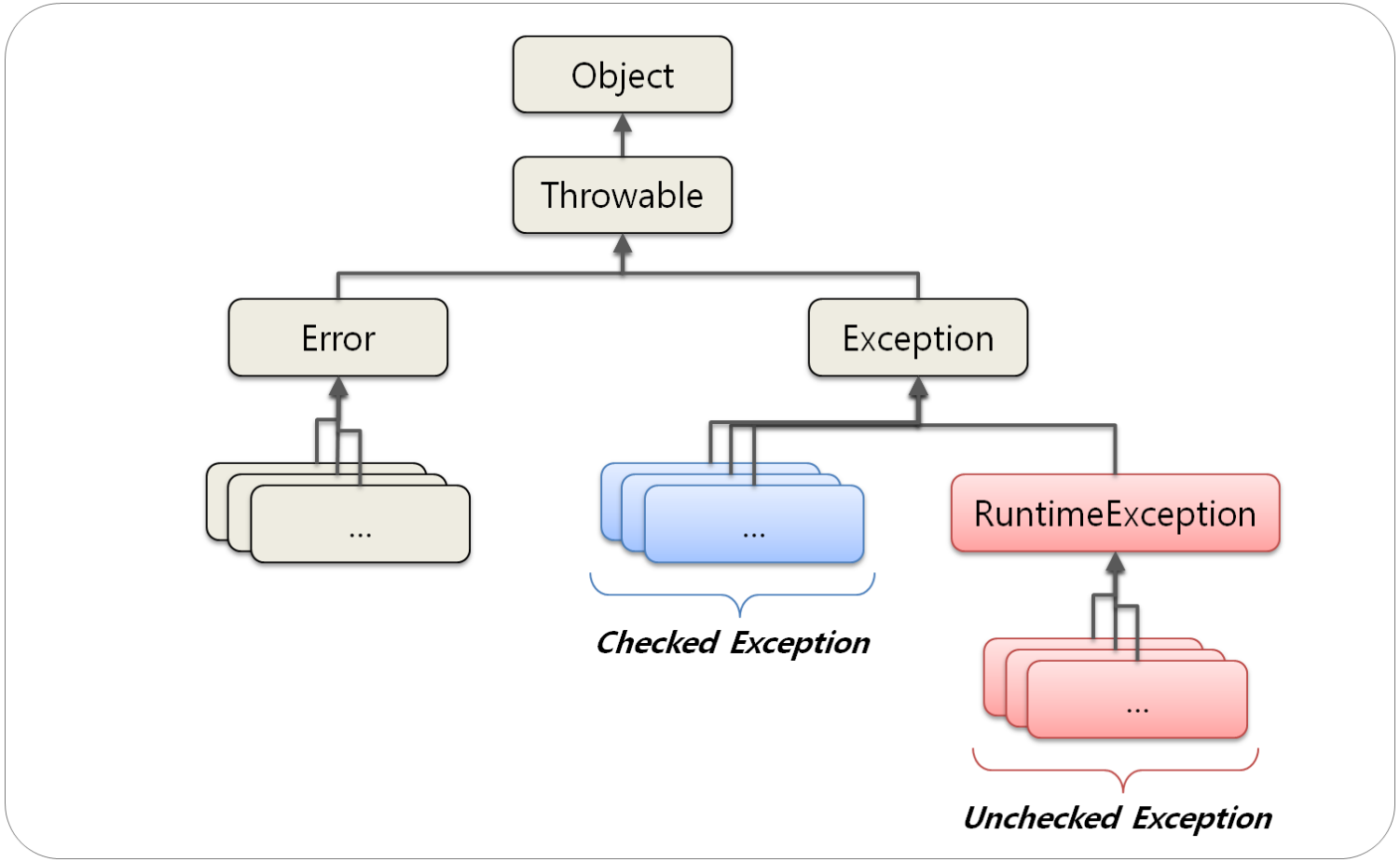
# Java 예외의 종류와 에러 처리 방법

- 조민규 -

# 예외(Exception)의 종류



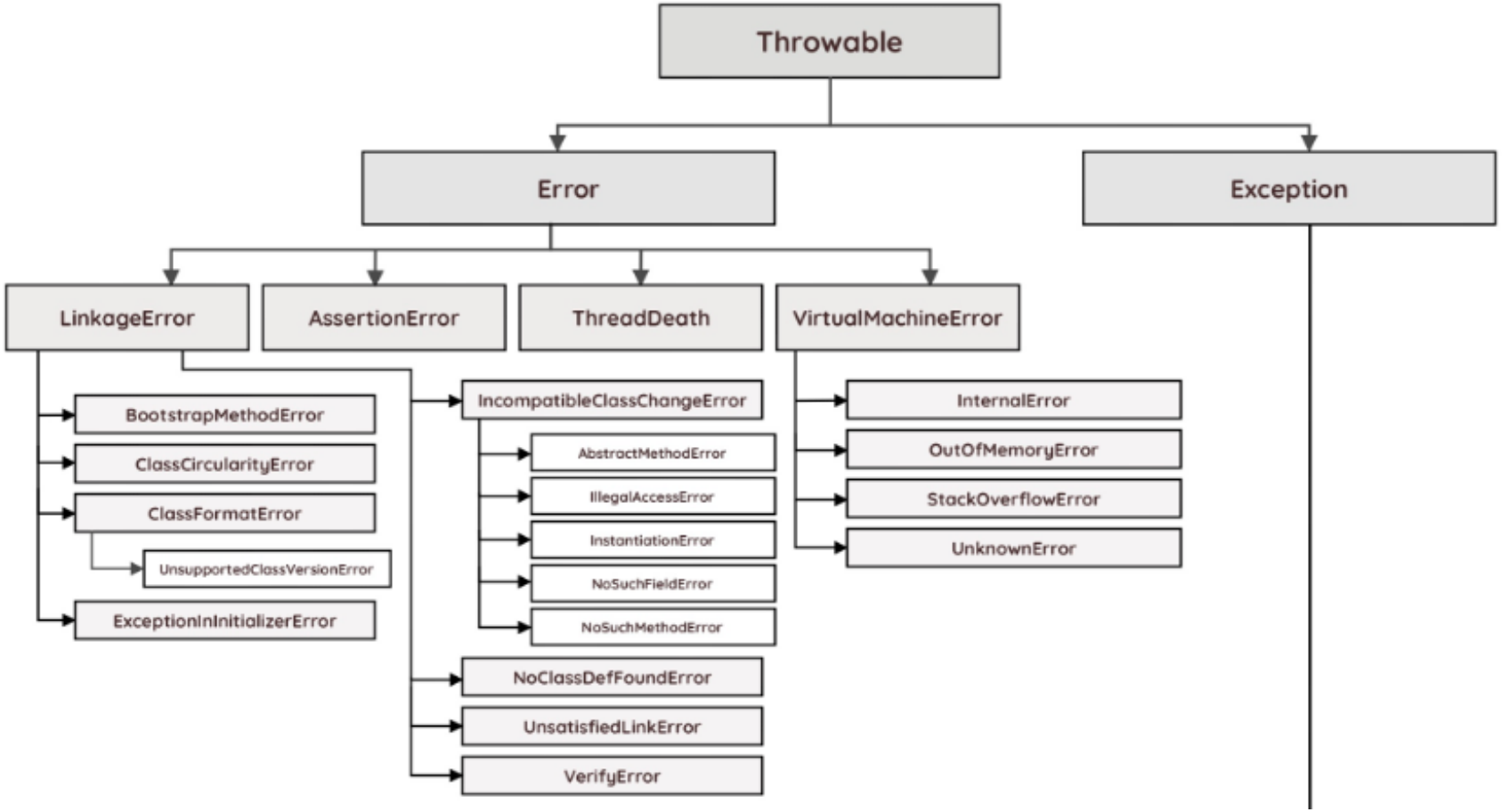
# 예외(Exception)의 종류

- 에러(Error)
- 예외(Exception)
  - 체크 예외(Exception)
  - 언체크 예외(Uncheck Exception)

# 에러(Error)

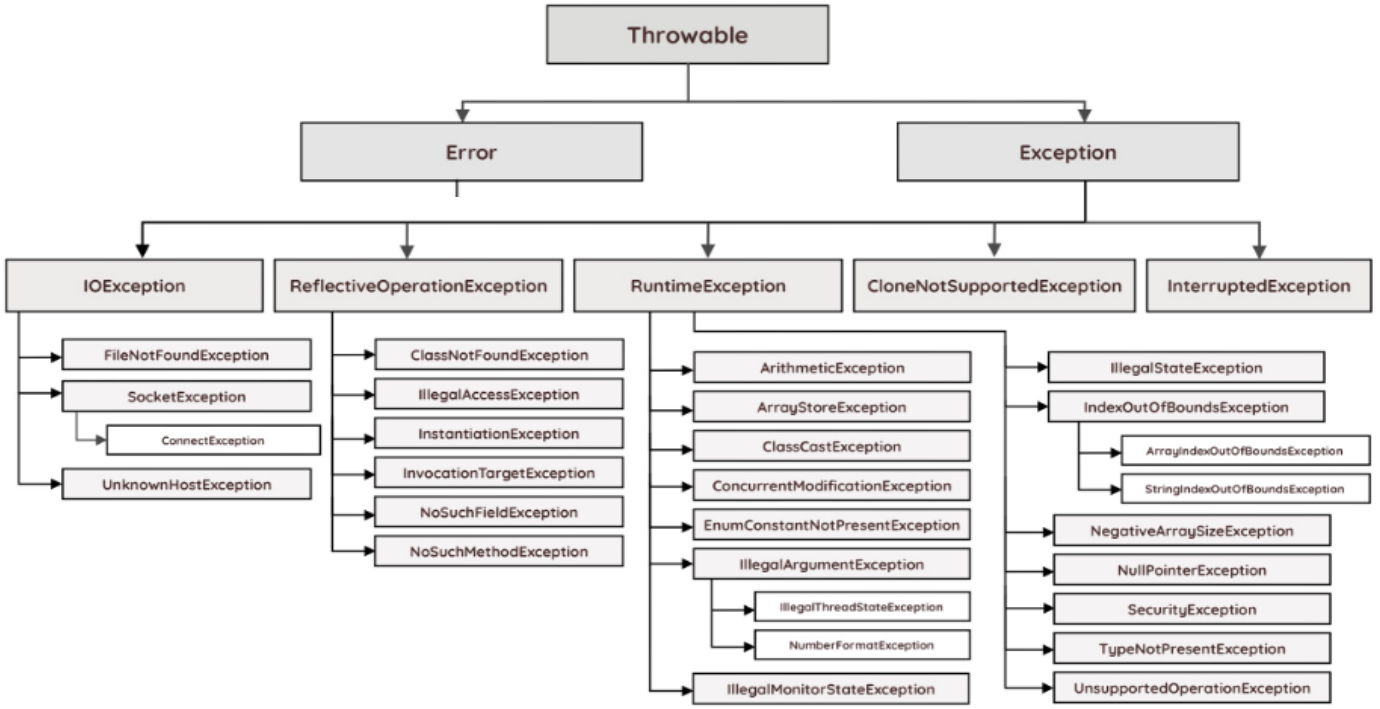
- java.lang.Error 클래스의 서브 클래스들
- 메모리가 부족하는 등 시스템이 비정상적인 경우에 발생
- 주로 JVM에서 발생시키며, 애플리케이션 코드에서 잡아서도 안되고 잡아서 대응도 할 수 없음

# 에러(Error)


Throwable
Error
Exception
LinkageError
AssertionError
ThreadDeath
VirtualMachineError
BootstrapMethodError
ClassCircularityError
ClassFormatError
UnsupportedClassVersionError
ExceptionInInitializerError
IncompatibleClassChangeError
AbstractMethodError
IllegalAccessError
InstantiationError
NoSuchFieldError
NoSuchMethodError
NoClassDefFoundError
UnsatisfiedLinkError
VerifyError
InternalError
OutOfMemoryError
StackOverflowError
UnknownError

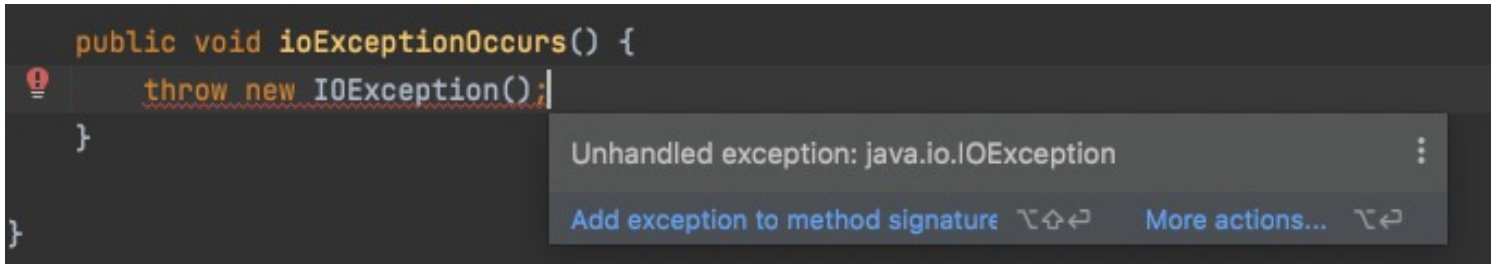
# 예외(Exception)

- 애플리케이션 코드에서 문제가 발생한 경우에 사용
- 체크 예외와 언체크 예외로 나누어짐



# 체크 예외(Check Exception)

- RuntimeException 클래스를 상속하지 않은 Exception 클래스들
- 예상치 못한 예외로써 반드시 예외를 처리하는 코드를 작성해야 함
  - throws …
  - try-catch or try-with-resources
- 예외를 처리하지 않으면 컴파일 에러 발생



# 체크 예외(Check Exception)

- 대표적인 경우: IOException
- Ex) 파일을 열었으면 닫아줘야됨 → 안닫는 것은 의도한게 아님
- 반드시 파일이 닫히도록 체크 예외

# 언체크 예외(Uncheck Exception)

- RuntimeException를 상속한 Exception들은 예외처리를 강제 X = 에러를 처리하지 않아도 컴파일 에러 X
- 프로그램에 문제가 있을 때 오류가 발생하도록 의도된 에러
- 예상하지 못했던 에러가 아니므로 예외 처리를 강제하지 않음

```
public void illegalArgumentExceptionOccurs() {
    throw new IllegalArgumentException();
}
```

# 언체크 예외(Uncheck Exception)

- 대표적인 경우: NullPointerException
- Ex) Email로 사용자를 조회했을 때 없을 수 있음
  = 프로그램 상에서 예상 가능한 에러 → 언체크 예외
- catch or throw해서 처리가 아니라 에러 처리 클래스로 에러 코드 반환

# 예외 처리 방법

1. 예외 복구
   - 예외가 발생한 경우 재시도 해서 예외를 복구
   - Ex) API 호출 실패 시에 retry 적용

2. 예외 처리 회피
   - Throw해서 올바른 위치에서 처리
   - 무분별한 예외 넘겨주기가 되지 않도록 적절하게 사용 필요

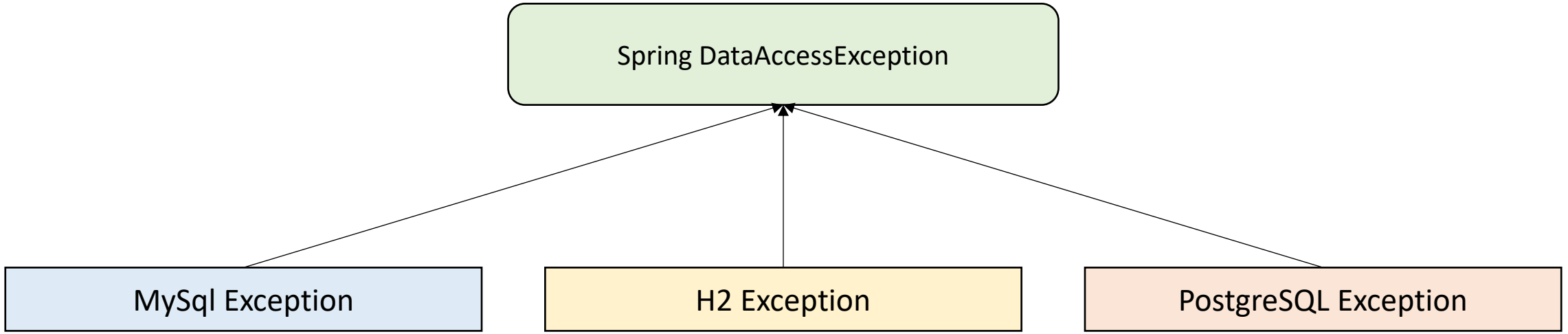
3. 예외 전환
   - 추상화된 예외로 바꾸는 경우
   - 런타임 예외로 포장하여 불필요한 처리를 줄이는 경우

# 예외 전환

- 적절한 의미를 부여하는 Exception으로 전환 → 대응이 쉬워짐
  - Ex) SQL Exception → DuplicatedUserIdException
- 각기 다른 DB에 의한 에러를 DataAcessException으로 추상화
  - 개발자는 서로 다른 경우를 고려하지 않아도 됨
- 체크 예외를 언체크 예외로 변경 → 무분별한 throws 문제를 해결

# Spring의 예외 전환 예시

- 각기 다른 DB에 의한 에러를 DataAcessException으로 추상화
  - 개발자는 서로 다른 DB를 고려하지 않아도 됨
  - 체크 예외를 언체크 예외로 전환하여 불필요한 throw 방지



```
org.springframework.dao.DataIntegrityViolationException: could not execute statement; SQL [n/a]; constraint ["PUBLIC.UK_MBMCQELTY0FBRVXP1Q58DN57T_INDEX_8 ON PUBLIC.MEMBER(EMAIL) VALUE
insert into member (id, created_at, is_enable, updated_at, email, quiz_level, quiz_size, resource_id) values (null, ?, ?, ?, ?, ?, ?, ?) [23505-200]]; nested exception is org.hibernat

    at org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:276)
    at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:233)
    at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExceptionIfPossible(AbstractEntityManagerFactoryBean.java:551)
    at org.springframework.dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible(ChainedPersistenceExceptionTranslator.java:61)
    at org.springframework.dao.support.DataAccessUtils.translateIfNecessary(DataAccessUtils.java:242)
    at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:152)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.data.jpa.repository.support.CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor.invoke(CrudMethodMetadataPostProcessor.java:174)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:97)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:215) <1 internal line>
    at com.mangkyu.employment.interview.app.member.repository.MemberRepositoryTest.selectMemberListByCycle(MemberRepositoryTest.java:40) <31 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <9 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <25 internal lines>
Caused by: org.hibernate.exception.ConstraintViolationException Create breakpoint : could not execute statement <28 internal lines>
    at org.springframework.orm.jpa.SharedEntityManagerCreator$SharedEntityManagerInvocationHandler.invoke(SharedEntityManagerCreator.java:311) <1 internal line>
    at org.springframework.data.jpa.repository.support.SimpleJpaRepository.save(SimpleJpaRepository.java:624) <4 internal lines>
    at org.springframework.data.repository.core.support.RepositoryMethodInvoker$RepositoryFragmentMethodInvoker.lambda$new$0(RepositoryMethodInvoker.java:289)
    at org.springframework.data.repository.core.support.RepositoryMethodInvoker.doInvoke(RepositoryMethodInvoker.java:137)
    at org.springframework.data.repository.core.support.RepositoryMethodInvoker.invoke(RepositoryMethodInvoker.java:121)
insert into member (id, created_at, is_enable, updated_at, email, quiz_level, quiz_size, resource_id) values (null, ?, ?, ?, ?, ?, ?, ?) [23505-200]
    at org.h2.message.DbException.getJdbcSQLException(DbException.java:459)
    at org.h2.message.DbException.getJdbcSQLException(DbException.java:429)
    at org.h2.message.DbException.get(DbException.java:205)
    at org.h2.message.DbException.get(DbException.java:181)
    at org.h2.index.BaseIndex.getDuplicateKeyException(BaseIndex.java:103)
    at org.h2.mvstore.db.MVSecondaryIndex.checkUnique(MVSecondaryIndex.java:221)
    at org.h2.mvstore.db.MVSecondaryIndex.add(MVSecondaryIndex.java:196)
    at org.h2.mvstore.db.MVTable.addRow(MVTable.java:531)
    at org.h2.command.dml.Insert.insertRows(Insert.java:195)
    at org.h2.command.dml.Insert.update(Insert.java:151)
    at org.h2.command.CommandContainer.executeUpdateWithGeneratedKeys(CommandContainer.java:272)
    at org.h2.command.CommandContainer.update(CommandContainer.java:191)
    at org.h2.command.Command.executeUpdate(Command.java:251)
    at org.h2.jdbc.JdbcPreparedStatement.executeUpdateInternal(JdbcPreparedStatement.java:191)
    at org.h2.jdbc.JdbcPreparedStatement.executeUpdate(JdbcPreparedStatement.java:152) <1 internal line>
```

# 올바른 예외 처리 방법

- 조치가 없는 try-catch → 상황에 맞는 조치 진행
- 조치가 불가능한 무분별한 throws → 언체크 예외로 전환
- try-catch → try-with-resources(다음에…)

# 애플리케이션 로직에서 언체크 예외를 사용해야 하는 이유

- 체크 예외
  - 개발자가 예상치 못한, 의도하지 않은 에러
  - 예외 처리가 강제됨
  - 처리가 강제되므로 스프링의 트랜잭션(@Transactional)이 롤백 처리하지 않음

- 언체크 예외
  - 개발자가 알고 있으며 발생가능한 에러
  - 예외 처리를 강제하지 않음
  - 처리가 강제되지 않으므로 스프링의 트랜잭션(@Transactional)이 롤백 처리

```java
@Service
@RequiredArgsConstructor
@Transactional(readOnly = true)
public class UserService {

    private final UserRepository userRepository;

    @Transactional
    public void ioExceptionOccurs(final User user) throws IOException {
        userRepository.save(user);
        throw new IOException();
    }

    @Transactional
    public void illegalArgumentExceptionOccurs(final User user) {
        userRepository.save(user);
        throw new IllegalArgumentException();
    }

}
```