_pack.university;

import University_pack.department.Department;
import java.util.ArrayList;

public class University {
    private String name;
    public ArrayList<Department> departments;

    public University(String name) {
        this.name = name;
        this.departments = new ArrayList<>();
    }

    public void addDepartment(Department department) {
        this.departments.add(department);
    }
}
```

```java
package University_pack.building;

public class Building {
    private String name;

    public Building(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

Inheritance

```java
// Parent class
class Animal {
    String name;

    // Constructor
    Animal(String name) {
        this.name = name;
    }

    // Method to display the animal sound
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
```

```java
// Child class
class Dog extends Animal {

    // Constructor
    Dog(String name) {
        super(name); // Call the parent class constructor
    }

    // Overriding the makeSound method
    @Override
    void makeSound() {
        super.makeSound(); // Call the parent class's makeSound method
        System.out.println(name + " says: Woof woof!");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy");
        myDog.makeSound();
    }
}
```

```
//output
Animal makes a sound
Buddy says: Woof woof!
```

Interface

```java
// Define the interface
interface Shape {
    double calculateArea();
    double calculatePerimeter();
}
```

```java
// Implement the interface
class Circle implements Shape {
    private double radius;

    // Constructor
    public Circle(double radius) {
        this.radius = radius;
    }

    // Implement calculateArea method
    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }

    // Implement calculatePerimeter method
    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }

    // Main method to test the Circle class
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        System.out.println("Area of Circle: " + circle.calculateArea());
        System.out.println("Perimeter of Circle: " + circle.calculatePerimeter());
    }
}
```

Access Modifiers

```java
// Class to demonstrate access modifiers
public class AccessModifiersDemo {

    // Default access - no modifier keyword
    String defaultMessage = "This is the default message (package-private).";

    // Private access - accessible only within this class
    private String privateMessage = "This is a private message.";

    // Protected access - accessible within package and subclasses
    protected String protectedMessage = "This is a protected message.";

    // Public access - accessible from anywhere
    public String publicMessage = "This is a public message.";

    // Method to print all messages
    public void showMessages() {
        System.out.println(defaultMessage);
        System.out.println(privateMessage);
        System.out.println(protectedMessage);
        System.out.println(publicMessage);
    }
}
```

```java
// Another class within the same package
class TestAccess {
    public static void main(String[] args) {
        AccessModifiersDemo demo = new AccessModifiersDemo();

        // Accessing fields with different access modifiers
        System.out.println("Accessing messages within the same package:");
        System.out.println(demo.defaultMessage);  // Accessible
        // System.out.println(demo.privateMessage); // Not accessible - would
cause error
        System.out.println(demo.protectedMessage); // Accessible
        System.out.println(demo.publicMessage);    // Accessible

        // Accessing through public method
        System.out.println("\nAccessing all messages through public method:");
        demo.showMessages();
    }
}
```

Task

- Go back to Tutorial_3 and have a look at the two examples
  - make the Tutorial's 3 task
- Study 4 Principles of Java
  - Abstraction: https://www.geeksforgeeks.org/abstraction-in-java-2/
  - Encapsulation: https://www.geeksforgeeks.org/encapsulation-in-java/
  - Inheritance: https://www.geeksforgeeks.org/inheritance-in-java/
  - Polymorphism: https://www.geeksforgeeks.org/polymorphism-in-java/
- Write a program to interact with 4 Principles of Java