KBTU KAZAKH-BRITISH TECHNICAL UNIVERSITY

Object-Oriented Programming & Design
Practice Work 3
2 points

Instructor: Izbassar Assylzhan

Deadline: 5th **week, in your practice time. The last defense on week 6. Please, read the class requirements carefully. Recall about naming conventions. Keep your classes separately.**

Problem 1

Total points: 0,5.

Create a class `Animal` with fields `name` and `age`. There should be a constructor with parameters, and two methods: `makeSound()` and `eat()`. You must overload the method `eat()` with a parameter `String food`. Also create a method `getInfo()` that prints all animal information.

Create a class `Dog` that extends `Animal`. As a specific field for `Dog`, create `breed`. The constructor must use `super(...)`. Override the methods `makeSound()` and `getInfo()`. Inside the overridden `getInfo()` method, call `super.getInfo()`.

In `MainApp`, create at least one `Animal` object and at least two `Dog` objects. Store them in a `List<Animal>`. Use a loop to call their methods and demonstrate polymorphism. The **package structure** should be:

- `pr3.animal.model`
- `pr3.animal.app`

Hint: When objects of type `Dog` are stored in `List<Animal>`, the overridden methods should be called automatically.

Problem 2

Total points: 1,5.

Create a class `Engine` with fields `type` and `horsepower`. There should be a constructor with parameters and a method `getEngineInfo()`.

Create a class `Vehicle` with fields `model`, `baseCost`, and `Engine engine`. This demonstrates composition (a vehicle has an engine). Provide a constructor with parameters and a method `calculateDeliveryCost()` with a basic implementation. Also create a method `getVehicleInfo()`.

Create a class `Car` that extends `Vehicle`. As a specific field, create `numberOfSeats`. The constructor must use `super(...)`. Override the method `calculateDeliveryCost()`. Overload a method `calculateDeliveryCost(double extraWeight)`.

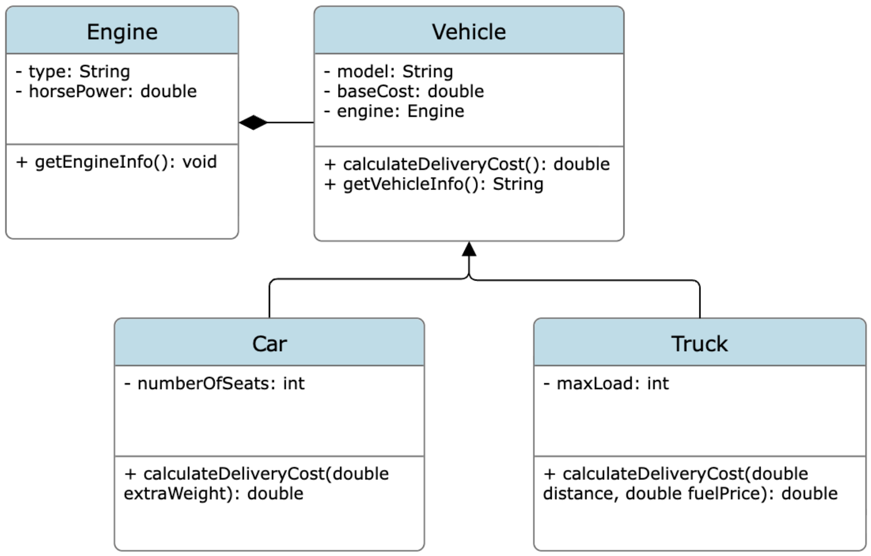
Engine
- type: String
- horsePower: double
+ getEngineInfo(): void
Vehicle
- model: String
- baseCost: double
- engine: Engine
+ calculateDeliveryCost(): double
+ getVehicleInfo(): String
Car
- numberOfSeats: int
+ calculateDeliveryCost(double extraWeight): double
Truck
- maxLoad: int
+ calculateDeliveryCost(double distance, double fuelPrice): double

Figure 1: The UML class diagram for the current task.

Create a class `Truck` that extends `Vehicle`. As a specific field, create `maxLoad`. The constructor must use `super(...)`. Override the method `calculateDeliveryCost()`. Overload a method `calculateDeliveryCost(double distance, double fuelPrice)`.

Create a service class `DeliveryService` with the following methods:

- `printAllVehicles(List<Vehicle> vehicles)`
- `calculateAllDeliveries(List<Vehicle> vehicles)`
- `calculateTotalCost(List<Vehicle> vehicles)`

In `MainApp`, create at least two `Engine` objects, at least one `Car`, and one `Truck`. Store vehicles in a `List<Vehicle>`. Use service methods to test all functionality. The **package structure** should be consist of:

- `pr3.delivery.model`
- `pr3.delivery.service`
- `pr3.delivery.app`

Hint: Demonstrate composition by accessing engine information inside `getVehicleInfo()` and demonstrate polymorphism using `List<Vehicle>`.