Web programming using Java technologies

Exam

Name:

Group:

Duration: 2 hours Total: 100 points

Part I – Single-Choice Questions (20 points)

One correct answer per question.

1. A REST API allows clients to cache responses for read-only operations. Which HTTP method benefits most from proper HTTP caching semantics?

A. POST B. PUT C. GET D. PATCH

2. Which HTTP status code best indicates a successful creation of a resource?

A. 200 B. 201 C. 204 D. 409

3. Which Spring feature enables loose coupling by resolving dependencies at runtime?

A. Aspect-Oriented Programming B. Dependency Injection

C. Auto-configuration D. Bean post-processing

4. Which Spring annotation is primarily used to define explicit bean creation logic?

A. @Component B. @Bean C. @Service D. @Repository

5. In JPA, what is the main purpose of the persistence context?

A. Execute SQL queries B. Cache entities and track changes

C. Manage database connections D. Serialize entities to JSON

6. Which fetch type is recommended by default for JPA associations?

A. EAGER B. LAZY C. MANUAL D. IMMEDIATE

7. Which test verifies the behavior of a single method in isolation?

A. Integration test B. End-to-end test

C. Unit test D. System test

8. Which Spring annotation is used to define transactional boundaries?

A. @Repository B. @Transactional

C. @Entity D. @Autowired

9. Which situation should result in a 400 Bad Request?

A. Resource not found B. Invalid request payload

C. Concurrent update conflict D. Server crash

10. Which testing principle improves test reliability?

A. Sharing database state between tests B. Fast and deterministic execution

C. Testing multiple layers at once D. Using real external services

Part II – Multiple-Choice Questions (30 points)

Tick the correct answers. One or more answers may be correct. No partial points.

11. Which are good REST API design practices? (3 correct answers)

☐ Use meaningful HTTP status codes

☐ Keep APIs stateless

☐ Encode business logic in URLs

☐ Use nouns rather than verbs in endpoints

12. Which statements about the Spring context are true? (3 correct answers)

☐ Beans are created and managed by the Spring container

☐ Dependency injection reduces tight coupling

☐ Beans must always be singletons

☐ The context controls bean lifecycle

13. Which statements about relationships between JPA entities are correct? (2 correct answers)

☐ @ManyToOne relationships are typically mapped on the owning side

- ☐ @OneToMany is eagerly fetched by default
- ☐ Bidirectional relationships must be kept consistent in application code
- ☐ mappedBy indicates the owning side of the relationship

14. Which responsibilities belong to the Service layer? (3 correct answers)
    - ☐ Business rule enforcement
    - ☐ Transaction demarcation
    - ☐ HTTP request mapping
    - ☐ Coordination of multiple repositories
15. Which statements about unit testing are correct? (3 correct answers)
    - ☐ Dependencies should be mocked when appropriate
    - ☐ Unit tests should avoid loading the full Spring context
    - ☐ Unit tests validate database mappings
    - ☐ Unit tests should be fast
16. Which scenarios are better suited for integration tests? (3 correct answers)
    - ☐ Verifying repository queries
    - ☐ Testing controller logic in isolation
    - ☐ Checking transaction behavior
    - ☐ Validating JPA mappings

Part III – Short Open Questions (30 points)

Answer must fit in the white space below each question. Partial points are given.

17. Explain the difference between constructor injection and field injection in Spring, and why one of them is preferred.

18. What is the role of @Valid in Spring applications?

19. Explain how transactions are managed in Spring with JPA, at what layer the transaction boundaries are typically defined and the reason behind it.

20. Explain the purpose of Spring AOP (aspects) and give one real-world use case where aspects are preferable to explicit code.

21. What should be the main focus of a unit test for a service class?

22. Why should unit tests avoid real databases and external systems?

Part IV – Code review exercise (20 points)

Answer must fit in the white space below the code snippet. Partial points are given.

Scenario

You are working on a backend for a library system. Users can borrow a book.

Business rules:

- A book can be borrowed only if it is available.

- When borrowed, the book becomes unavailable and a loan is created.

Task

Identify at least 4 issues in the following code, by checking that the code implements all business requirements from above, respects REST design and complies with best practices. For each issue, briefly explain why it is a problem and how it should be fixed.

Code under review

```
@RestController
@RequestMapping("/api")
public class BorrowController {

    @Autowired
    private BorrowService borrowService;

    @GetMapping("/borrowBook")
    public ResponseEntity<Object> borrow(@RequestParam Long bookId,
                                         @RequestParam Long userId) {
        try {
            Loan loan = borrowService.borrowBook(bookId, userId);
            return ResponseEntity.ok("Borrowed successfully. Loan id = " + loan.getId());
        } catch (Exception e) {
            return ResponseEntity.ok("Error: " + e.getMessage());
        }
    }
}
```

```java
@Service
public class BorrowService {

    @Autowired
    private BookRepository bookRepository;

    @Autowired
    private LoanRepository loanRepository;

    public Loan borrowBook(Long bookId, Long userId) {
        Book book = bookRepository.findById(bookId).orElse(null);

        if (book == null) {
            throw new RuntimeException("Book not found");
        }

        book.setAvailable(false);
        bookRepository.save(book);
```

```java
        Loan loan = new Loan();
        loan.setBook(book);
        loan.setUserId(userId);
        loan.setLoanDate(LocalDateTime.now());

        Loan saved = loanRepository.save(loan);

        bookRepository.save(book);

        return saved;
    }
}
```

```java
public interface BookRepository extends JpaRepository<Book, Long> { }

public interface LoanRepository extends JpaRepository<Loan, Long> { }
```