🧱 **Assessment Problem Statement**

🎯 **Title:**

Automated Vehicle Manufacturing System using Factory and Builder Patterns

---

🏭 **Problem Context**

You are working for an **automobile manufacturing company** that produces **different types of vehicles** — such as **Cars, Bikes, and Trucks**.

Each type of vehicle requires:

- A **factory** to decide *which category of vehicle* to create.
- A **builder** to construct the vehicle step-by-step (engine, wheels, seats, color, etc.) depending on the model configuration (Basic, Sport, Luxury).

The company plans to **automate the creation and configuration** of these vehicles for faster production and better customization.

🧩 **Requirements**

1. Implement a **VehicleFactory** (Factory Pattern) that can produce:
    - Car
    - Bike
    - Truck

Each vehicle must have its own **builder class** that defines how it's assembled.

2. Implement **Builder Pattern** for constructing vehicles in a flexible way:
    - A **Director** should define how to build each configuration (Basic / Sport / Luxury).
    - Each vehicle builder should include methods like:
        - set_engine()
        - set_wheels()
        - set_color()
        - set_features()
        - get_vehicle()
3. When a client requests a new vehicle, the system should:
    - Ask the factory to create the required vehicle type.
    - Use the correct builder to configure it based on a given model (e.g., Sport).
    - Return the final assembled vehicle with all details.

4. Print the final configuration for:
    - A **Sport Car**
    - A **Basic Bike**
    - A **Luxury Truck**

💡 Expected Output Example

```
--- Vehicle Created ---
Type: Car
Model: Sport
Engine: V8 Turbo
Wheels: 4 Alloy Wheels
Color: Red
Features: Sports Mode, Leather Seats, ABS

--- Vehicle Created ---
Type: Bike
Model: Basic
Engine: 150cc
Wheels: 2 Steel Wheels
Color: Black
Features: Standard Brakes

--- Vehicle Created ---
Type: Truck
Model: Luxury
Engine: V12 Heavy Duty
Wheels: 6 Off-Road Wheels
Color: Metallic Blue
Features: GPS, AC Cabin, Trailer Assist
```

⚙️ **What You Need to Implement**

1. **Vehicle (Product Class)**
    - Common base for all vehicles.
2. **Vehicle Builders (Builder Pattern)**
    - CarBuilder, BikeBuilder, TruckBuilder
3. **Vehicle Director**
    - Handles the build order and model configuration.
4. **Vehicle Factory (Factory Pattern)**
    - Returns the correct builder instance based on vehicle type.
5. **Client Code**
    - Requests vehicles and configurations through the factory and director.

🧠 **Skills Tested**

- Understanding of **creational design patterns**
- Applying **Factory Pattern** to manage object creation logic
- Applying **Builder Pattern** for flexible stepwise object construction
- Clean class design and abstraction

🕐 **Estimated Time: 60–90 minutes**

Bonus: Add an option for reading vehicle type and model from user input or a config file.