This code demonstrates the **Builder Design Pattern**, which is used to construct complex objects step by step without needing to pass all parameters at once. It allows more readable and flexible object creation, especially when there are many parameters involved.

**Class Breakdown:**

1. **Pizza (Product Class)**:
   * The Pizza class represents the final object that will be built using the PizzaBuilder class.
   * It has four attributes:
     + size (e.g., "Large", "Medium")
     + crust (e.g., "Thin Crust", "Thick Crust")
     + cheese (e.g., "Mozzarella", "Cheddar")
     + toppings (a list of additional ingredients like "Pepperoni", "Mushrooms").
   * The constructor initializes these attributes, and the toString() method is overridden to provide a human-readable string representation of the pizza when printed.
2. **PizzaBuilder (Builder Class)**:
   * This is the **Builder** class used to build a Pizza object in a step-by-step manner.
   * The builder has attributes similar to those of the Pizza class (size, crust, cheese, and toppings), but these are set one at a time through methods.
   * **Builder Methods**:
     + setSize(String size) sets the pizza size and returns the builder object to allow method chaining.
     + setCrust(String crust) sets the crust type.
     + setCheese(String cheese) sets the type of cheese.
     + addTopping(String topping) adds a topping to the toppings list.
     + build() constructs and returns a new Pizza object, passing all the attributes from the builder to the Pizza constructor.
   * This pattern ensures that the pizza is constructed with all the provided parameters once build() is called.
3. **Client (BuilderPatternDemo)**:
   * This is the **client code** that demonstrates how to use the PizzaBuilder to create a Pizza.
   * The pizza is created by chaining builder method calls:
     + setSize("Large"): Sets the pizza size to "Large".
     + setCrust("Thin Crust"): Specifies that the crust should be thin.