Java Demonstration of Different Design Patterns

- Singleton
- Factory
- Strategy
- Observer
- Decorator
- Command
- Template Method
- Adapter
- Builder
- Composite
- Proxy

For collaboration:

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Creational Design
 Pattern: Singleton

```
static class AppConfig {

 private static final AppConfig INSTANCE =
new AppConfig();
  private AppConfig() {}

     public static AppConfig getInstance()
{
             return INSTANCE;
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Creational Design
Pattern: Factory

```
static class PaymentFactory {
        public static PaymentStrategy
getPayment(String type) {
            return switch (type) {
                case "paypal" -> new
PayPalPayment();
                case "card" -> new
CreditCardPayment();
                default -> throw new
IllegalArgumentException("Unknown type");
            };
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Behavioral Design
Pattern: Strategy

```
 interface PaymentStrategy {
        void pay(int amount);
    }

         static  class  CreditCardPayment  implements
PaymentStrategy {
        public void pay(int amount) {
              System.out.println("Paid " + amount + "
with Credit Card");
        }
    }

           static  class  PayPalPayment  implements
PaymentStrategy {
        public void pay(int amount) {
              System.out.println("Paid " + amount + "
with PayPal");
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Behavioral Design
Pattern: Observer

```
interface Subscriber {
        void update(String news);
    }

    static class EmailSubscriber implements Subscriber {
        public void update(String news) {
            System.out.println("Email received: " + news);
        }
    }

    static class NewsPublisher {
         private final List<Subscriber> subs = new ArrayList<>
();
        public void subscribe(Subscriber s) {
            subs.add(s);
        }
        public void notifySubs(String news) {
            subs.forEach(s -> s.update(news));
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Structural Design
Pattern: Decorator

```
interface Notifier {
        void send(String msg);
    }

    static class BasicNotifier implements Notifier {
        public void send(String msg) {
            System.out.println("Sending: " + msg);
        }
    }

    static class SlackDecorator implements Notifier {
        private final Notifier wrap;
        public SlackDecorator(Notifier wrap) {
            this.wrap = wrap;
        }
        public void send(String msg) {
            wrap.send(msg);
            System.out.println("Also sent to Slack: " + msg);
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Structural Design
 Pattern: Command

```
interface Command {
        void execute();
    }

     static class SaveCommand implements Command
{
        public void execute() {
            System.out.println("File saved.");
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Behavioral Design
Pattern: Template Method

```
abstract static class ReportGenerator {
        public final void generate() {
            fetchData();
            formatData();
            export();
        }
        abstract void fetchData();
        abstract void formatData();
        void export() {
            System.out.println("Exported report");
        }
    }

    static class SalesReport extends ReportGenerator {
        void fetchData() {
            System.out.println("Fetched sales data");
        }
        void formatData() {
            System.out.println("Formatted sales report");
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Structural Design
Pattern: Adapter

```
interface JsonExporter {
        String export();
    }

    static class XmlReport {
        String generateXML() {
            return "<data>xml</data>";
        }
    }

    static class XmlToJsonAdapter implements JsonExporter {
        private final XmlReport xml;
        public XmlToJsonAdapter(XmlReport xml) {
            this.xml = xml;
        }
        public String export() {
            return "{\"converted\": true}";
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Creational Design Pattern: Builder

```java
static class User {
        private final String name;
        private final int age;
        private final String email;

        private User(Builder builder) {
            this.name = builder.name;
            this.age = builder.age;
            this.email = builder.email;
        }

        static class Builder {
            private String name;
            private int age;
            private String email;

            public Builder name(String name) { this.name = name; return this; }
            public Builder age(int age) { this.age = age; return this; }
            public Builder email(String email) { this.email = email; return this; }
            public User build() {
                return new User(this);
            }
        }

        public String toString() {
            return name + ", " + age + ", " + email;
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Structural Design
 Pattern: Composite

```
interface UIComponent {
        void render();
    }

    static class Button implements UIComponent {
        public void render() {
            System.out.println("Rendering Button");
        }
    }

    static class Panel implements UIComponent {
        private final List<UIComponent> children = new ArrayList<>();
        public void add(UIComponent c) { children.add(c); }
        public void render() {
            System.out.println("Rendering Panel");
            children.forEach(UIComponent::render);
        }
    }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Category: Structural Design
 Pattern: Proxy

```
interface Image {
       void display();
   }

   static class RealImage implements Image {
       private final String filename;
       public RealImage(String filename) {
           this.filename = filename;
           load();
       }
       private void load() {
           System.out.println("Loading " + filename);
       }
       public void display() {
           System.out.println("Displaying " + filename);
       }
   }

   static class ProxyImage implements Image {
       private RealImage real;
       private final String filename;
       public ProxyImage(String filename) {
           this.filename = filename;
       }
       public void display() {
           if (real == null) real = new RealImage(filename);
           real.display();
       }
   }
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A

Invoking the design patterns

```java
public static void main(String[] args) {
        // Singleton
        AppConfig config = AppConfig.getInstance();

        // Factory + Strategy
        PaymentStrategy payment = PaymentFactory.getPayment("card");
        payment.pay(500);

        // Observer
        NewsPublisher pub = new NewsPublisher();
        pub.subscribe(new EmailSubscriber());
        pub.notifySubs("New release is live!");

        // Decorator
        Notifier notifier = new SlackDecorator(new BasicNotifier());
        notifier.send("Build done");

        // Command
        Command save = new SaveCommand();
        save.execute();

        // Template
        ReportGenerator report = new SalesReport();
        report.generate();

        // Adapter
        JsonExporter adapter = new XmlToJsonAdapter(new XmlReport());
        System.out.println("JSON: " + adapter.export());

        // Builder
        User user = new User.Builder()
                .name("Alice")
                .age(28)
                .email("alice@example.com")
                .build();
        System.out.println("User: " + user);

        // Composite
        Panel panel = new Panel();
        panel.add(new Button());
        panel.add(new Button());
        panel.render();

        // Proxy
        Image img = new ProxyImage("photo.jpg");
        img.display(); // Loads and displays
        img.display(); // Just displays
    }
}
```

Email
nestorabiawuh@gmail.com

LinkedIn
Nestor Abiangang A