# Object-Oriented Programming Assignment 1 (Car configuration)

**Output:**

**Text

Description automatically generated**

# Test Car Class

Using public static void main statement print out the sample code given. In this class we use the new keyword to instantiate the class by allocating desired memory for an associated new object. We also set our two fuel amounts using the setFuel() method.

public class TestCar

{

public static void main(String[] args) {

Car car = new Car("AUDI R8");

Engine engine = new Engine("V8", 43);

car.add(engine);

Wheel wheel = new Wheel("Michelin15", 15);

car.add(wheel);

car.setFuelVol(100);

System.out.printf("Current fuel: %.2f\n",car.getFuelVol());

car.drive();

car.printState();

car.setFuelVol(50);

System.out.printf("Current fuel: %.2f\n",car.getFuelVol());

car.drive();

car.printState();

}

}

# Car class

My constructors will initialize the ‘new’ objects. Add object will tie the engine and the wheel objects to car.

Our get method (getFuel) gets the fuel input. The other methods calibrate the fuel level (setFuel) and the drive method sets distance and total km then sets fuel to 0;

We also have a method print statement which prints the output.

public class Car

{

private String carName;

private double distanceTravel;

private double totalKilo;

private double fuelVol;

private Engine engine;

private Wheel wheel;

public Car(String carName)

{

// initialise instance variables

this.carName = carName;

}

public void add(Engine engine){

// add engine

this.engine = engine;

}

public void add(Wheel wheel){

// add wheel

this.wheel = wheel;

}

public void setFuelVol(double fuelVol){

// set fuel level

this.fuelVol = fuelVol;

}

public double getFuelVol(){

// return fuel level

return fuelVol;

}

public void drive() {

double distanceTravel = engine.totalDistanceTravel(wheel, fuelVol);

this.distanceTravel = distanceTravel;

totalKilo += distanceTravel;

fuelVol = 0;

}

public void printState(){

System.out.printf("Configuration: Car Body %s\n", carName);

System.out.printf("Engine name: %s\n",engine.getName());

System.out.printf("Engine turns per litre: %.1f\n", engine.getTurnsPl());

System.out.printf("Engine's turns count: %.1f\n", engine.getTotalNoTurns());

System.out.printf("Wheel name: %s\n", wheel.getName());

System.out.printf("Wheel radius: %.2f cm\n", wheel.getRad());

System.out.printf("Wheel circumference (distance per turn): %.2f cm\n",

wheel.turn());

System.out.printf("Distance this trip: %.2f\n", distanceTravel);

System.out.printf("Total distance travelled: %.2f\n", totalKilo);

System.out.printf("Current fuel status: %.2f liters\n\n", fuelVol);

}

}

# Engine

In this class we are calculating turnsPl and total number of turns using the radius and circumference formulas.

public class Engine

{

private String name;

private double turnsPl;

private double totalNoTurns;

public Engine(String name, double turnsPl)

{

this.name = name;

this.turnsPl = turnsPl;

}

public String getName(){

// engine name

return name;

}

public double getTurnsPl(){

// turns per litre

return turnsPl;

}

public double getTotalNoTurns(){

// engine turns

return totalNoTurns;

}

public double totalDistanceTravel(Wheel wheel, double fuelVol){

// formula to calculate number of turns

double noTurns = fuelVol \* turnsPl;

totalNoTurns += noTurns;

return noTurns \* wheel.turn();

}

}

# Wheel Class

In this class we set up the instance variables for the wheel name, radius and circumference in order to be able to calculate turns in the engine class making sure to return the values.

public class Wheel

{

private String name;

private double rad;

private double circ;

public Wheel(String name, double rad)

{

this.name = name;

this.rad = rad;

circ = (2 \* Math.PI) \* rad;

}

public double turn(){

// cicumference of wheel

return circ;

}

public double getRad(){

// radius of wheel

return rad;

}

public String getName(){

// name of engine

return name;

}

}

Other example to test accuracy:

Text

Description automatically generated