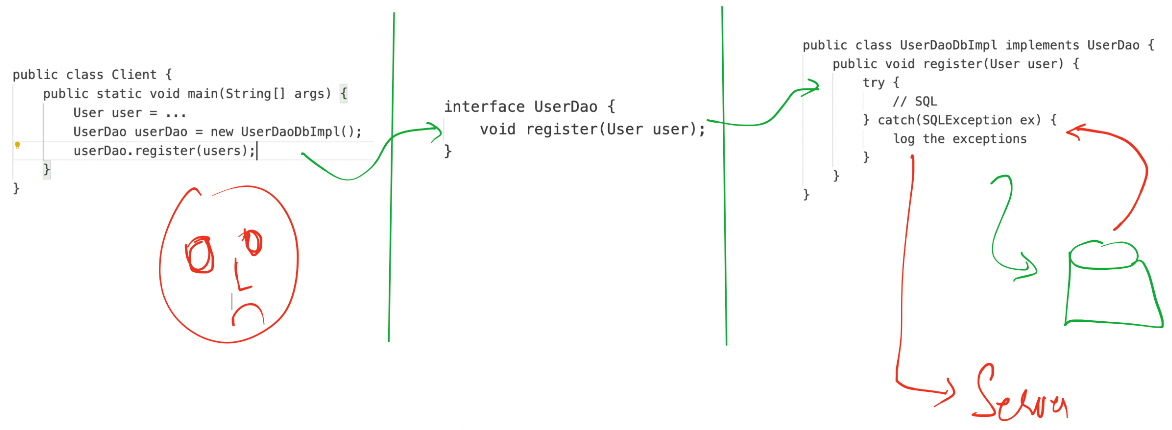
Case 1:
Client has no clue about exceptions occurred on server side/implementation side

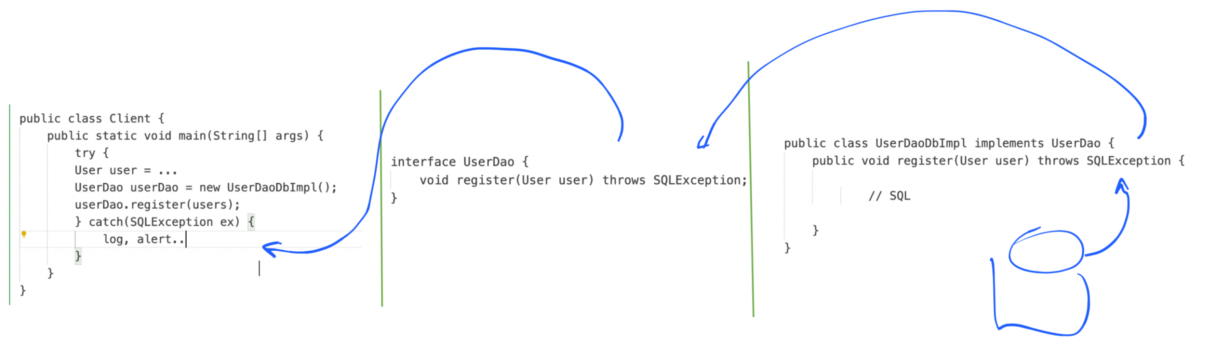


```
public class Client {
    public static void main(String[] args) {
        User user = ...
        UserDao userDao = new UserDaoDbImpl();
        userDao.register(users);
    }
}
```

```
interface UserDao {
    void register(User user);
}
```

```
public class UserDaoDbImpl implements UserDao {
    public void register(User user) {
        try {
            // SQL
        } catch(SQLException ex) {
            log the exceptions
        }
    }
}
```

Case 2: propagating the exceptions



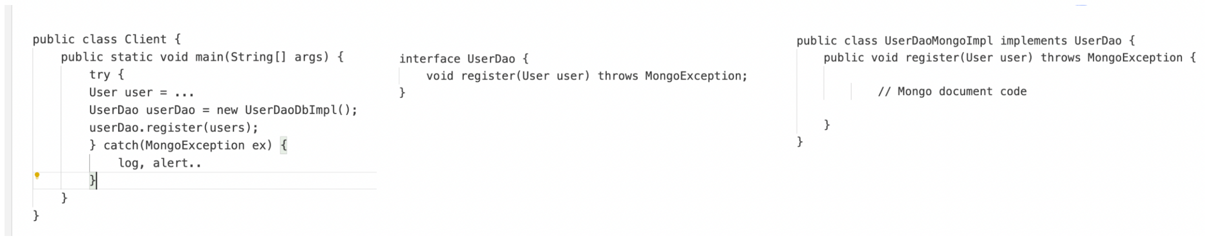
```
public class Client {
    public static void main(String[] args) {
        try {
        User user = ...
        UserDao userDao = new UserDaoDbImpl();
        userDao.register(users);
        } catch(SQLException ex) {
            log, alert..
        }
    }
}
```

```
interface UserDao {
    void register(User user) throws SQLException;
}
```

```
public class UserDaoDbImpl implements UserDao {
    public void register(User user) throws SQLException {

        // SQL

    }
}
```



```
public class Client {
    public static void main(String[] args) {
        try {
        User user = ...
        UserDao userDao = new UserDaoDbImpl();
        userDao.register(users);
        } catch(MongoException ex) {
            log, alert..
        }
    }
}
```

```
interface UserDao {
    void register(User user) throws MongoException;
}
```

```
public class UserDaoMongoImpl implements UserDao {
    public void register(User user) throws MongoException {

        // Mongo document code

    }
}
```

More problems:

1) No abstraction, exposing the backend details like SQLException, infact the exception will also have table name, column names,,,
2) Exception messages are not explanatory
   ORA9001: unique key constraint violation exception

Case 3: Prefer using User defined exception classes for abstraction

```java
public class Client {
    public static void main(String[] args) {
        try {
        User user = ...
        UserDao userDao = new UserDaoDbImpl();
        userDao.register(users);
        } catch(PersistenceException ex) {
            log, alert..
        }
    }
}
```

```java
class PersistenceException extends Exception {
    ...
}

interface UserDao {
    void register(User user) throws PersistenceException;
}
```

```java
public class UserDaoDbImpl implements UserDao {
    public void register(User user) throws PersistenceException {
        try {
            // SQL
        } catch(SQLException ex) {
         if(ex.getErrorCode() == 1504) {
            throw new PersistenceException("User with " + user.getMail() + " already exists!!");
         }
        }

    }
}
```

```java
public class UserDaoMongoImpl implements UserDao {
    public void register(User user) throws MongoException {
        try {
            // SQL
        } catch(MongoException ex) {
         if(ex.getErrorCode() == 9813) {
            throw new PersistenceException("User with " + user.getMail() + " already exists!!");
         }
        }

    }
}
```