✅ 1. How do you test methods that call external APIs and databases but must remain unit tests?

Answer:
Mock the API call (e.g., using `RestTemplate`) and the database call (e.g., using a mocked `JpaRepository`). Keep the test isolated and avoid loading Spring context.

✅ 2. What are test doubles? Name different types.

Answer:
Test doubles are generalized fake objects used in testing. Types:

- **Dummy:** Passed but not used
- **Stub:** Returns pre-set values
- **Mock:** Verifies interaction
- **Spy:** Partial mock
- **Fake:** Working but simplified implementation (e.g., in-memory DB)

✅ 3. How to unit test code that depends on `System.currentTimeMillis()` or `LocalDateTime.now()`?

Answer:
Extract the time logic to a separate `TimeProvider` bean and mock it during tests.

java Copy Edit

```
when(timeProvider.now()).thenReturn(LocalDateTime.of(2024, 1, 1, 10, 0));
```

✅ 4. How to handle testing multithreaded code?

Answer:
Use `CountDownLatch`, `ExecutorService`, or `Awaitility` to synchronize threads during testing. Also, test for race conditions or data inconsistency.

✅ 5. What is Test-Driven Development (TDD)? How do you apply it?

Answer:
TDD is writing tests **before** writing the actual code. Cycle:

1. Write failing test
2. Write minimal code to pass test
3. Refactor

It encourages better design and high test coverage.

✅ 6. How to test private methods without reflection?

Answer:
Refactor logic into smaller, public methods or move it to a new testable class. Test via the public interface instead.

✅ 7. How do you handle unit tests in microservices architecture?

Answer:

- Test business logic in isolation using mocks.
- Avoid calling actual HTTP endpoints; mock Feign clients or `RestTemplate`.
- Use contract testing (e.g., **Spring Cloud Contract**) for API boundaries.

✅ 8. How do you ensure your mocks don't become tightly coupled to implementation?

Answer:
Mock behavior, not structure. Test **what** the service does (e.g., a method is called), not **how** it's implemented.

✅ 9. How can you verify that no unexpected method was called in a test?

Answer:

java Copy Edit

```
verifyNoMoreInteractions(mock1, mock2);
```

✅ 10. How do you mock void methods that throw exceptions?

Answer:

java Copy Edit

```java
doThrow(new RuntimeException("Error")).when(emailService).sendEmail(any());
```

✅ 11. How do you test code using `@Transactional` or DB rollback logic?

Answer:
Test only the logic that should run **before rollback**. For integration testing, use an embedded DB (like H2) and assert rollback behavior using `@Transactional`.

✅ 12. What is the difference between `@WebMvcTest`, `@SpringBootTest`, and `@DataJpaTest`?

Answer:

- `@WebMvcTest` : Loads only web layer (controllers + MVC config)
- `@SpringBootTest` : Loads full Spring context
- `@DataJpaTest` : Loads only JPA layer with embedded DB

✅ 13. How do you write tests that simulate retry logic or timeout handling?

Answer:
Use a mock that throws exceptions on first few calls, then returns success:

java Copy Edit

```java
when(api.call()) .thenThrow(new TimeoutException()) .thenReturn("Success");
```

✅ 14. What are parameter resolvers in JUnit 5?

Answer:
Custom resolvers to inject complex objects into test methods dynamically (like mocks, test data, etc.)

java Copy Edit

```
@ExtendWith(MyParameterResolver.class)
```

✅ 15. How do you test the behavior of circuit breakers (e.g., Resilience4j)?

Answer:

- Inject a `CircuitBreakerRegistry`
- Configure test properties with low failure thresholds
- Mock external calls and assert `CallNotPermittedException` is thrown