[문제 1] 트랜잭션 테스트이다.

_____라인을 채워서

MemberService 의 register()는 saveMember()와 sendWelcomeMail()을 호출한다.

- 둘 중 하나라도 실패하면 전체 트랜잭션이 롤백되어야 한다.

```
@Service
public class MemberService {
    @Autowired private MemberRepository repo;
    @Autowired private MailService mailService;

    @Transactional
    public void register(Member m) {
        repo.save(m);
        mailService.sendWelcomeMail(m);
    }

}
```

[문제 2] readOnly 옵션 실습이다.

_____라인을 채워서

getMemberById() 메서드가 읽기 전용 트랜잭션으로 설정되도록 하시오.

```
public class MemberQueryService {
    @Autowired private MemberRepository repo;

    @Transactional(readOnly = true)
    public Member getMemberById(Long id) {
        return repo.findById(id).orElseThrow();
    }
}
```

[문제 3] 자기 호출 시 트랜잭션 미적용 문제이다.

_____라인을 채워서

outer()에서 inner() 호출 시 트랜잭션이 적용되지 않는 이유를 확인하시오.

```
public class MyService {
    _____________________________
    public void outer() {
        inner();
    }

    public void inner() {
        throw new RuntimeException("에러");
    }
}
```

자기 자신 내부 호출은 프록시 우회를 통해 호출해야한다.

[문제 4] 커스텀 롤백 조건 실습이다.

_____라인을 채워서

OutOfStockException 발생 시 롤백되도록 하시오.

```
public class OrderService {
    @Transactional(rollbackFor = OutOfStockException.class)
    public void orderItem(Long itemId) throws OutOfStockException {
        if (!inStock(itemId)) throw new OutOfStockException();
    }
```

상황	처리 방식
`OutOfStockException`이 unchecked	생략 가능 (`@Transactional`만으로도 롤백)
`OutOfStockException`이 checked	`@Transactional(rollbackFor = OutOfStockException.class)` 필요

[문제 5] REQUIRES_NEW 트랜잭션 실습이다.

_____라인을 채워서

saveLog()가 별도 트랜잭션으로 작동하도록 하시오.

```
public class LogService {
   @Transactional(propagation = Propagation.REQUIRES_NEW)
    public void saveLog(String msg) {
        logRepository.save(new Log(msg));
    }
}
```

[문제 6] 트랜잭션 시간 제한 실습이다.

_____라인을 채워서

2 초 초과 시 자동 롤백되도록 설정하시오.

```
public class TimeoutService {
    __@Transactional(timeout = 2)_
    public void slowProcess() throws InterruptedException {
        Thread.sleep(5000);
    }
}
```

[문제 7] 예외 타입에 따른 롤백 실험이다.

_____라인을 채워서

CheckedException 에 대해서도 롤백되도록 하시오.

```
public class ExceptionService {
    @Transactional(rollbackFor = IOException.class)
    public void test() throws IOException {
        throw new IOException();
    }
}
```

[문제 8] 수동 트랜잭션 제어 실습이다. 진짜 하나도 모르겠음

_____라인을 채워서

TransactionTemplate 을 활용해 롤백 처리를 하시오.

```
public class ManualTxService {
    @Autowired private PlatformTransactionManager txManager;

    public void process() {
        TransactionTemplate template = new TransactionTemplate(txManager);
        template.execute(status -> {
            status.setRollbackOnly();
            return null;
        });
    }
}
```

[문제 9] 트랜잭션 로그 확인 실습이다.

_____라인을 채워서

트랜잭션 커밋/롤백 로그를 출력하시오.

```
application.yml 설정 예시:
logging:
  level:
    org.springframework.transaction: _DEBUG_____________________
```

[문제 10] 트랜잭션 실패 시 재시도 처리 실습이다.

_____라인을 채워서

데드락 발생 시 자동 재시도 하도록 하시오.

```
@Service
public class StockService {
    _@Retryable_____________
    public void updateStock() {
        // 비즈니스 로직
    }
}
```

[문제 11] 트랜잭션 내에서 지연 로딩 N+1 문제를 해결하시오.

_____라인을 채워서

JPA Lazy 로딩으로 인한 N+1 문제를 해결하시오.

```
@Service
public class PostService {
    @Autowired private PostRepository postRepo;

    _________________________________
    public List<Post> getPosts() {
        List<Post> posts = postRepo.findAll();
        posts.forEach(p -> p.getAuthor().getName());
        return posts;
    }
}
```

[문제 12] 여러 트랜잭션 매니저 사용 시 특정 매니저를 지정하시오.

_____라인을 채워서

멀티 데이터소스 환경에서 트랜잭션 매니저를 지정하시오.

```
@Service
public class AuditService {
    @Autowired private AuditRepository auditRepo;

    _________________________________("auditTransactionManager")
    public void saveAudit(AuditLog log) {
        auditRepo.save(log);
    }
}
```

[문제 13] 트랜잭션 내 비동기 호출 문제를 해결하시오.

_____라인을 채워서

비동기 메서드가 트랜잭션 범위 안에서 동작하도록 하시오.

```
@Service
public class ReportService {
    @Autowired private AsyncMailService mailService;

    @Transactional
    public void completeReport() {
        // 보고서 저장
        _____________________________;
    }
}
```

[문제 14] AOP 에서 트랜잭션과 로깅 순서를 제어하시오.

_____라인을 채워서

트랜잭션보다 로깅이 먼저 실행되도록 하시오.

```
@Aspect
@Component
@Order(_____________________________)
public class LoggingAspect {
    @Before("execution(* com.myapp..*(..))")
    public void logStart() {
        System.out.println("메서드 시작 로그");
    }
}
```

[문제 15] 트랜잭션 중 중첩 예외 처리 전략을 설계하시오.

_____라인을 채워서

외부 시스템 실패는 무시하고 DB 저장은 계속되도록 하시오.

```
@Transactional
public void processPayment(Order order) {
    try {
        paymentGateway.send(order);
    } catch (_____________________________) {
        log.warn("결제 시스템 응답 실패");
    }
    orderRepository.save(order);
}
```