Bahria University, Islamabad

Department of Software Engineering

Software Design & Architecture Lab

(Spring-2025)

Teacher: Engr. AAMIR SOHAIL

Student Name: Majid Rahman

Enrollment No: 01-131232-041

Lab Journal: 07

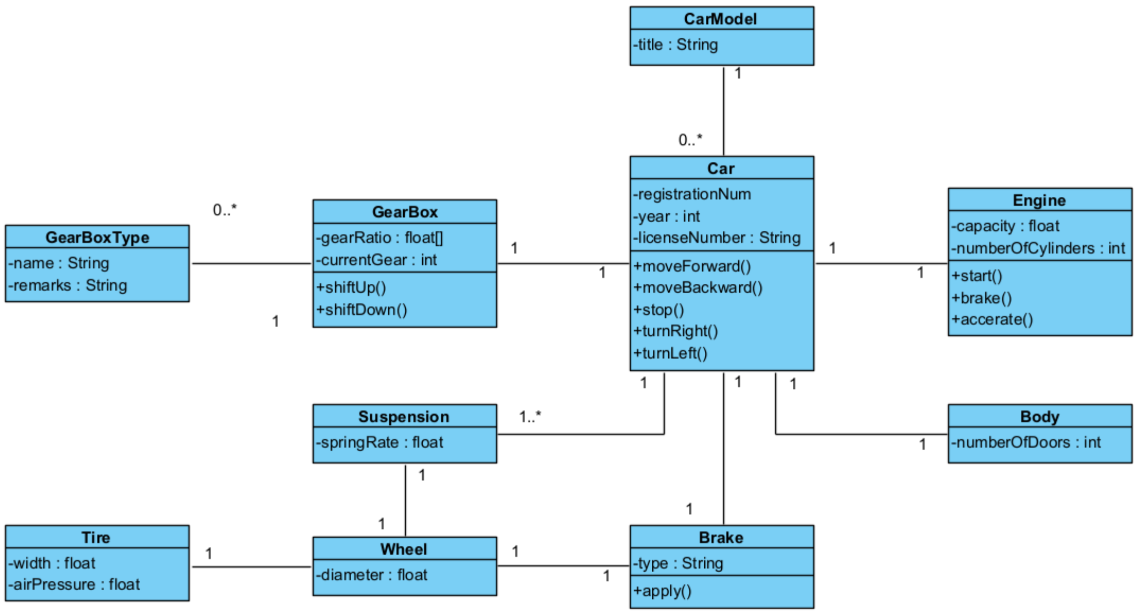
Date: 23/03/2025

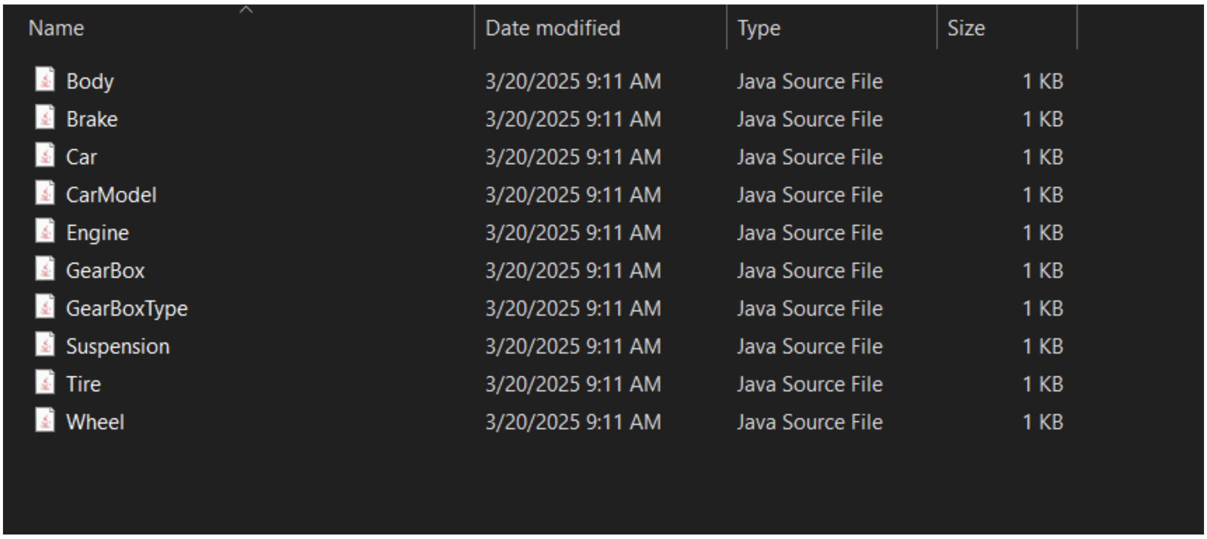
Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	3		5		
2	3				
3	3				
4	3				
5	3				

Comments:

Signature

TASK1:





Name	Date modified	Type	Size
Body	3/20/2025 9:11 AM	Java Source File	1 KB
Brake	3/20/2025 9:11 AM	Java Source File	1 KB
Car	3/20/2025 9:11 AM	Java Source File	1 KB
CarModel	3/20/2025 9:11 AM	Java Source File	1 KB
Engine	3/20/2025 9:11 AM	Java Source File	1 KB
GearBox	3/20/2025 9:11 AM	Java Source File	1 KB
GearBoxType	3/20/2025 9:11 AM	Java Source File	1 KB
Suspension	3/20/2025 9:11 AM	Java Source File	1 KB
Tire	3/20/2025 9:11 AM	Java Source File	1 KB
Wheel	3/20/2025 9:11 AM	Java Source File	1 KB

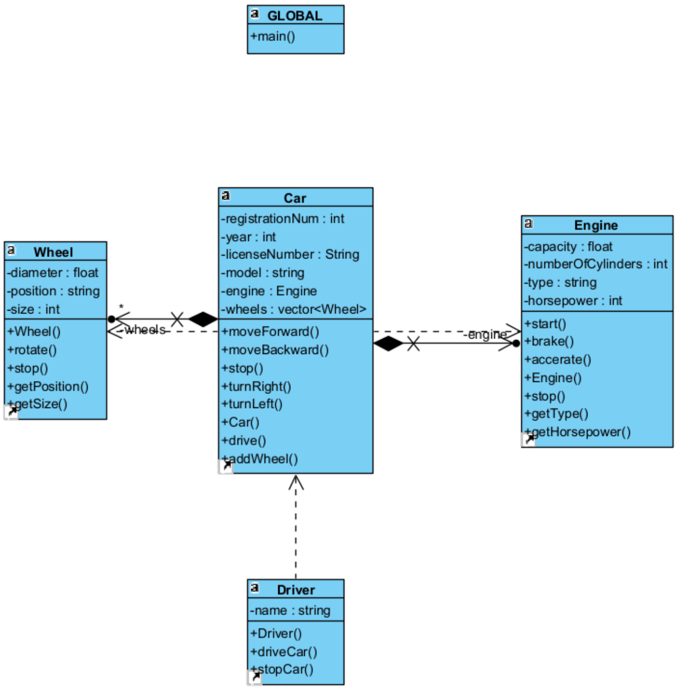
TASK # 2

```
#include <iostream>

#include <string>

#include <vector>  // Include this to use std::vector


// Class for the Engine

class Engine {

public:

    Engine(std::string type, int horsepower)

        : type(type), horsepower(horsepower) {

    }


    void start() {

        std::cout << "The " << type << " engine with " << horsepower << " horsepower is now running.\n";

    }


    void stop() {

        std::cout << "The " << type << " engine has been stopped.\n";

    }


    std::string getType() const {

        return type;

    }


    int getHorsepower() const {

        return horsepower;

    }


private:

    std::string type;

    int horsepower;
```

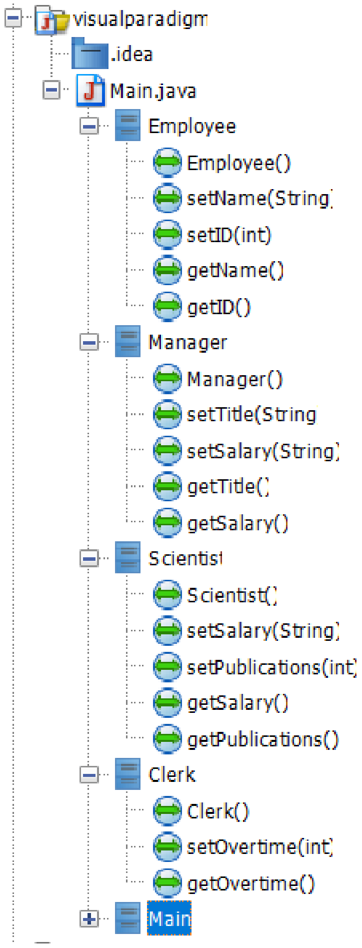
```
};

// Class for the Wheel
class Wheel {
public:
    Wheel(std::string position, int size)
        : position(position), size(size) {
    }

    void rotate() {
        std::cout << "The " << position << " wheel is rotating with size " << size << " inches.\n";
    }

    void stop() {
        std::cout << "The " << position << " wheel has stopped rotating.\n";
    }

    std::string getPosition() const {
        return position;
    }

    int getSize() const {
        return size;
    }

private:
    std::string position;  // Position: Front Left, Front Right, Rear Left, Rear Right
    int size;              // Size in inches
};

// Class for the Car
class Car {
public:
    Car(std::string model, Engine engine)
        : model(model), engine(engine) {
    }

    void drive() {
        std::cout << "Driving the " << model << " car.\n";
        engine.start();
        for (Wheel& wheel : wheels) {
            wheel.rotate();
        }
    }

    void stop() {
        std::cout << "Stopping the " << model << " car.\n";
        engine.stop();
        for (Wheel& wheel : wheels) {
            wheel.stop();
        }
    }

    void addWheel(Wheel wheel) {
        if (wheels.size() < 4) {
            wheels.push_back(wheel);
        }
        else {
            std::cout << "Cannot add more wheels. The car already has 4 wheels.\n";
        }
    }

private:
    std::string model;
    Engine engine;
    std::vector<Wheel> wheels;  // Declare wheels as a vector of Wheel objects
```

```
};

// Class for the Driver
class Driver {
public:
    Driver(std::string name) : name(name) {}

    void driveCar(Car& car) {
        std::cout << name << " is getting into the car.\n";
        car.drive();
    }

    void stopCar(Car& car) {
        std::cout << name << " is stopping the car.\n";
        car.stop();
    }

private:
    std::string name;
};

// Main function to demonstrate the classes
int main() {
    // Creating an Engine
    Engine myEngine("V8", 450);

    // Creating Wheels
    Wheel frontLeft("Front Left", 18);
    Wheel frontRight("Front Right", 18);
    Wheel rearLeft("Rear Left", 18);
    Wheel rearRight("Rear Right", 18);

    // Creating a Car
```

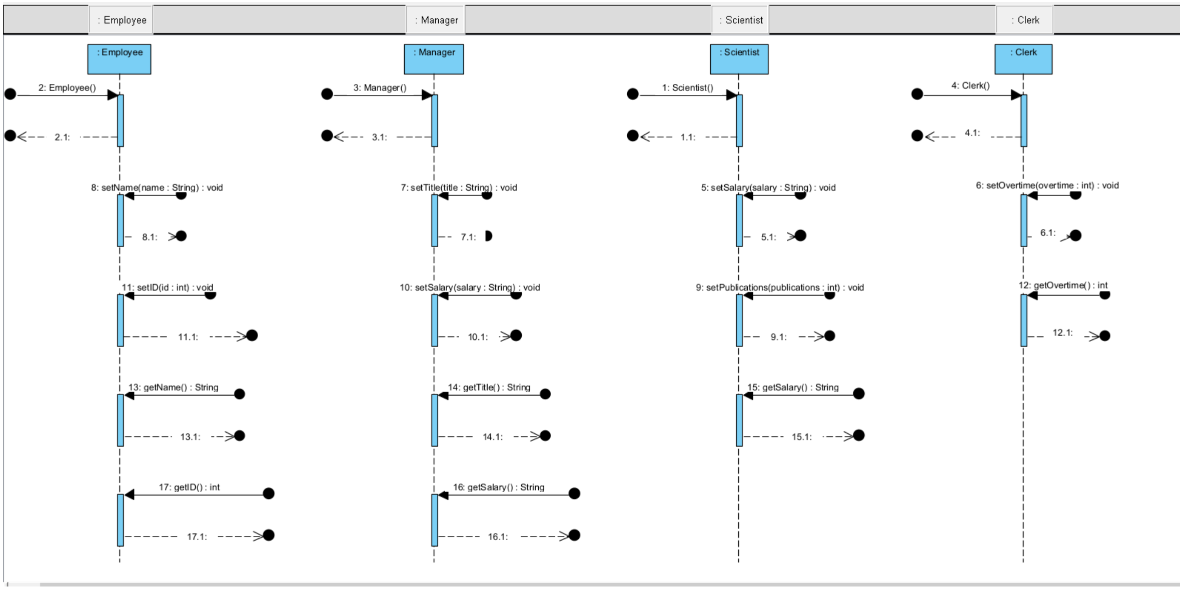
```
Car myCar("Mustang", myEngine);

// Adding wheels to the car
myCar.addWheel(frontLeft);
myCar.addWheel(frontRight);
myCar.addWheel(rearLeft);
myCar.addWheel(rearRight);

// Creating a Driver
Driver myDriver("John");

// Driver driving the car
myDriver.driveCar(myCar);
myDriver.stopCar(myCar);

return 0;
 }
```



Task 3:



```
visualparadigm
  .idea
  Main.java
    Employee
      Employee()
      setName(String)
      setID(int)
      getName()
      getID()
    Manager
      Manager()
      setTitle(String
      setSalary(String)
      getTitle()
      getSalary()
    Scientist
      Scientist()
      setSalary(String)
      setPublications(int)
      getSalary()
      getPublications()
    Clerk
      Clerk()
      setOvertime(int)
      getOvertime()
    Main
```


: Employee
: Manager
: Scientist
: Clerk
: Employee
: Manager
: Scientist
: Clerk
2: Employee()
3: Manager()
1: Scientist()
4: Clerk()
2.1:
3.1:
1.1:
4.1:
8: setName(name : String) : void
7: setTitle(title : String) : void
5: setSalary(salary : String) : void
6: setOvertime(overtime : int) : void
8.1:
7.1:
5.1:
6.1:
11: setID(id : int) : void
10: setSalary(salary : String) : void
9: setPublications(publications : int) : void
12: getOvertime() : int
11.1:
10.1:
9.1:
12.1:
13: getName() : String
14: getTitle() : String
15: getSalary() : String
13.1:
14.1:
15.1:
17: getID() : int
16: getSalary() : String
17.1:
16.1: