Exercise 1: Single-Choice Questions (45 points)

Each question is worth **3 points**. Select only **one correct answer** per question.

1. Which SOLID principle emphasizes depending on abstractions rather than concrete implementations?

```
A) Single Responsibility Principle

B) Dependency Inversion Principle

C) Interface Segregation Principle

D) Liskov Substitution Principle
```

2. Which architectural pattern separates a system into clearly bounded areas, each managing its own data and functionality?

```
A) Shared Database

B) Domain-Driven Design (Bounded Contexts)

C) Remote Procedure Invocation

D) File Transfer Integration
```

3. The Adapter design pattern is primarily used for:

```
A) Creating complex objects

B) Decoupling abstraction from implementation

C) Allowing incompatible interfaces to collaborate

D) Dynamically adding behavior to objects
```

4. What integration style is best suited for asynchronous, decoupled communication between systems?

```
A) Shared Database Integration

B) RPC Integration

C) File Transfer Integration

D) Message Broker Integration
```

5. Which pattern ensures a class has only one instance within a system?

```
A) Prototype

B) Builder

C) Singleton

D) Factory Method
```

6. The Decorator pattern is best used to:

```
A) Optimize memory usage

B) Simplify complex interfaces

C) Dynamically extend object behavior

D) Decouple abstraction from implementation
```

7. Which Enterprise Integration Pattern standardizes various input formats into one common format?

```
A) Message Filter

B) Aggregator

C) Normalizer

D) Splitter
```

8. Which pattern involves wrapping a payload with additional metadata for improved routing, logging, and tracking?

```
A) Envelope Pattern

B) Normalizer

C) Message Translator

D) Aggregator
```

9. Which design pattern provides a surrogate to control access to another object?

```
A) Composite

B) Proxy

C) Decorator

D) Bridge
```

10. Which Domain-Driven Design (DDD) pattern explicitly protects internal domain models from external complexity?

```
A) Shared Kernel

B) Anti-Corruption Layer

C) Published Language

D) Open-Host Service
```

11. Which design pattern clearly separates the algorithm from the object using it, allowing interchangeable algorithms at runtime?

```
A) Observer

B) Strategy

C) Command

D) Template Method
```

12. Which pattern ensures objects can be composed into structures and handled uniformly?

```
A) Decorator

B) Composite

C) Bridge

D) Flyweight
```

13. The Facade pattern is best suited for:

```
A) Simplifying a complex subsystem interface

B) Managing object lifecycle

C) Dynamically adding object responsibilities

D) Optimizing memory usage
```

14. Which Enterprise Integration Pattern explicitly captures messages that fail to deliver successfully for later inspection?

```
A) Message Router

B) Message Translator

C) Dead Letter Channel

D) Aggregator
```

15. Which pattern explicitly divides a single incoming message into multiple separate messages for independent processing?

```
A) Aggregator

B) Splitter

C) Wire Tap

D) Detour
```

Exercise 2: Design Patterns Analysis (20 points)

You're reviewing the architecture of an online ticket-booking platform. Patterns currently used are:

- `TicketBookingFacade`: Simplifies the booking process by managing subsystems (hotels, flights, car rentals).
- `NotificationDecorator`: Dynamically stacks SMS and Email notifications.
- `PaymentAdapter`: Standardizes payment processing from different external gateways.
- `BookingPrototype`: Creates booking templates via cloning for efficiency.

Clearly identify patterns correctly applied and patterns incorrectly applied. For each incorrect usage, explicitly suggest a more suitable pattern and justify your choice.

Exercise 3: Architectural Analysis and Domain Modeling (25 points)

You're designing a patient monitoring system for hospitals that tracks patient vital signs, alerts anomalies in real-time, integrates various medical devices from different manufacturers, and securely manages patient data according to medical regulations.

1. Domain Identification (DDD) Identify and briefly describe bounded contexts relevant to the scenario. (6 points)

2. Architectural Patterns and Styles Specify and justify suitable architectural patterns and styles from the list provided (7 points):

- Layered Architecture
- Microservices Architecture
- Event-Driven Architecture (EDA)
- Service-Oriented Architecture (SOA)
- Client-Server
- Peer-to-Peer
- Monolithic Architecture

3. Critical Non-Functional Requirements (NFRs) Select and justify five critical NFRs. List of most common NFRs is provided for convenience (6 points):

Availability | Performance | Scalability | Security | Reliability | Usability | Maintainability | Interoperability | Portability | Robustness | Flexibility | Auditability | Traceability | Compliance | Accessibility | Fault Tolerance | Recoverability | Testability | Modifiability | Efficiency

4. Enterprise Integration Patterns Choose and describe three relevant integration patterns to achieve robust integration (6 points):

- Publish-Subscribe
- Point-to-Point Channel
- Message Translator
- Content-Based Router
- Aggregator
- Splitter
- Message Filter
- Dead Letter Channel
- Envelope Pattern
- Normalizer
- Recipient List
- Wire Tap
- Control Bus
- Detour