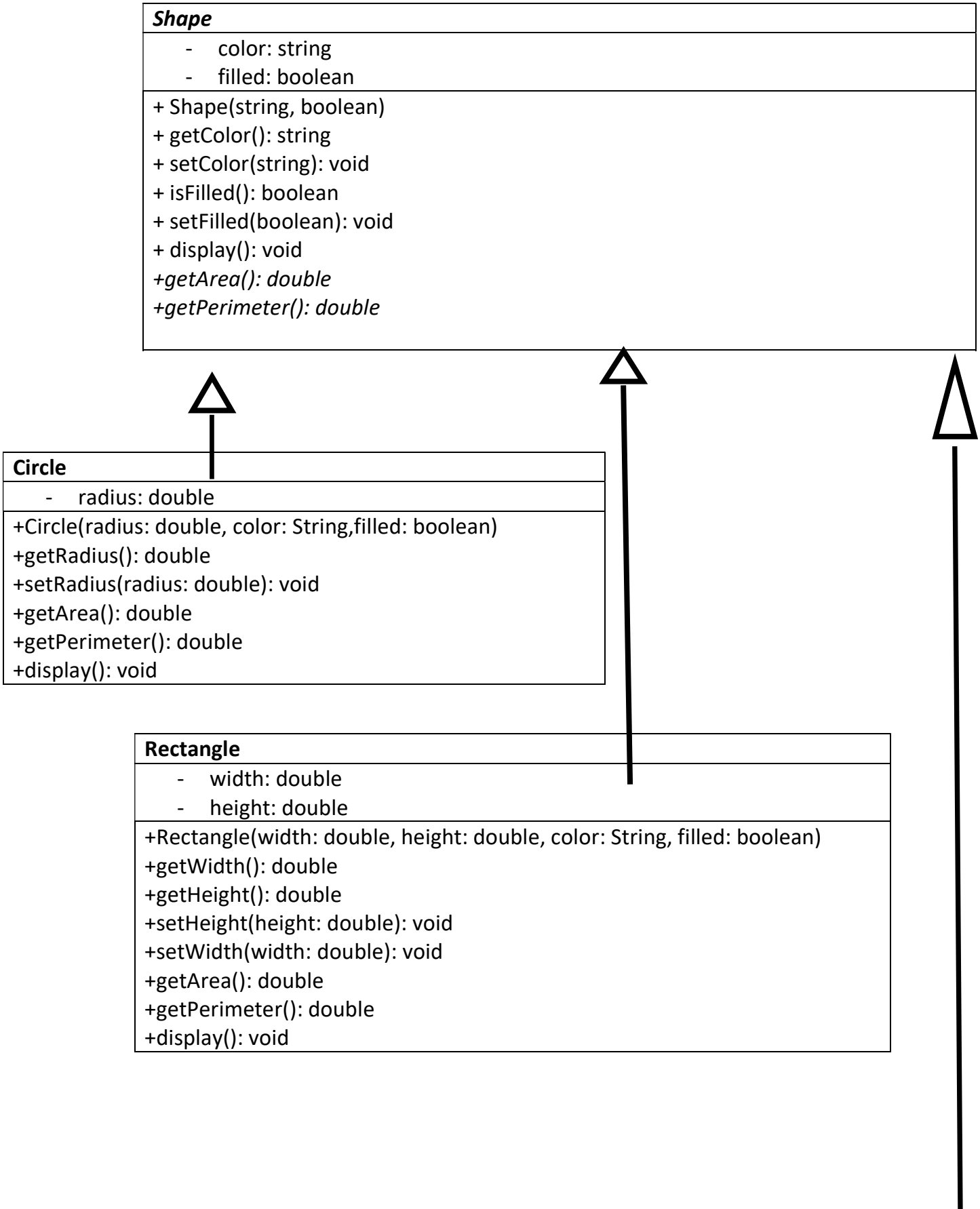
Bài tập về Java Basic (tiếp)

Bài 1. Cho biểu đồ lớp sau:



Square
- side: double
+Square(side: double, color: String, filled: boolean) +getSide(): double +setSide(height: double): void +getArea(): double +getPerimeter(): double +display(): void

Yêu cầu: Định nghĩa lớp cơ sở Shape. Định nghĩa các lớp Circle, Square, Rectangle là các lớp dẫn xuất của Shape. Viết chương trình chính minh họa cách sử dụng.

Bài 2.

- **Xây dựng một giao diện** có tên `Vehicle` để đại diện cho các phương tiện giao thông. Giao diện này bao gồm:

  - Phương thức `void start()` để khởi động phương tiện.
  - Phương thức `void stop()` để dừng phương tiện.
  - Phương thức `double getSpeed()` để lấy tốc độ hiện tại của phương tiện.

- **Tạo lớp `Car`** (Ô tô) triển khai giao diện `Vehicle`. Lớp này chứa:

  - Trường dữ liệu `speed` (private, kiểu `double`): tốc độ hiện tại của ô tô.
  - Phương thức khởi tạo có tham số để khởi tạo tốc độ ban đầu của ô tô.
  - Triển khai phương thức `start()`: đặt tốc độ của ô tô là 50 km/h khi khởi động.
  - Triển khai phương thức `stop()`: đặt tốc độ về 0 khi dừng.
  - Triển khai phương thức `getSpeed()` để trả về tốc độ hiện tại của ô tô.

- **Tạo lớp `Bicycle`** (Xe đạp) triển khai giao diện `Vehicle`. Lớp này chứa:

  - Trường dữ liệu `speed` (private, kiểu `double`): tốc độ hiện tại của xe đạp.
  - Phương thức khởi tạo có tham số để khởi tạo tốc độ ban đầu của xe đạp.
  - Triển khai phương thức `start()`: đặt tốc độ của xe đạp là 15 km/h khi khởi động.
  - Triển khai phương thức `stop()`: đặt tốc độ về 0 khi dừng.
  - Triển khai phương thức `getSpeed()` để trả về tốc độ hiện tại của xe đạp.

- **Tạo lớp `Motorcycle`** (Xe máy) triển khai giao diện `Vehicle`. Lớp này chứa:

  - Trường dữ liệu `speed` (private, kiểu `double`): tốc độ hiện tại của xe máy.
  - Phương thức khởi tạo có tham số để khởi tạo tốc độ ban đầu của xe máy.

- Triển khai phương thức `start()`: đặt tốc độ của xe máy là 80 km/h khi khởi động.
- Triển khai phương thức `stop()`: đặt tốc độ về 0 khi dừng.
- Triển khai phương thức `getSpeed()` để trả về tốc độ hiện tại của xe máy.

- **Viết chương trình chính (`Main`)** để:

  - Tạo một danh sách gồm các đối tượng `Vehicle` (bao gồm `Car`, `Bicycle`, và `Motorcycle`).
  - Khởi động các phương tiện, hiển thị tốc độ ra màn hình sau đó dừng các phương tiện.

Bài 3.

- **Xây dựng giao diện `Vehicle`** để đại diện cho các phương tiện giao thông với các yêu cầu sau:

  - Phương thức `void startEngine()` để khởi động động cơ.
  - Phương thức `void stopEngine()` để tắt động cơ.
  - Phương thức `double getFuelConsumption()` để tính mức tiêu thụ nhiên liệu (lít/km).
  - Phương thức `double getSpeed()` để lấy tốc độ hiện tại của phương tiện.

- **Xây dựng giao diện `ElectricVehicle`** (phương tiện điện), kế thừa từ `Vehicle`. Giao diện này thêm phương thức:

  - `double getBatteryLevel()` để lấy mức pin hiện tại (phần trăm).
  - `void chargeBattery(double amount)` để sạc thêm pin cho phương tiện (đơn vị: %).

- **Tạo lớp `Car`** (ô tô) triển khai giao diện `Vehicle`. Lớp này chứa:

  - Trường dữ liệu `fuelConsumption` (private, kiểu `double`): mức tiêu thụ nhiên liệu (lít/km).
  - Trường dữ liệu `speed` (private, kiểu `double`): tốc độ hiện tại của xe.
  - Trường dữ liệu `engineOn` (private, kiểu `boolean`): cho biết động cơ đang bật hay tắt.
  - Phương thức khởi tạo có tham số để khởi tạo giá trị mức tiêu thụ nhiên liệu và tốc độ ban đầu của xe.
  - Triển khai phương thức `startEngine()`: bật động cơ và đặt tốc độ ban đầu.
  - Triển khai phương thức `stopEngine()`: tắt động cơ và đặt tốc độ về 0.
  - Triển khai phương thức `getFuelConsumption()` để trả về mức tiêu thụ nhiên liệu.
  - Triển khai phương thức `getSpeed()` để trả về tốc độ hiện tại của xe.

- **Tạo lớp `ElectricCar`** (ô tô điện) kế thừa từ lớp `Car` và triển khai thêm giao diện `ElectricVehicle`. Lớp này chứa:

  - Trường dữ liệu `batteryLevel` (private, kiểu `double`): mức pin hiện tại của xe điện (0 - 100%).
  - Phương thức khởi tạo có tham số để khởi tạo giá trị mức pin và mức tiêu thụ nhiên liệu của ô tô.
  - Triển khai phương thức `getBatteryLevel()` để trả về mức pin hiện tại.
  - Triển khai phương thức `chargeBattery(double amount)` để tăng mức pin (với điều kiện tổng mức pin không vượt quá 100%).

- **Tạo lớp `Motorcycle`** (xe máy) triển khai giao diện `Vehicle`. Lớp này chứa:

- Trường dữ liệu `fuelConsumption` (private, kiểu `double`): mức tiêu thụ nhiên liệu.
- Trường dữ liệu `speed` (private, kiểu `double`): tốc độ hiện tại.
- Phương thức khởi tạo có tham số để khởi tạo giá trị tốc độ và mức tiêu thụ nhiên liệu của xe máy.
- Triển khai các phương thức `startEngine()`, `stopEngine()`, `getFuelConsumption()`, và `getSpeed()`.

- **Tạo lớp `TrafficManagementSystem`** để quản lý và theo dõi các phương tiện giao thông. Lớp này chứa:

  - Danh sách các phương tiện (`List<Vehicle> vehicles`).
  - Phương thức `void addVehicle(Vehicle vehicle)` để thêm một phương tiện vào hệ thống.
  - Phương thức `void displayVehiclesInfo()` để hiển thị thông tin chi tiết về tất cả các phương tiện trong hệ thống.
  - Phương thức `void manageFuelAndBattery()` để kiểm tra mức nhiên liệu và pin của các phương tiện, hiển thị cảnh báo nếu cần nạp nhiên liệu hoặc sạc pin.

- **Viết chương trình chính (`Main`)** để:

  - Tạo một số đối tượng `Car`, `ElectricCar`, và `Motorcycle`.
  - Thêm các đối tượng vào hệ thống quản lý giao thông.
  - Gọi các phương thức khởi động, kiểm tra nhiên liệu, và hiển thị thông tin các phương tiện