1. Singleton

👉 *Exercise*: Create a Settings class that stores app preferences (like theme = dark/light).

- Only **one instance** should exist.
- Test: Settings.getInstance().setTheme("Dark");

2. Factory Method

👉 *Exercise*: Build a NotificationFactory that creates EmailNotification or SMSNotification.

- Test: factory.create("SMS").send("Hello Student");

3. Abstract Factory

👉 *Exercise*: Create a CarPartsFactory that produces either **LuxuryCar parts** (leather seat, turbo engine) or **EconomyCar parts** (fabric seat, basic engine).

- Test: factory.createSeat().getMaterial();

4. Builder

👉 *Exercise*: Design a ComputerBuilder to assemble a PC with optional GPU, RAM size, SSD.

- Test: new ComputerBuilder().addRAM(16).addSSD(512).build();

5. Prototype

👉 *Exercise*: Implement a MazeRoom object with doors, then clone it to make similar rooms instead of re-creating from scratch.

- Test: room.clone() gives a copy with same properties.

6. Adapter

👉 *Exercise*: You have an OldPrinter class with printText(String), and a new system expects printDocument(Document).

- Write an adapter so old printer works with new system.

7. Decorator

👉 *Exercise*: Base class Message → add decorators like EncryptedMessage and HTMLFormattedMessage.

- Test: new EncryptedMessage(new Message("Hello")).getText();

8. Facade

👉 *Exercise*: Make a TravelFacade that books **Flight + Hotel + Taxi** in one call.

- Test: facade.bookTrip("Paris");

9. Composite

👉 *Exercise*: Create a **Menu system** where a Menu can contain MenuItems or other Menus.

- Test: Print full menu tree with one display() call.

10. Proxy

👉 *Exercise*: Create an ImageProxy that loads a high-resolution image only when display() is called.

- Test: Lazy load simulation → first call loads, second call reuses.