JAVA - J2EE Batch 2

Name – Aman Yadav

E-mail : prakashaman5@gmail.com

Phone: +919519131321

Assignment-4

3 Problem Statement - Implementing Polymorphism using Abstract class and interface

Define Vehicle interface, AbstractManufacturer abstract class,Car class,Bike class and VehicleService Class as given below:

Interface Vehicle

create the following abstract method.

```
+maxSpeed(String type) : int
```

Abstract class AbstractManufacturer

Declare the following private properties.

```
-name : String
-modelName : String
-type : String
```

- Provide getter for all properties

Declare abstract method

```
+getManufacturerInformation() : String
```

Car Class

- Make the class as subclass of AbstractManufacturer and implements Vehicle interface.
- Define parameterized constructor passing three parameters to initialize name,modelName and type.
- Override the abstract methods and follow the instructions given as comments for the business logic.

Bike Class

- Make the class as subclass of AbstractManufacturer and implements Vehicle interface.

- Define parameterized constructor passing three parameters to initialize name,modelName and type.
- Override the abstract methods and follow the instructions given as comments for the business logic.

VehicleService Class has the following three methods

```
        +createCar(String,String,String) : Car
        +createBike(String,String,String) : Bike
        +compareMaxSpeed(Vehicle,Vehicle) : int
Follow the comments to complete the business logic for all three methods
```

4 PROGRAM

```
Copy the program into Codelabs/Any of the IDE, complete the instructions as
per problem statement

/*
Class is having 3 fields name, modelName and type.
Type varies for different vehicles.
eg. Car is of type sedan, sports...
Bike is of type cruiser, sports...
 */
public abstract class AbstractManufacturer {
    public String getModelName() {
        return null;
    }

    public String getType() {
        return null;
    }

    public String getName() {
        return null;
    }

    public abstract String getManufacturerInformation();
}


public class Bike extends AbstractManufacturer implements Vehicle {
    public Bike(String name, String modelName, String bikeType) {
    }

    /*
   Method returns maximum speed depending upon their types
   For Sports-300kmh
   For cruiser-170kmh
    */
    @Override
    public int maxSpeed(String bikeType) {
        return 0;
    }

    /*
    should return in the format : Bike{Manufacturer name:'name',Model
Name:'modelName',Type:'type'}
```

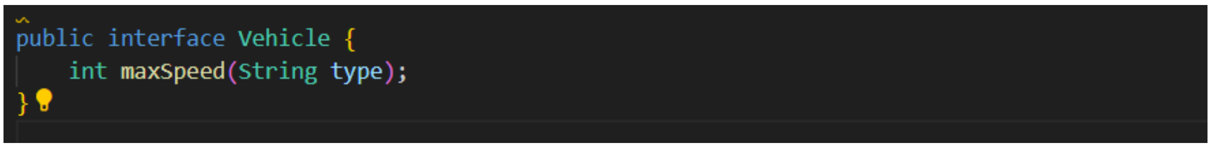
```
     */
    @Override
    public String getManufacturerInformation() {
        return null;
    }
}
public class Car extends AbstractManufacturer implements Vehicle {
    public Car(String name, String modelName, String carType) {
    }

    /*
    Method returns maximum speed depending upon their types
    For sports-250kmh
    For sedan-190kmh
     */
    @Override
    public int maxSpeed(String carType) {
        return 0;
    }

    /*
    should return in the format : Car{Manufacturer name:'name',Model
Name:'modelName',Type:'type'}
     */
    @Override
    public String getManufacturerInformation() {
        return null;
    }
}

public interface Vehicle {
    int maxSpeed(String type);
}


public class VehicleService {
    /*
    create a Car object and return it
     */
    public Car createCar(String name, String modelName, String type) {
        return null;
    }

    /*
    create a bike object and return it
     */
    public Bike createBike(String name, String modelName, String type) {
        return null;
    }

    /*
    Method should compare the speed only if the vehicle is of "SPORTS"
type.
    Method should return 0 when speeds are equal otherwise should return
maximum vehicle speed.
    Method returns -1 if the type is not "SPORTS"
    */
    public int compareMaxSpeed(Vehicle first, Vehicle second) {
        /*
        Vehicle objects should be downcasted to their respective concrete
types
```

```
            */
            return 0;
        }
    }
```

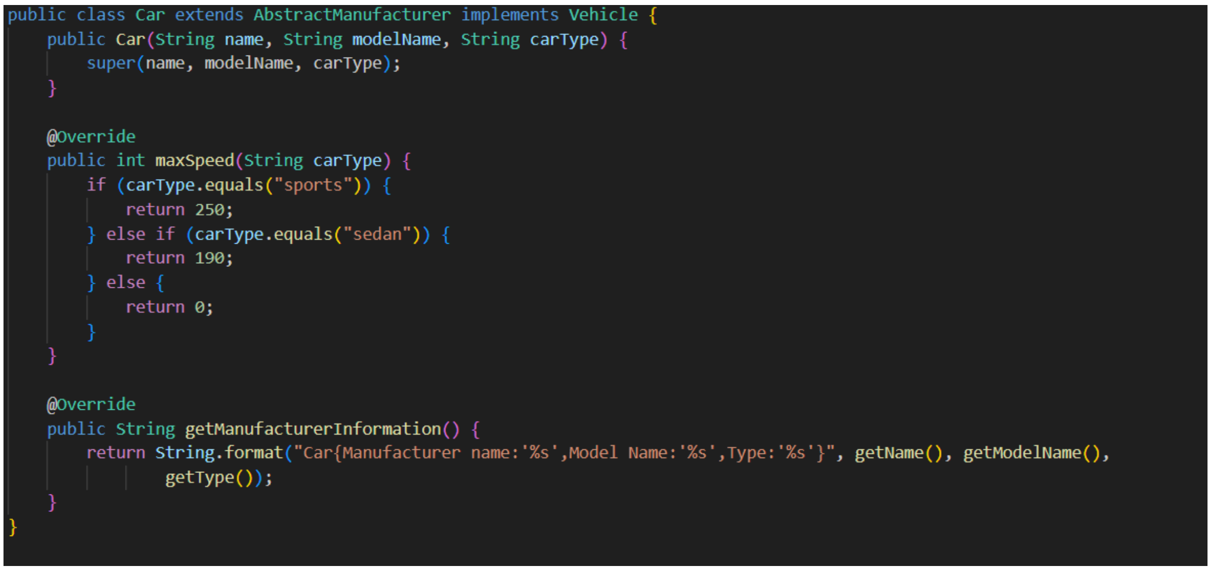
4.1 Instructions

- Avoid printing unnecessary values other than expected output as given in sample
- Take care of whitespace/trailing whitespace
- Do not change the provided class/method names unless instructed
- Follow best practices while coding

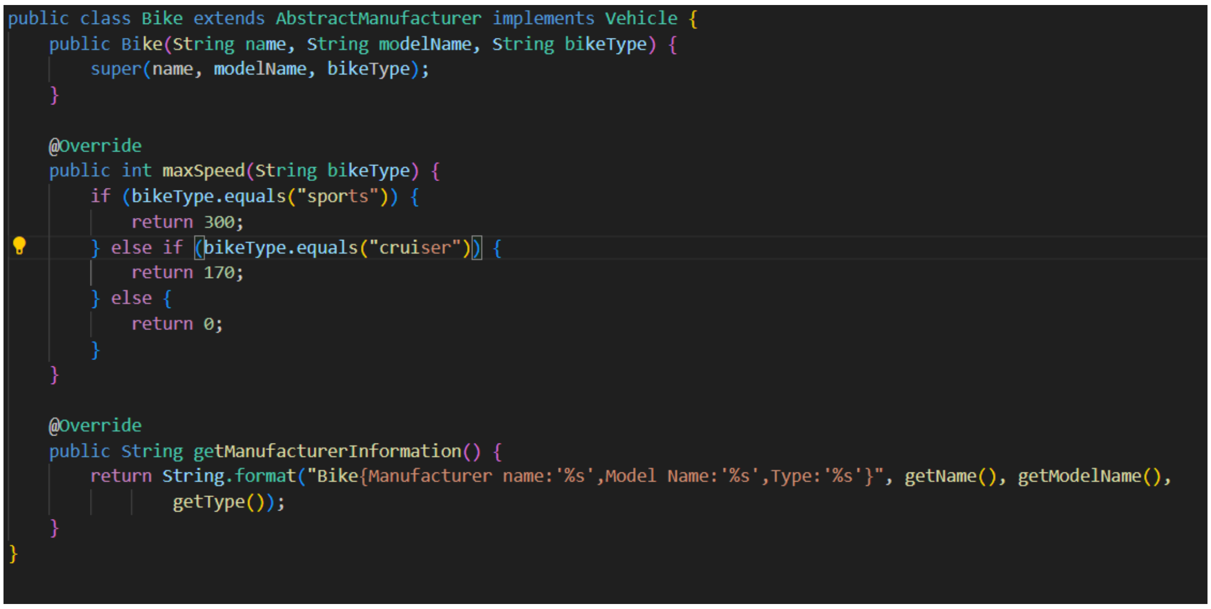
→



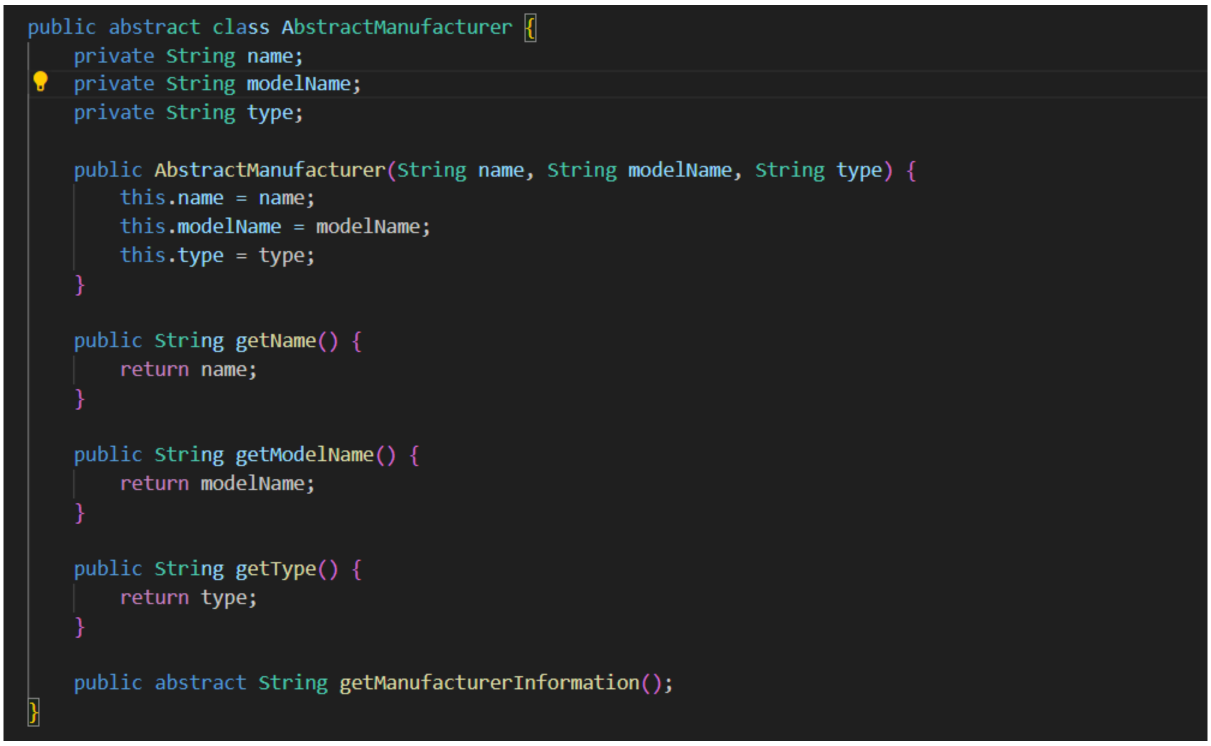
```java
public interface Vehicle {
    int maxSpeed(String type);
}
```



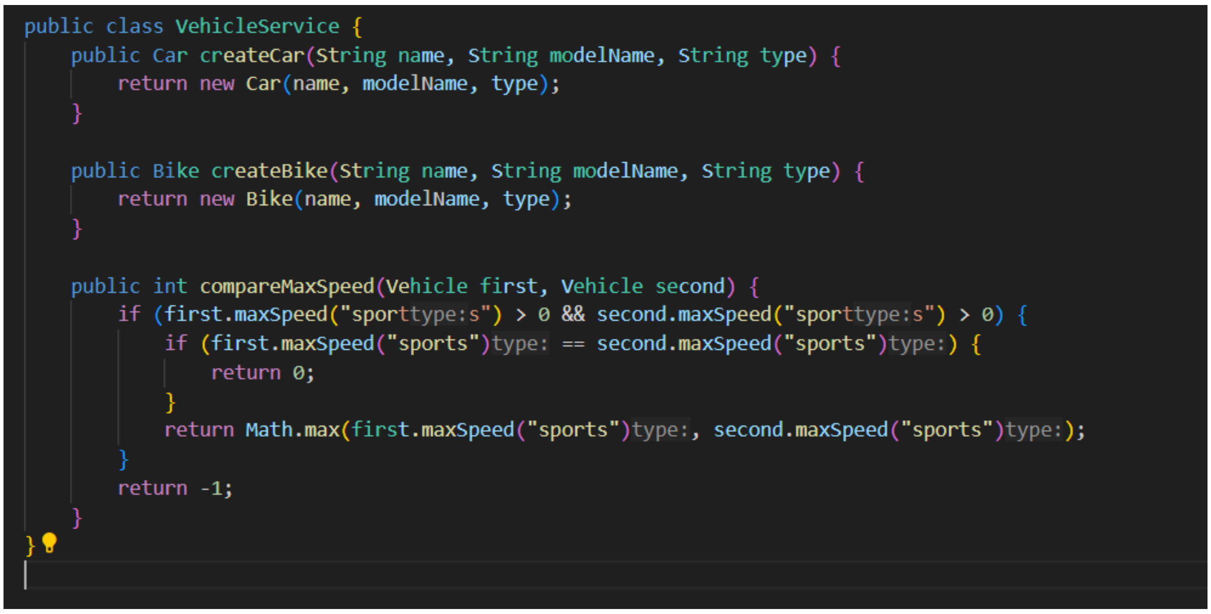
```java
public class Car extends AbstractManufacturer implements Vehicle {
    public Car(String name, String modelName, String carType) {
        super(name, modelName, carType);
    }

    @Override
    public int maxSpeed(String carType) {
        if (carType.equals("sports")) {
            return 250;
        } else if (carType.equals("sedan")) {
            return 190;
        } else {
            return 0;
        }
    }

    @Override
    public String getManufacturerInformation() {
        return String.format("Car{Manufacturer name:'%s',Model Name:'%s',Type:'%s'}", getName(), getModelName(),
                getType());
    }
}
```



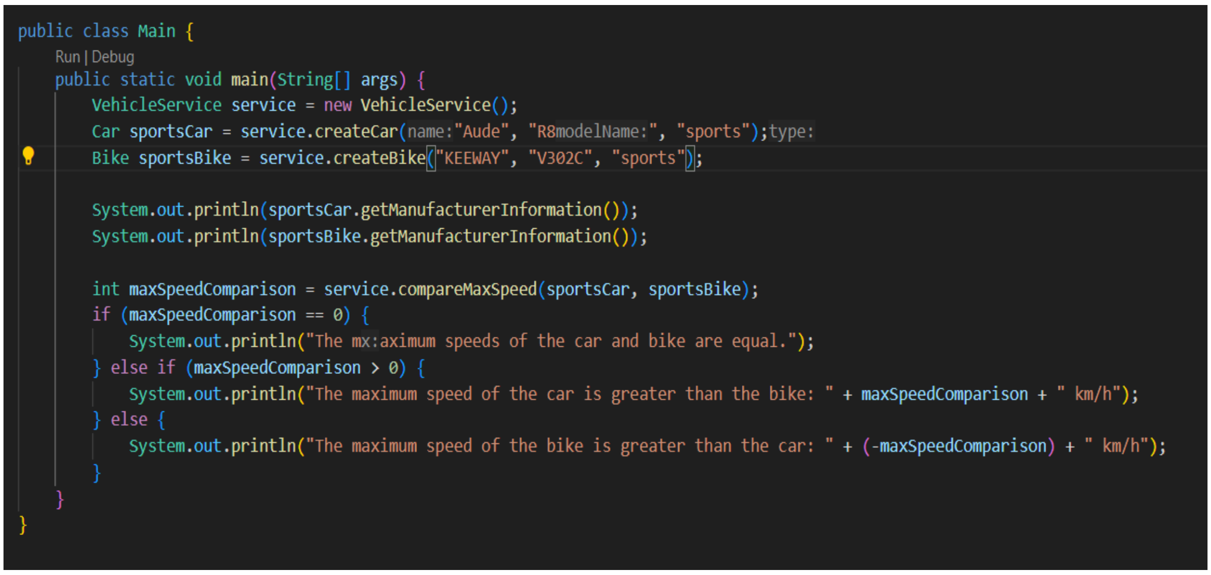
```java
public class Bike extends AbstractManufacturer implements Vehicle {
    public Bike(String name, String modelName, String bikeType) {
        super(name, modelName, bikeType);
    }

    @Override
    public int maxSpeed(String bikeType) {
        if (bikeType.equals("sports")) {
            return 300;
        } else if (bikeType.equals("cruiser")) {
            return 170;
        } else {
            return 0;
        }
    }

    @Override
    public String getManufacturerInformation() {
        return String.format("Bike{Manufacturer name:'%s',Model Name:'%s',Type:'%s'}", getName(), getModelName(),
                getType());
    }
}
```



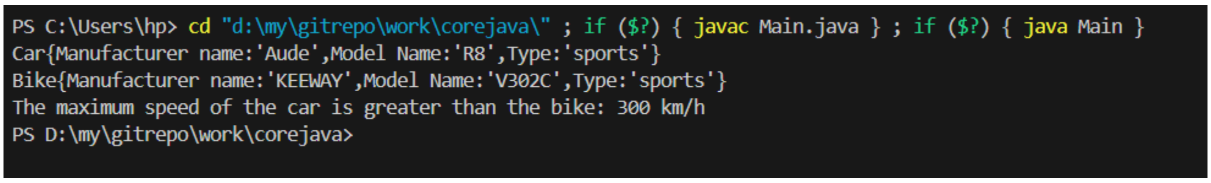
```java
public abstract class AbstractManufacturer {
    private String name;
    private String modelName;
    private String type;

    public AbstractManufacturer(String name, String modelName, String type) {
        this.name = name;
        this.modelName = modelName;
        this.type = type;
    }

    public String getName() {
        return name;
    }

    public String getModelName() {
        return modelName;
    }

    public String getType() {
        return type;
    }

    public abstract String getManufacturerInformation();
}
```



```java
public class VehicleService {
    public Car createCar(String name, String modelName, String type) {
        return new Car(name, modelName, type);
    }

    public Bike createBike(String name, String modelName, String type) {
        return new Bike(name, modelName, type);
    }

    public int compareMaxSpeed(Vehicle first, Vehicle second) {
        if (first.maxSpeed("sporttype:s") > 0 && second.maxSpeed("sporttype:s") > 0) {
            if (first.maxSpeed("sports")type: == second.maxSpeed("sports")type:) {
                return 0;
            }
            return Math.max(first.maxSpeed("sports")type:, second.maxSpeed("sports")type:);
        }
        return -1;
    }
}
```



```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        VehicleService service = new VehicleService();
        Car sportsCar = service.createCar(name:"Aude", "R8modelName:", "sports");type:
        Bike sportsBike = service.createBike("KEEWAY", "V302C", "sports");

        System.out.println(sportsCar.getManufacturerInformation());
        System.out.println(sportsBike.getManufacturerInformation());

        int maxSpeedComparison = service.compareMaxSpeed(sportsCar, sportsBike);
        if (maxSpeedComparison == 0) {
            System.out.println("The mx:aximum speeds of the car and bike are equal.");
        } else if (maxSpeedComparison > 0) {
            System.out.println("The maximum speed of the car is greater than the bike: " + maxSpeedComparison + " km/h");
        } else {
            System.out.println("The maximum speed of the bike is greater than the car: " + (-maxSpeedComparison) + " km/h");
        }
    }
}
```



```
PS C:\Users\hp> cd "d:\my\gitrepo\work\corejava\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Car{Manufacturer name:'Aude',Model Name:'R8',Type:'sports'}
Bike{Manufacturer name:'KEEWAY',Model Name:'V302C',Type:'sports'}
The maximum speed of the car is greater than the bike: 300 km/h
PS D:\my\gitrepo\work\corejava>
```