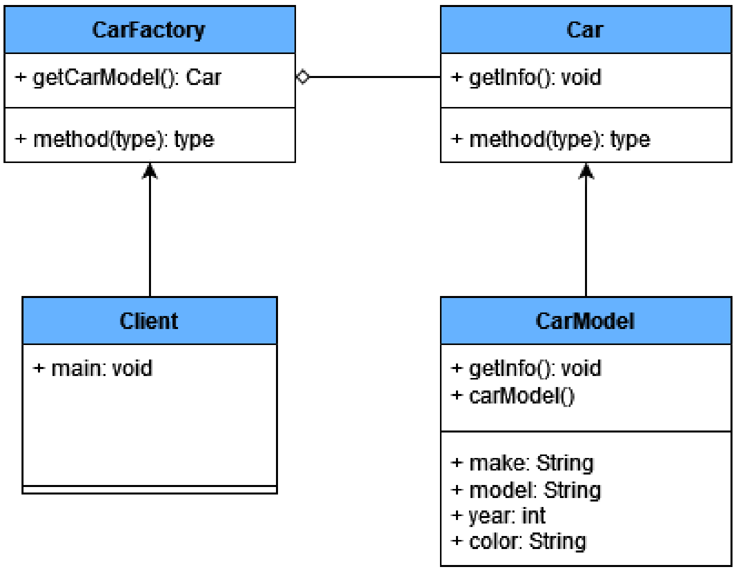
Assignment 9: Flyweight Design Pattern

What is Flyweight Design Pattern?

Flyweight is a structural design pattern that lets you fit more objects into the available amount of RAM by sharing common parts of state between multiple objects instead of keeping all of the data in each object.

Structure (Class Diagram)



Implementation (Code)

```
import java.util.*;

// Flyweight interface
interface Car {
    void getInfo(String owner, String location, int mileage);
}

// Concrete flyweight class
class CarModel implements Car {
    private String make;
    private String model;
    private int year;
    private String color;

    public CarModel(String make, String model, int year, String color) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.color = color;
    }

    public void getInfo(String owner, String location, int mileage) {
        System.out.println("\nCar Model: " + make + " " + model + " " + year + " " + color);
        System.out.println("Current owner: " + owner);
        System.out.println("Current location: " + location);
        System.out.println("Current mileage: " + mileage);
    }
}
```

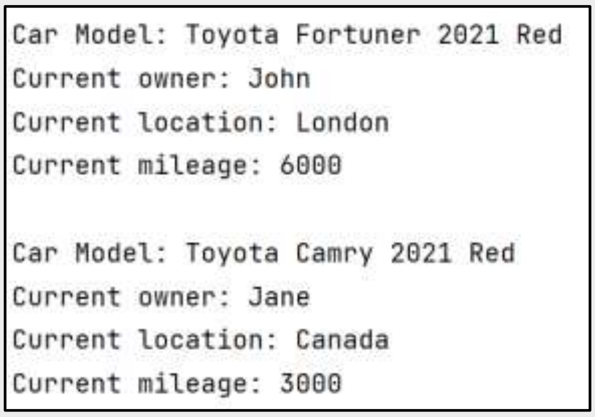
```java
// Flyweight factory
class CarFactory {
    private static Map<String, Car> carModels = new HashMap<>();
    public static Car getCarModel(String make, String model, int year, String color) {
        String key = make + model + year + color;
        Car carModel = carModels.get(key);
        if (carModel == null) {
            carModel = new CarModel(make, model, year, color);
            carModels.put(key, carModel);
        }
        return carModel;
    }
}

// Client code
public class Client {
    public static void main(String[] args) {
        Car car1 = CarFactory.getCarModel("Toyota", "Fortuner", 2021, "Red");
        Car car2 = CarFactory.getCarModel("Toyota", "Camry", 2021, "Red");

        // Both car1 and car2 will reference the same CarModel object
        if (car1 == car2) {
            System.out.println("Car1 and Car2 reference the same CarModel object");
        }

        car1.getInfo("John", "London", 6000);
        car2.getInfo("Jane", "Canada", 3000);
    }
}
```

Output:



```
Car Model: Toyota Fortuner 2021 Red
Current owner: John
Current location: London
Current mileage: 6000

Car Model: Toyota Camry 2021 Red
Current owner: Jane
Current location: Canada
Current mileage: 3000
```

Applicability

1. Use the Flyweight pattern only when your program must support a huge number of objects which barely fit into available RAM.
2. The benefit of applying the pattern depends heavily on how and where it's used. It's most useful when:
   - an application needs to spawn a huge number of similar objects
   - this drains all available RAM on a target device
   - the objects contain duplicate states which can be extracted and shared between multiple objects