Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object

We have considered a dealership that produces both winter and summer vehicles where a car’s type will be based on the types of tires and exhaust. Tires could be either winter or summer. The exhaust could be either a single pipe exit or a duel pipe exit.

We are going to create a Part interface representing parts of a car such as the tire type and exhaust type and concrete classes implementing the Part interface.

We then create a Car class having ArrayList of Part and a CarBuilder to build different types of Car objects by combining Item. BuilderPatternDemo, our demo class will use CarBuilder to build a Car.

1. Create Interface Part that represents the tires and exhaust of a car
2. Create abstract classes implementing the part interface providing functionality
3. Create concrete classes that extend Tires and Exhaust classes
4. Create Car class having part objects
5. Create Carbuilder class that is responsible to create car objects
6. BuilderPatternDemo uses CarBuilder to demonstrate builder pattern
7. Verify output