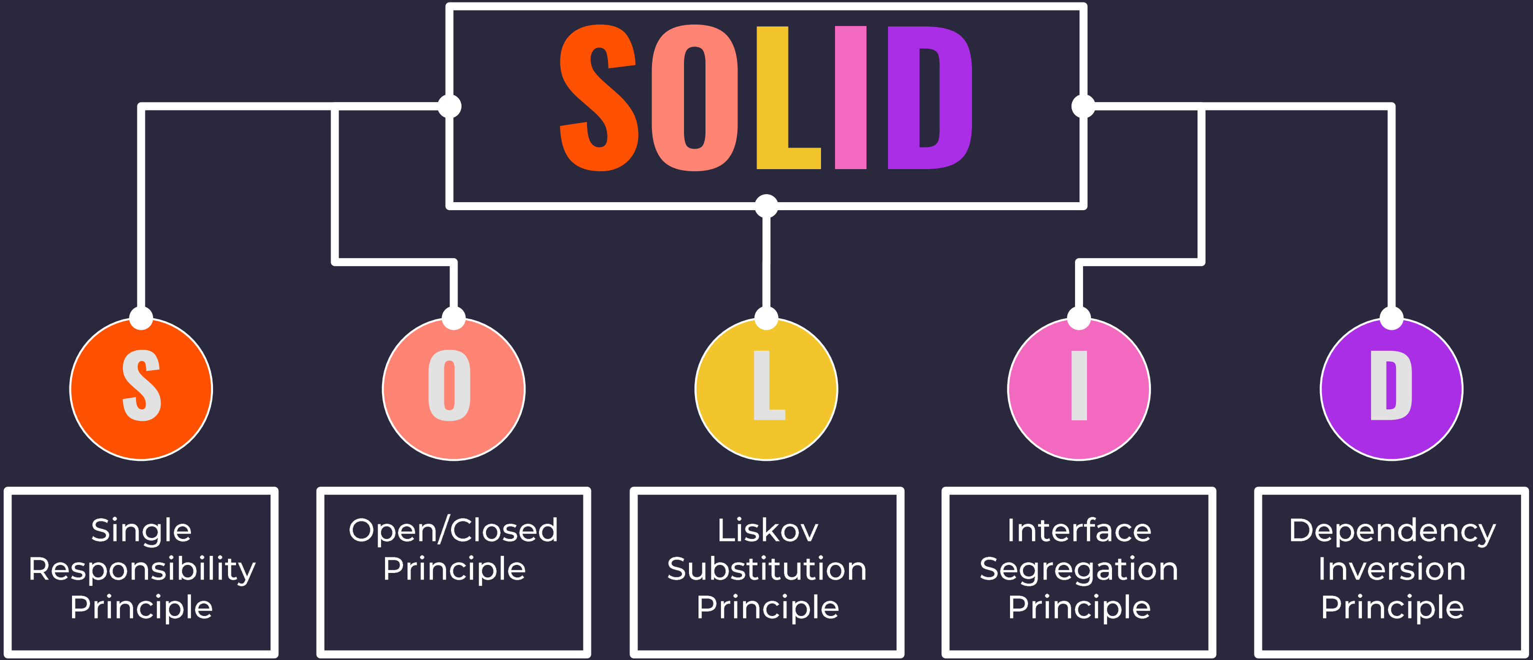
SQ

SOLID Principles

SQ


SOLID
S
O
L
I
D
Single Responsibility Principle
Open/Closed Principle
Liskov Substitution Principle
Interface Segregation Principle
Dependency Inversion Principle

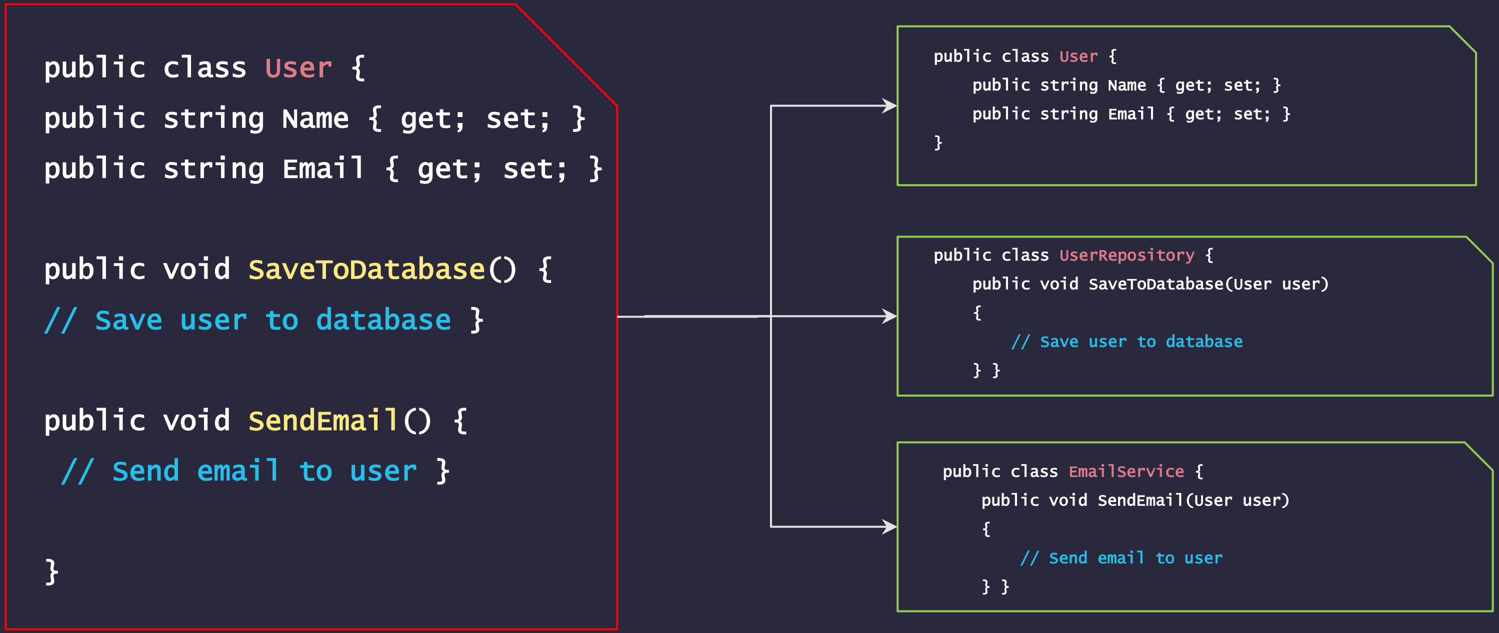
SQ

Single Responsibility Principle

Uncle Bob:

"There should never be more than one reason for a class to change."

Single Responsibility Principle



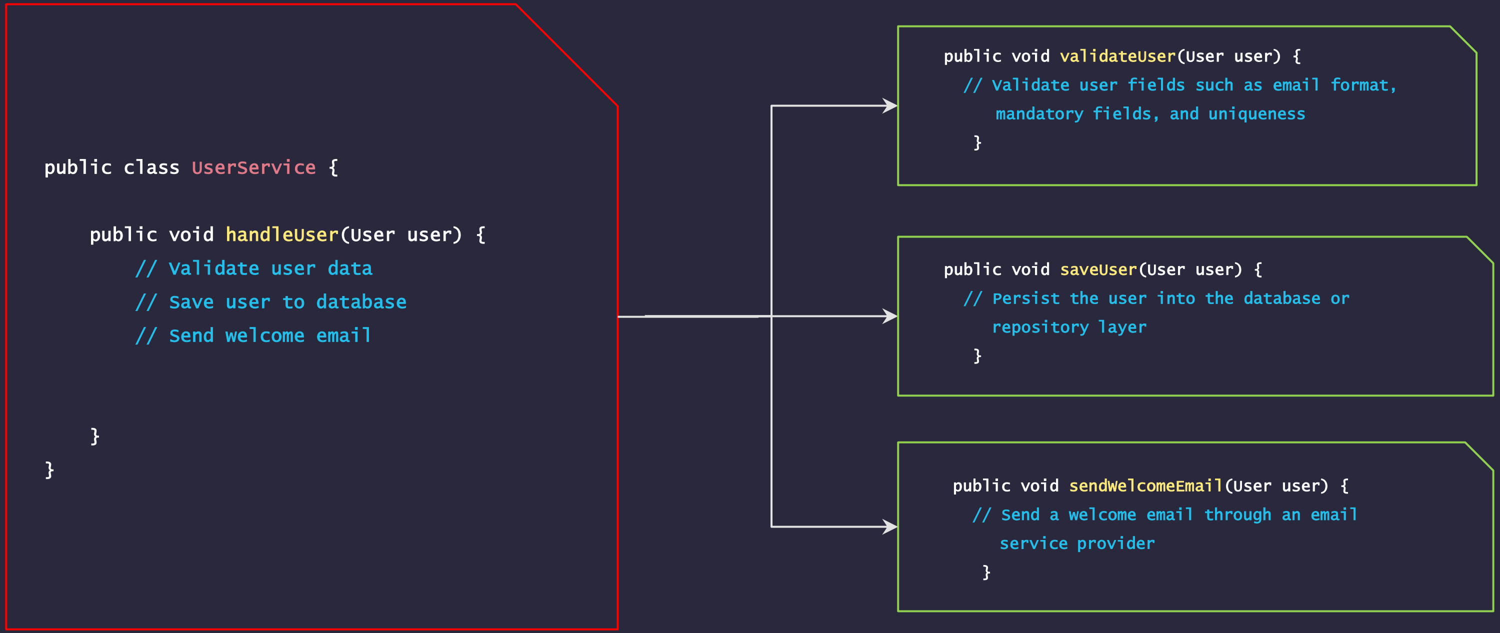
```
public class User {
public string Name { get; set; }
public string Email { get; set; }

public void SaveToDatabase() {
// Save user to database }

public void SendEmail() {
 // Send email to user }

}
```

```
public class User {
    public string Name { get; set; }
    public string Email { get; set; }
}
```

```
public class UserRepository {
    public void SaveToDatabase(User user)
    {
        // Save user to database
    } }
```

```
public class EmailService {
    public void SendEmail(User user)
    {
        // Send email to user
    } }
```

Single Responsibility Principle



```
public class UserService {

    public void handleUser(User user) {
        // Validate user data
        // Save user to database
        // Send welcome email


    }
}
```

```
public void validateUser(User user) {
  // Validate user fields such as email format,
     mandatory fields, and uniqueness
   }
```

```
public void saveUser(User user) {
  // Persist the user into the database or
     repository layer
   }
```

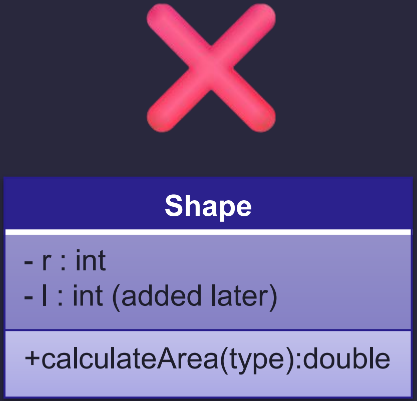
```
public void sendWelcomeEmail(User user) {
  // Send a welcome email through an email
     service provider
   }
```

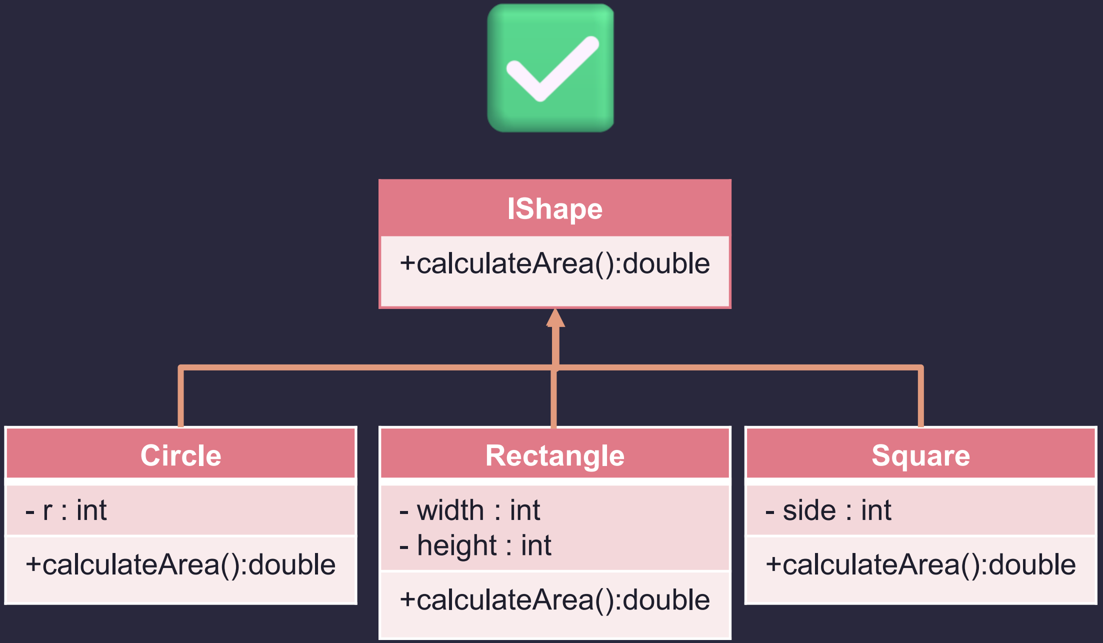
SQ

Open/Closed Principle

"The open-closed principle states that software entities (classes, modules, functions, and so on) should be open for extension but closed for modification."

Open/Closed Principle





SQ

Liskov Substitution Principle

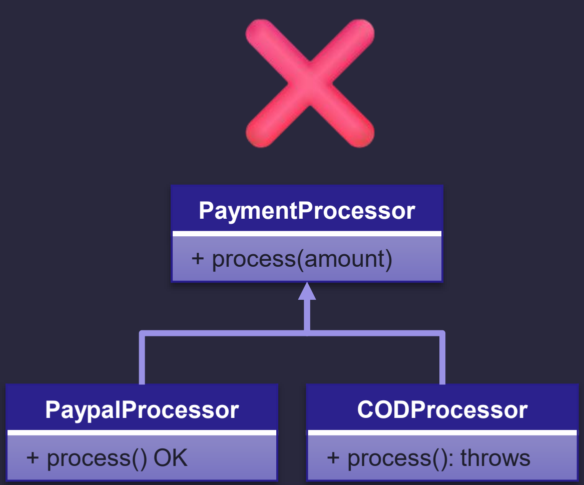
"Objects of a superclass should be replaceable with objects of its subclasses without affecting the correctness of the program."

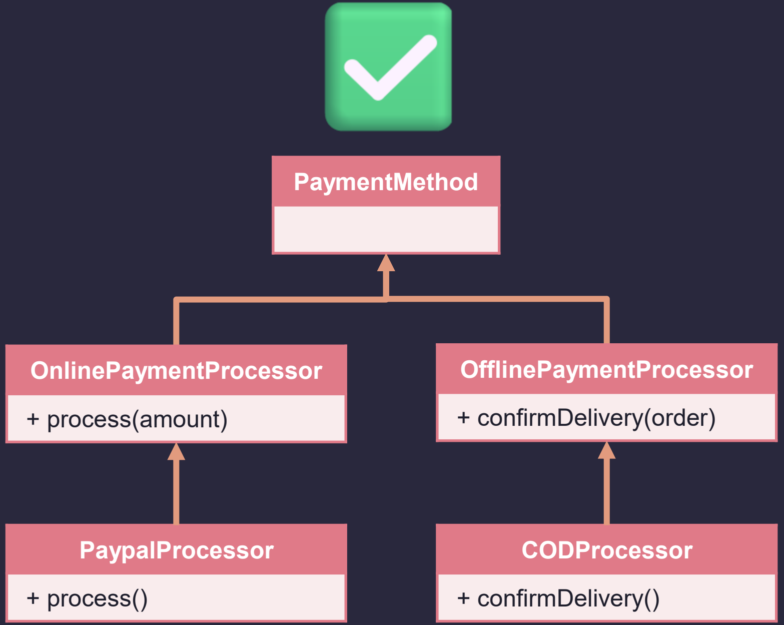
OR

"If you have class B inherits from class A then class A should be replaceable by class B without any changes."

SQ

Liskov Substitution Principle





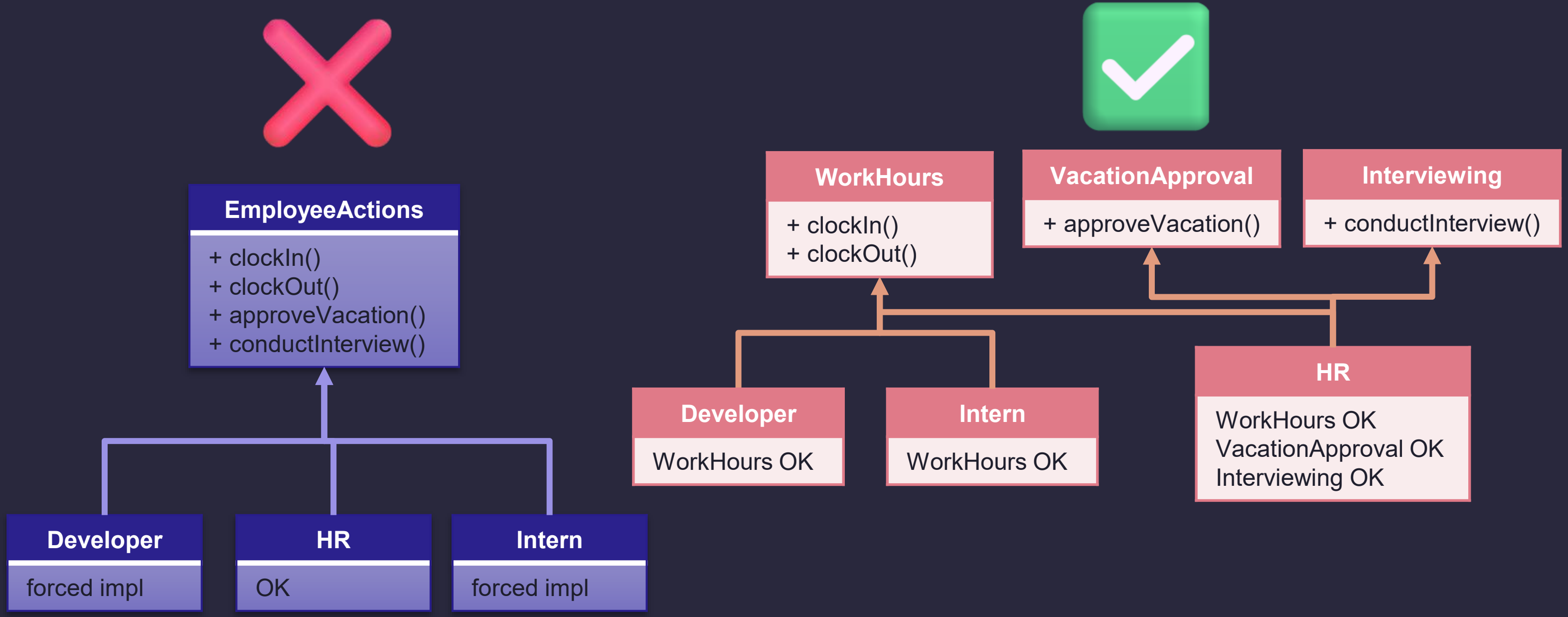
SQ

Interface Segregation Principle

"Clients should not be forced to depend on methods they do not use."

SQ

Interface Segregation Principle



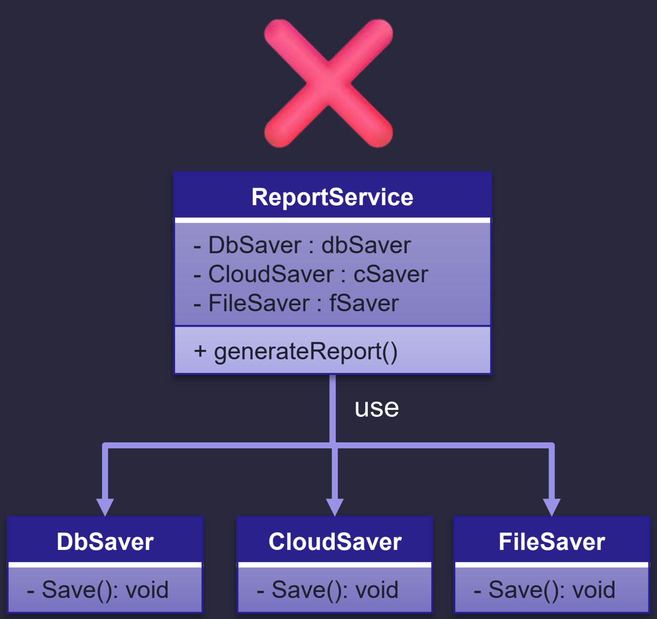
SQ

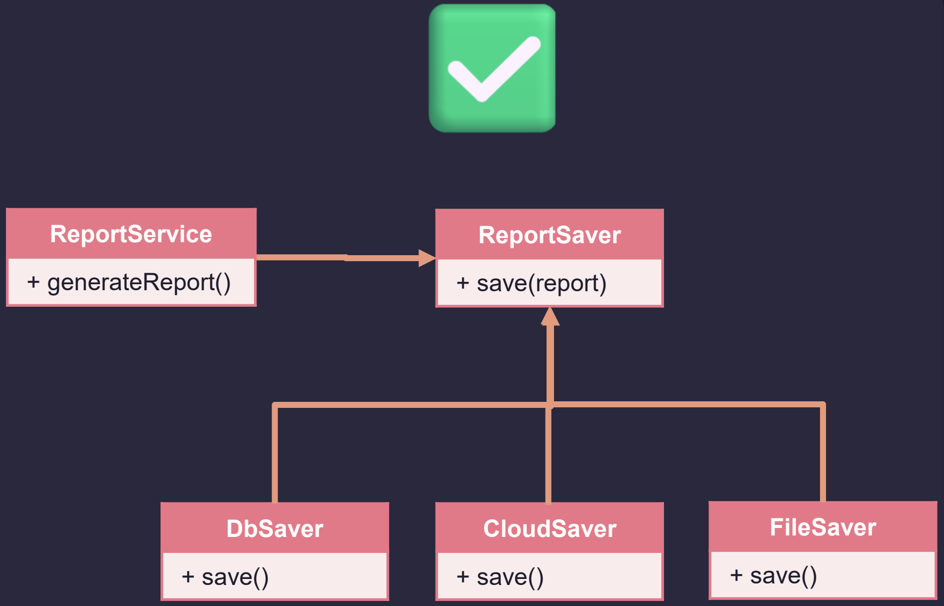
Dependency Inversion Principle

"High-level modules should not depend on low-level modules. Both should depend on abstractions."

SQ

Dependency Inversion Principle





SQ

THANKS !