DePaul University

Design Principles:

S.O.L.I.D.

(2)

Object-oriented Software Development
SE 350– Spring 2021

Vahid Alizadeh

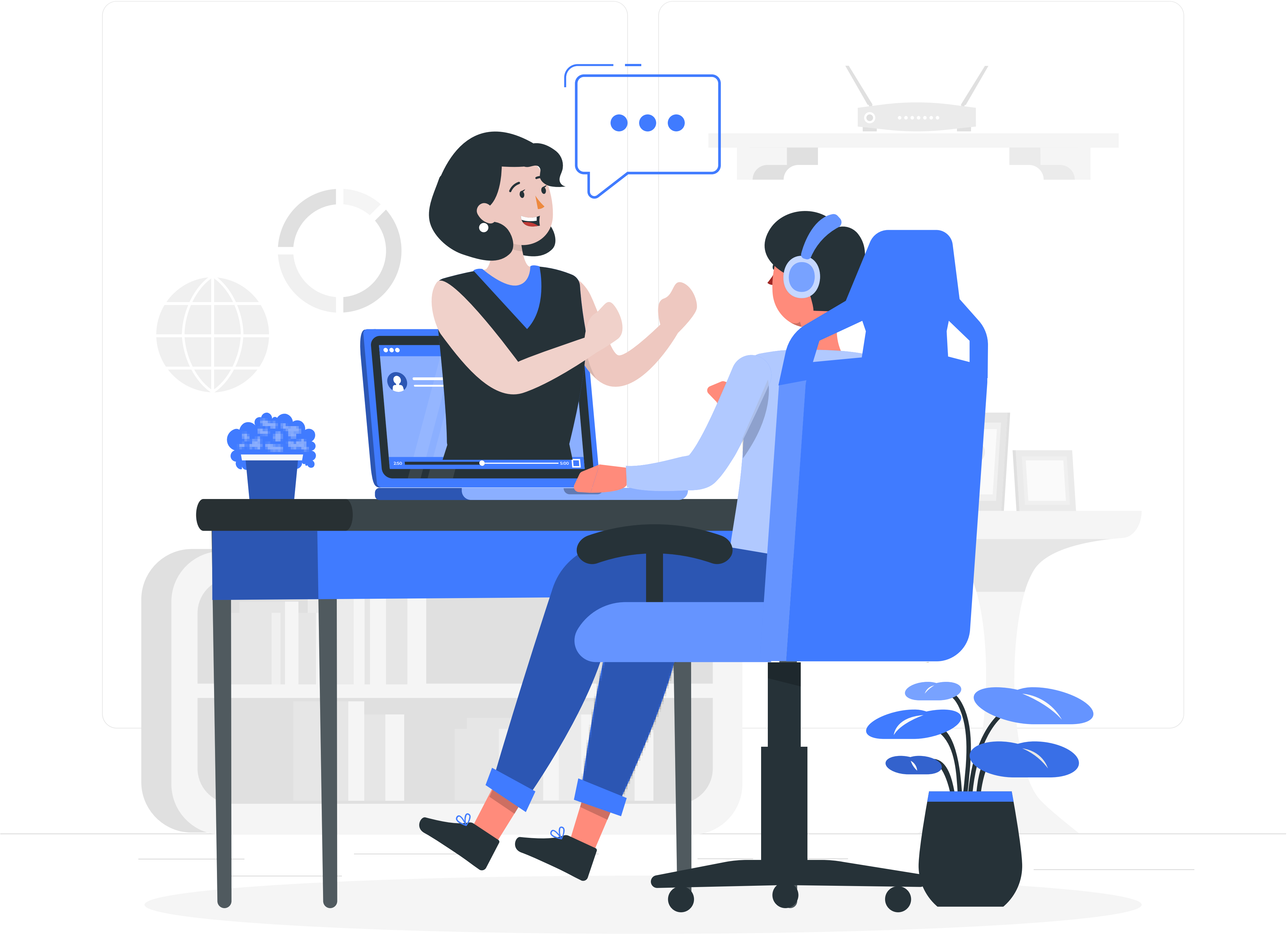
Week 6.2
May 6, 2021

Announcements

UML Class Diagram Q&A Session

Wednesday May 12, 2021

3:00 PM – 4:30 PM



Design Principles

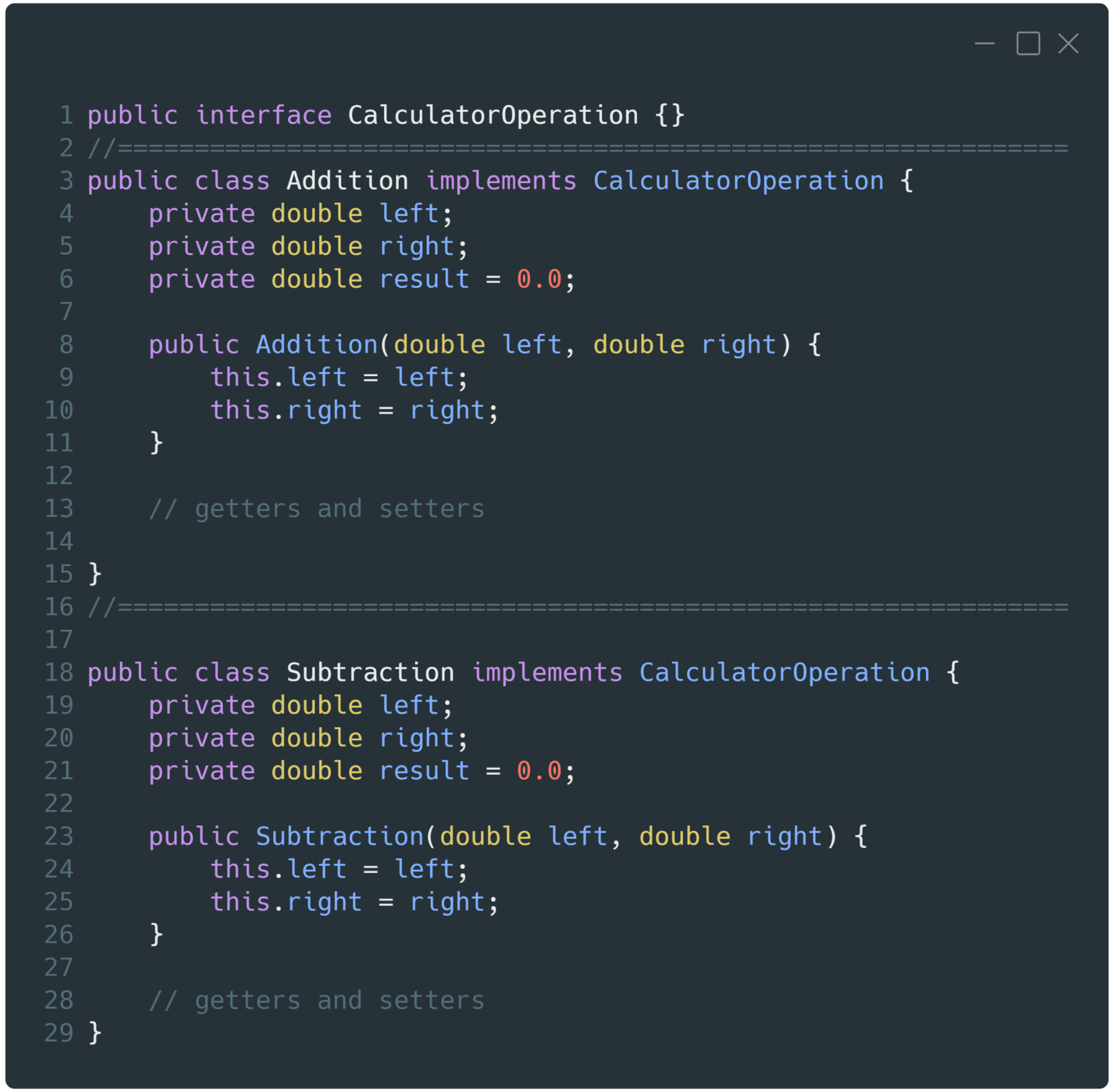
SOLID

Open-Closed Principle

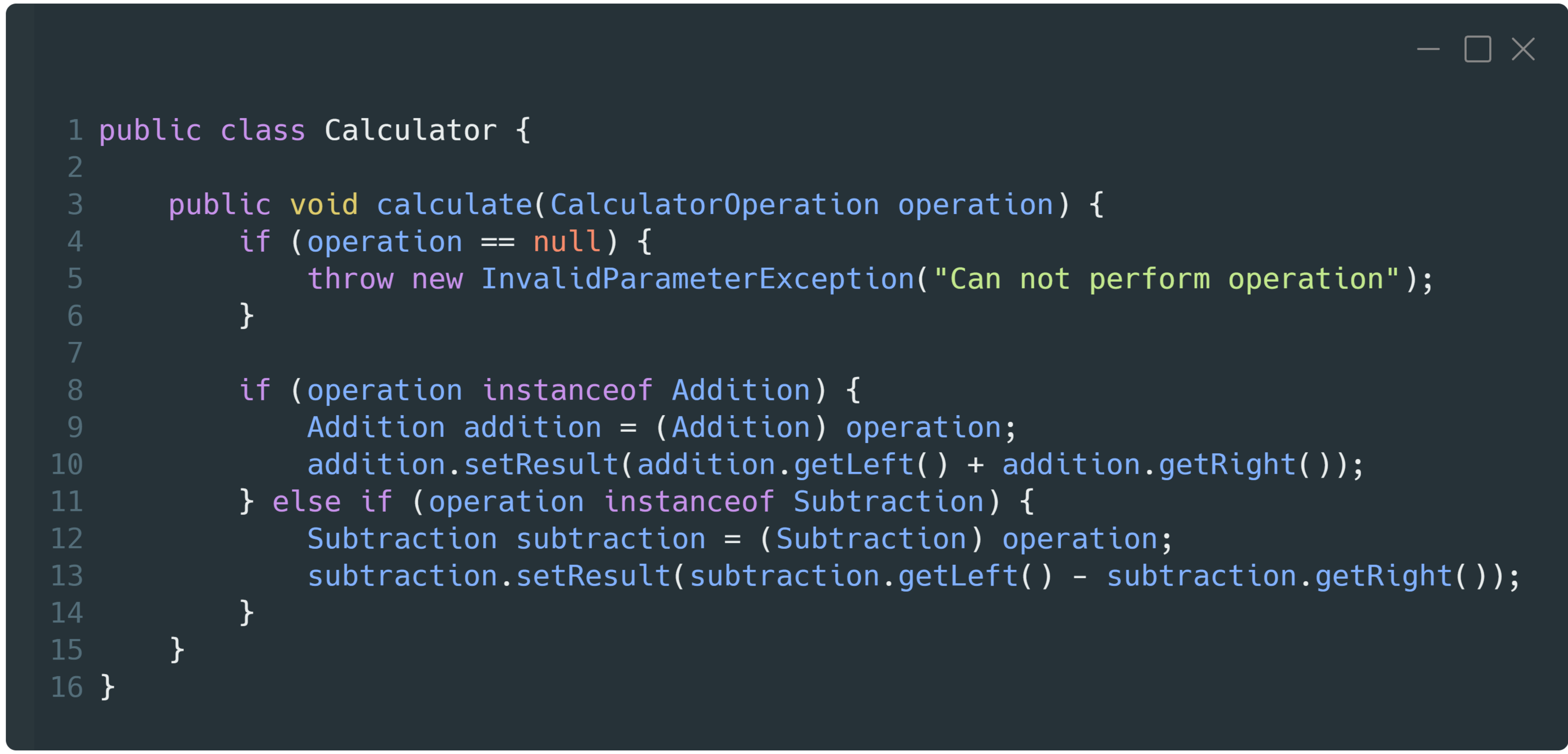
- **Component**
  - Can be anything from a single class to an entire program
- **Modification:**
  - Changing the code
- **Extension:**
  - Adding new functionality
- **OCP is usually done with the help of interfaces and abstract classes.**
- **How to make sure your code follows the Open/Closed Design Principle?**
  - Implementation inheritance
  - Interface inheritance

> **“Software components should be open for extension, but closed for modification”**

OCP Example: Calculator App Problem

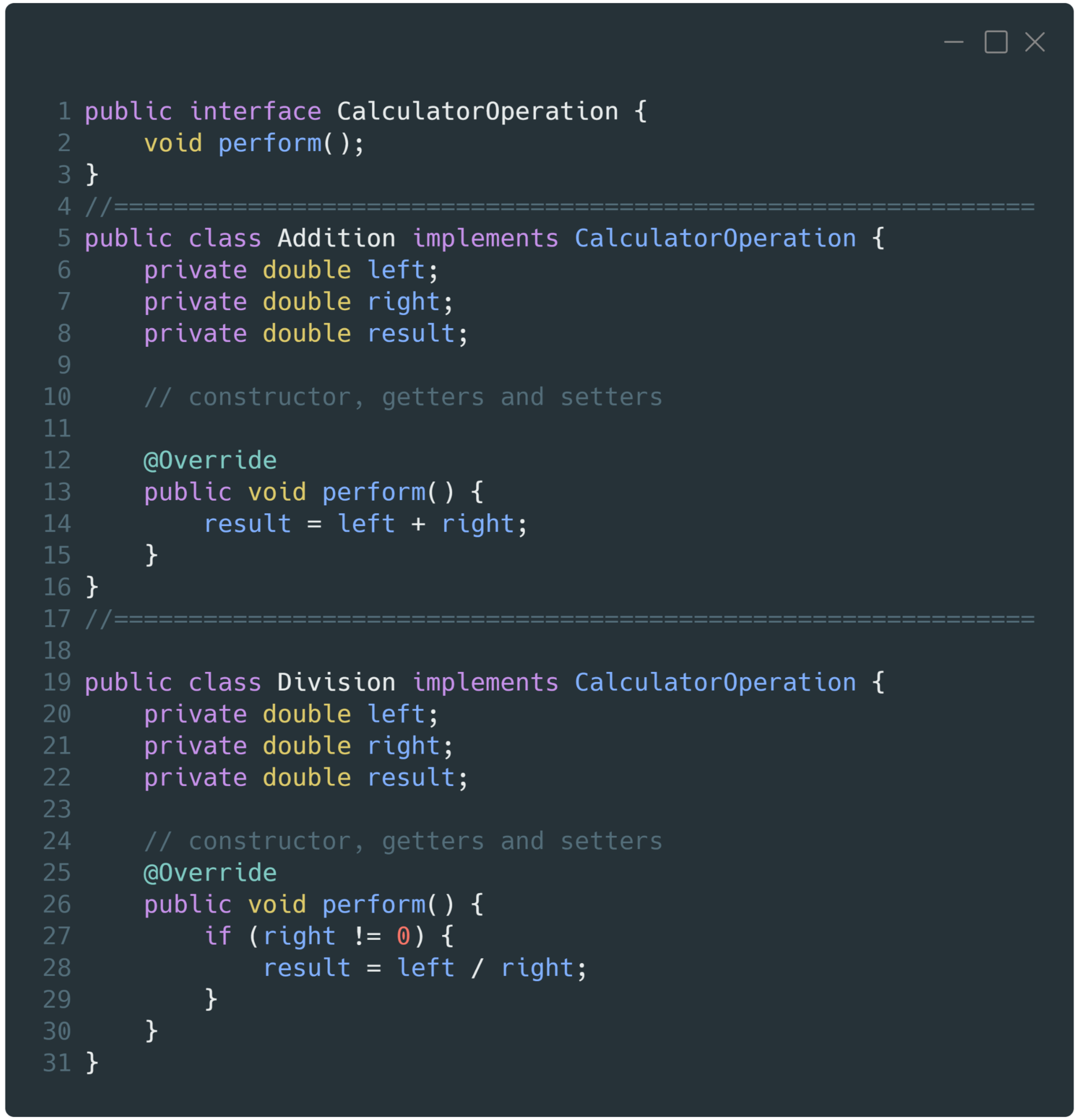


```java
public interface CalculatorOperation {}
//==============================================================
public class Addition implements CalculatorOperation {
    private double left;
    private double right;
    private double result = 0.0;

    public Addition(double left, double right) {
        this.left = left;
        this.right = right;
    }

    // getters and setters

}
//==============================================================

public class Subtraction implements CalculatorOperation {
    private double left;
    private double right;
    private double result = 0.0;

    public Subtraction(double left, double right) {
        this.left = left;
        this.right = right;
    }

    // getters and setters
}
```

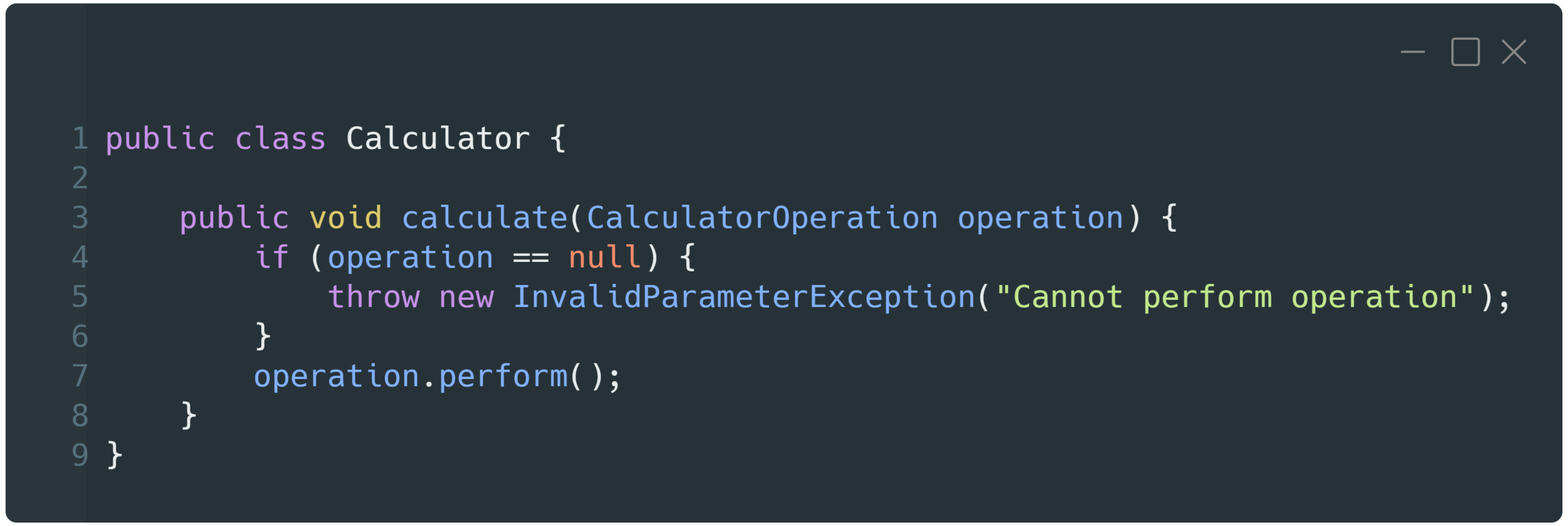


```java
public class Calculator {

    public void calculate(CalculatorOperation operation) {
        if (operation == null) {
            throw new InvalidParameterException("Can not perform operation");
        }

        if (operation instanceof Addition) {
            Addition addition = (Addition) operation;
            addition.setResult(addition.getLeft() + addition.getRight());
        } else if (operation instanceof Subtraction) {
            Subtraction subtraction = (Subtraction) operation;
            subtraction.setResult(subtraction.getLeft() - subtraction.getRight());
        }
    }
}
```

OCP Example: Calculator App Solution

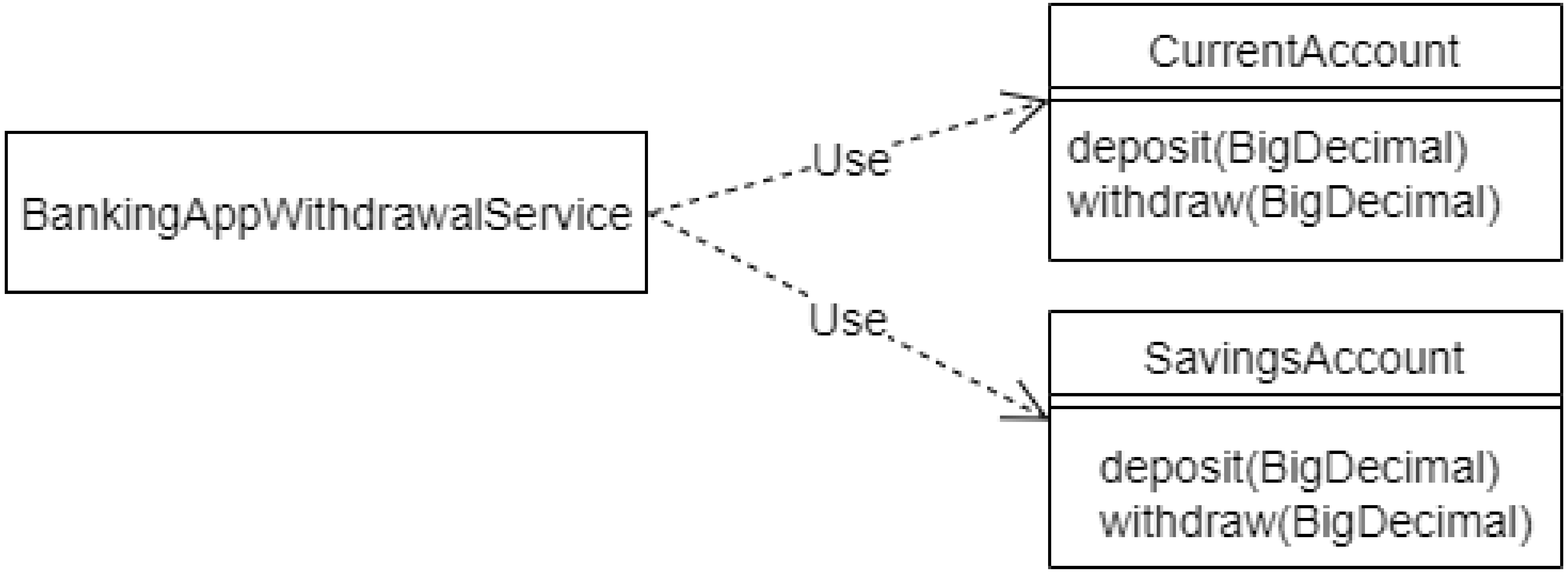


```java
public interface CalculatorOperation {
    void perform();
}
//=================================================================
public class Addition implements CalculatorOperation {
    private double left;
    private double right;
    private double result;

    // constructor, getters and setters

    @Override
    public void perform() {
        result = left + right;
    }
}
//=================================================================

public class Division implements CalculatorOperation {
    private double left;
    private double right;
    private double result;

    // constructor, getters and setters
    @Override
    public void perform() {
        if (right != 0) {
            result = left / right;
        }
    }
}
```

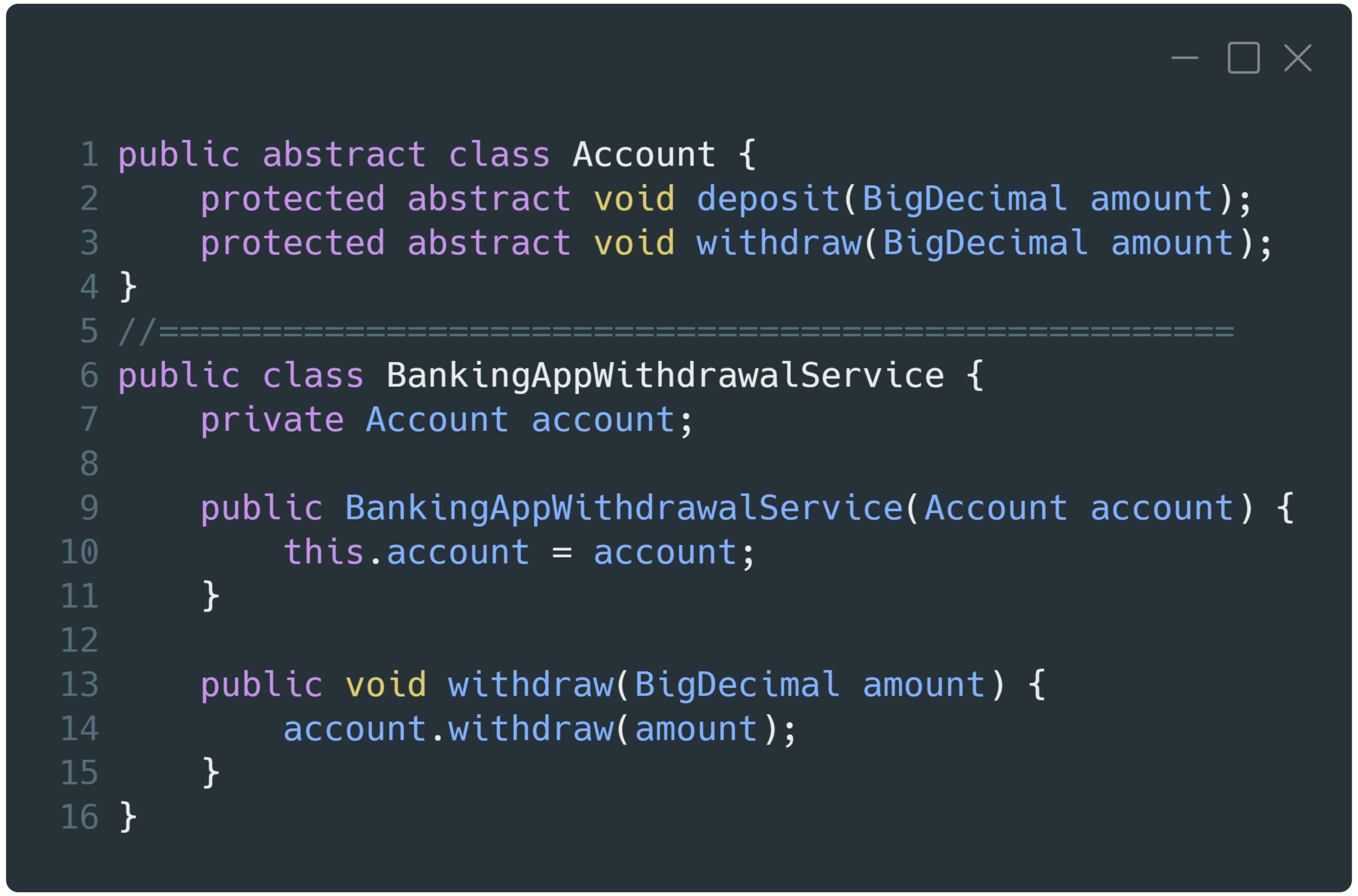


```java
public class Calculator {

    public void calculate(CalculatorOperation operation) {
        if (operation == null) {
            throw new InvalidParameterException("Cannot perform operation");
        }
        operation.perform();
    }
}
```

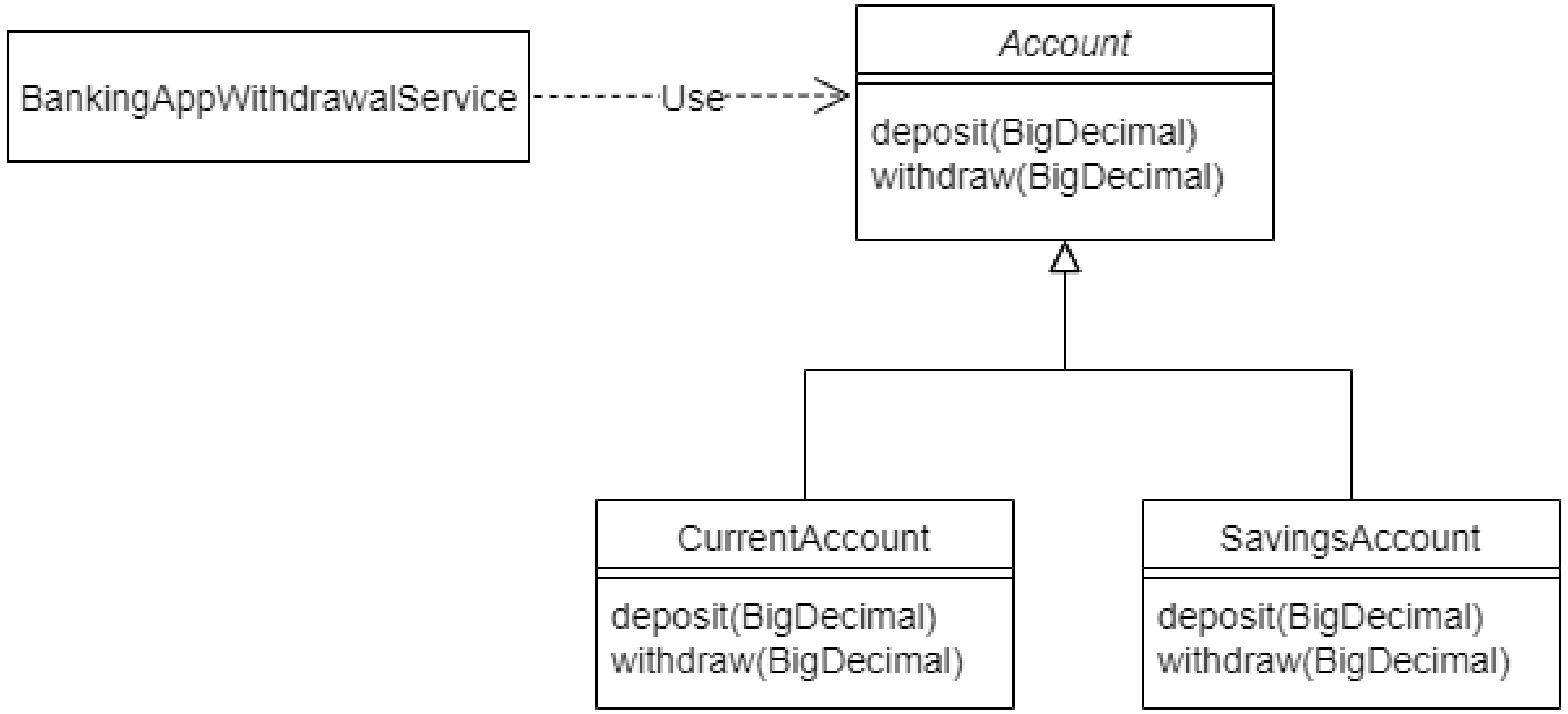
OCP Example: Banking System Problem



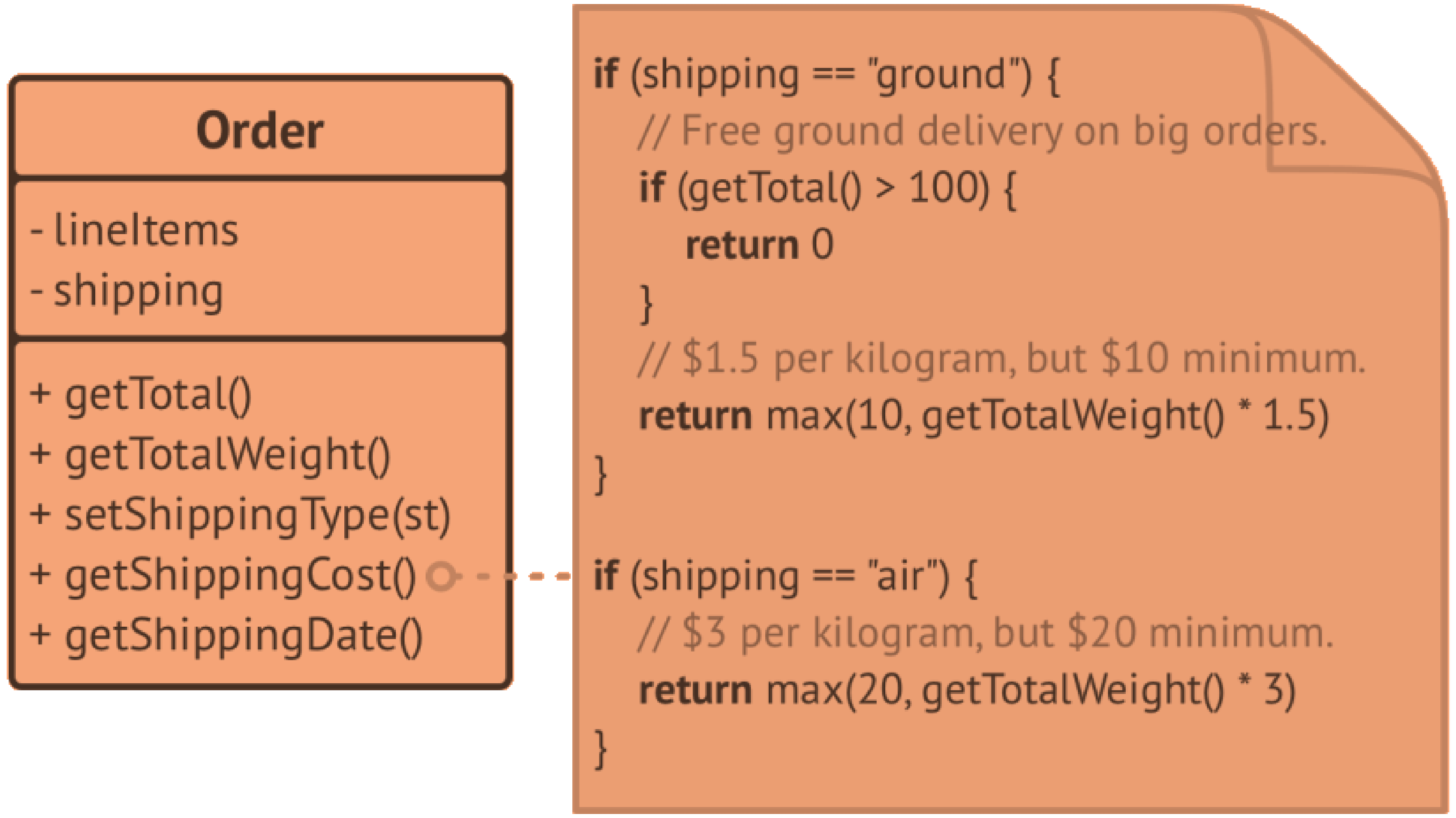
OCP Example: Banking System Solution



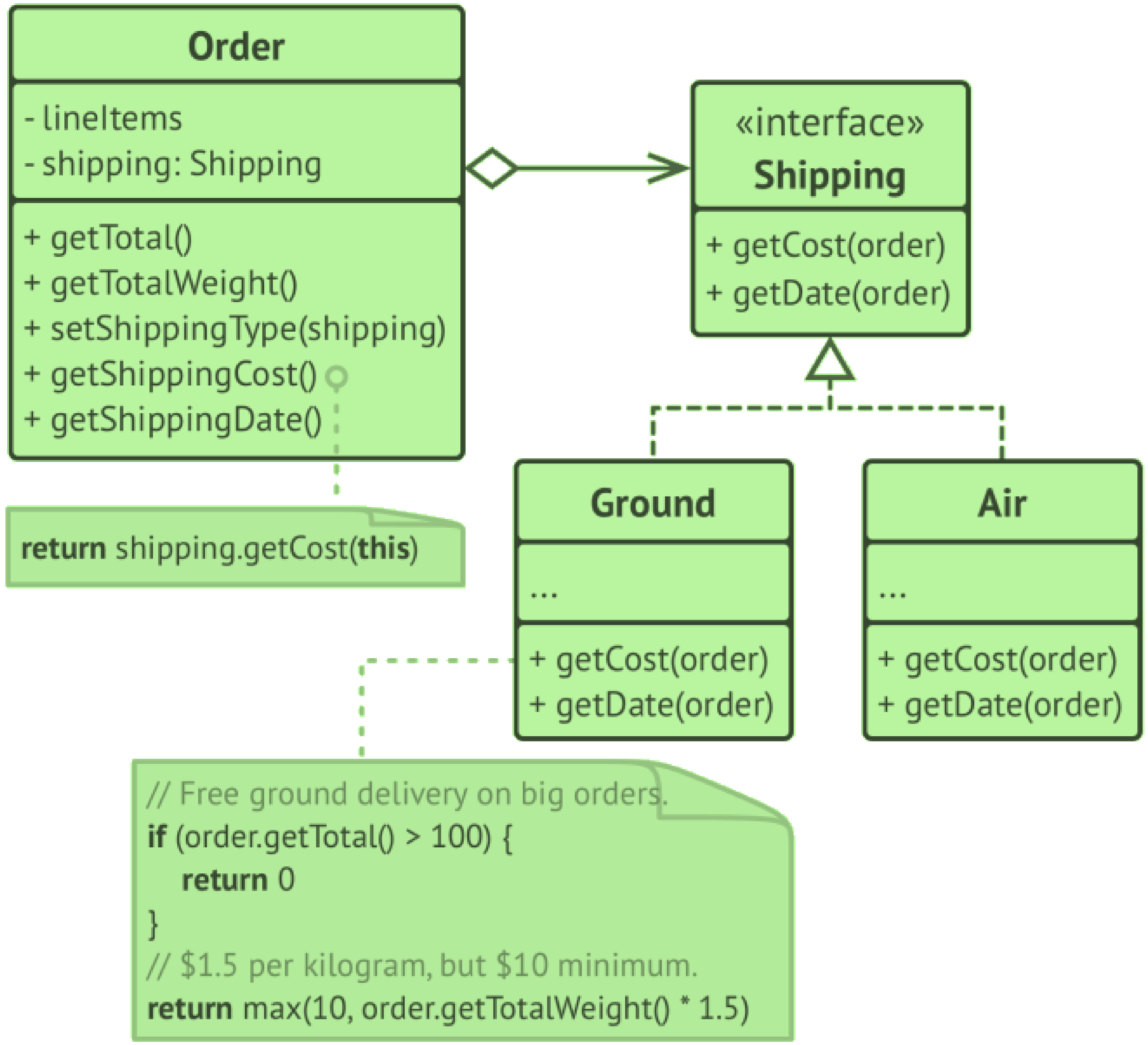
```java
public abstract class Account {
    protected abstract void deposit(BigDecimal amount);
    protected abstract void withdraw(BigDecimal amount);
}
//====================================================
public class BankingAppWithdrawalService {
    private Account account;

    public BankingAppWithdrawalService(Account account) {
        this.account = account;
    }

    public void withdraw(BigDecimal amount) {
        account.withdraw(amount);
    }
}
```



OCP Example: eCommerce Application



```
if (shipping == "ground") {
    // Free ground delivery on big orders.
    if (getTotal() > 100) {
        return 0
    }
    // $1.5 per kilogram, but $10 minimum.
    return max(10, getTotalWeight() * 1.5)
}

if (shipping == "air") {
    // $3 per kilogram, but $20 minimum.
    return max(20, getTotalWeight() * 3)
}
```



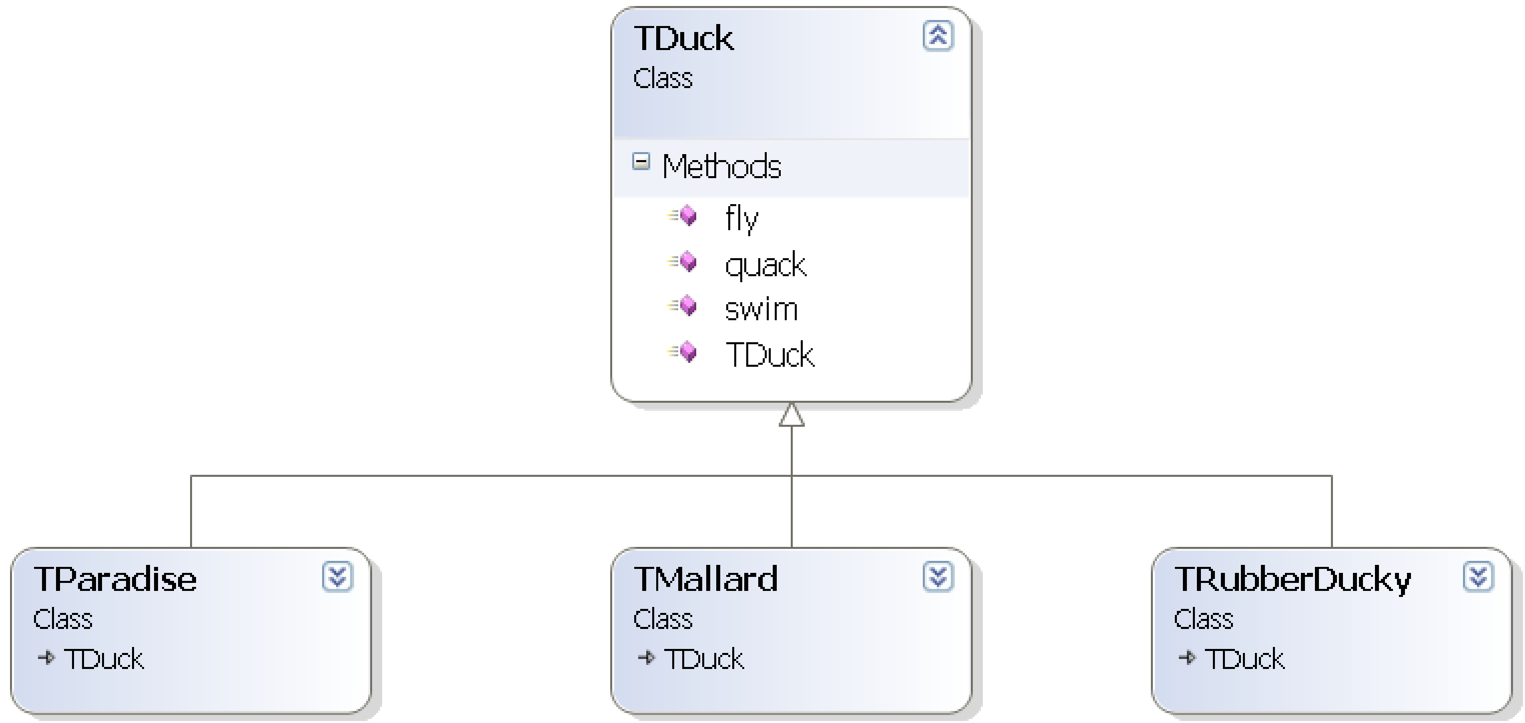
```
return shipping.getCost(this)
```

```
// Free ground delivery on big orders.
if (order.getTotal() > 100) {
    return 0
}
// $1.5 per kilogram, but $10 minimum.
return max(10, order.getTotalWeight() * 1.5)
```

Liskov Substitution Principle

- **Barbara Liskov - 1988**
  - If for each object o1 of type S there is an object o2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o1 is substituted for o2 then S is a subtype of T.
- **Design by contract – Bertrand Meyer**
  - helps us conform to the "is-a" relationship
- **Implementations of the same interface should never give a different result.**
- **To be substitutable, the subtype must behave like its supertype.**
- **Example Scenario: Using file system for testing purposes before developing database structure.**
- **How to make sure your code follows LSP?**
  - mindful programming

> **“Derived types must be completely substitutable for their base types”**

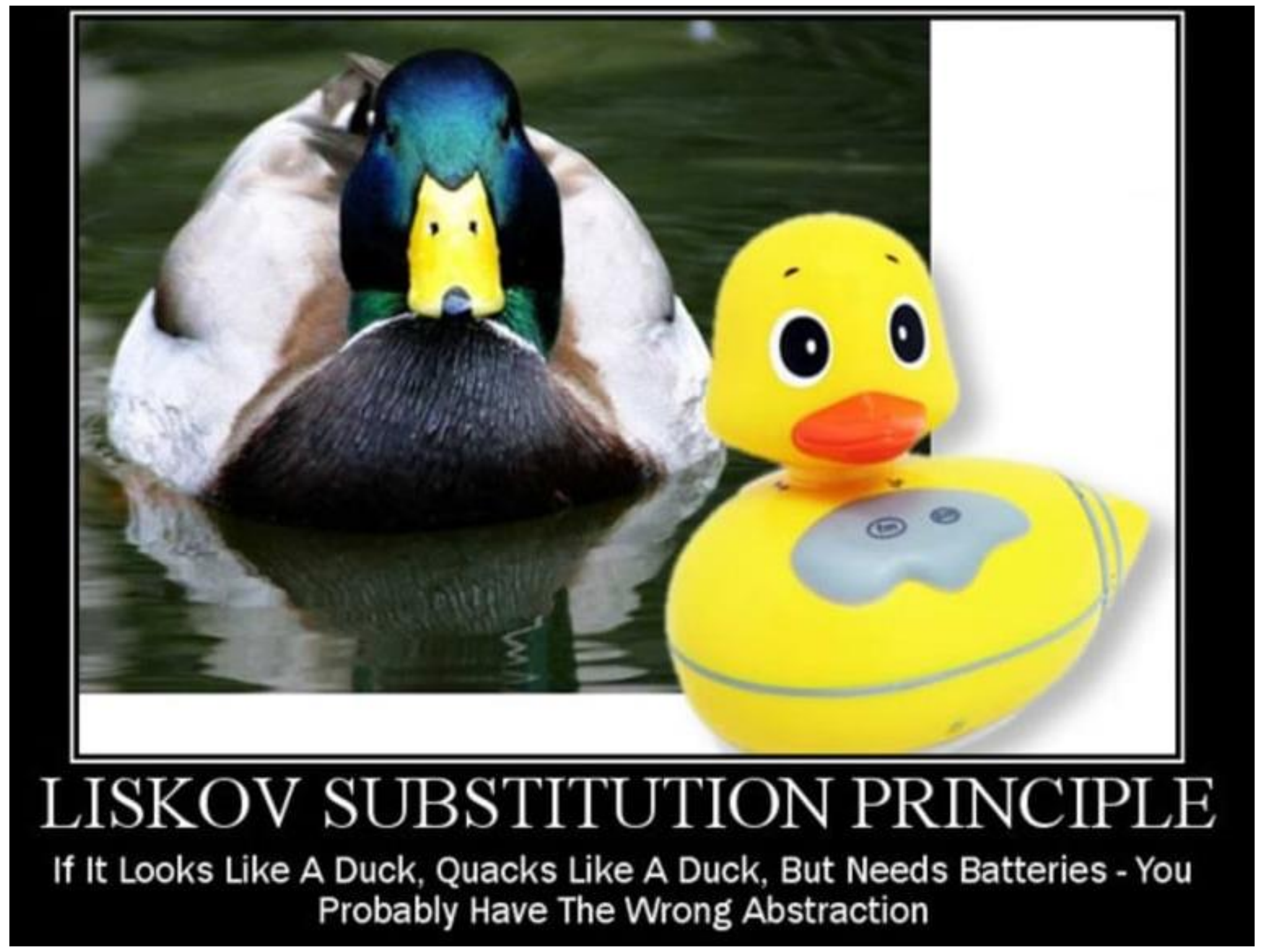


TDuck
Class
Methods
fly
quack
swim
TDuck

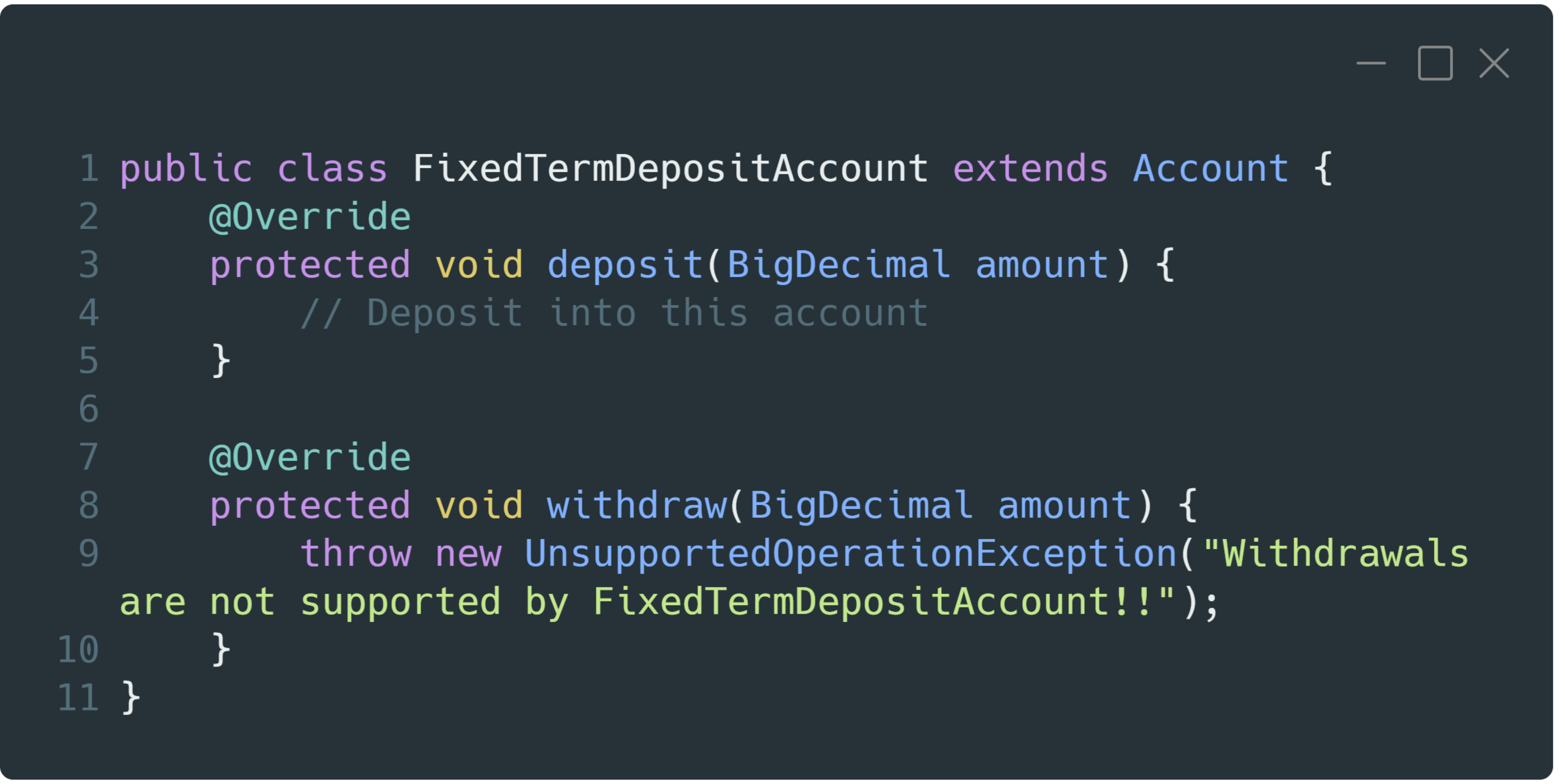
TParadise
Class
→ TDuck

TMallard
Class
→ TDuck

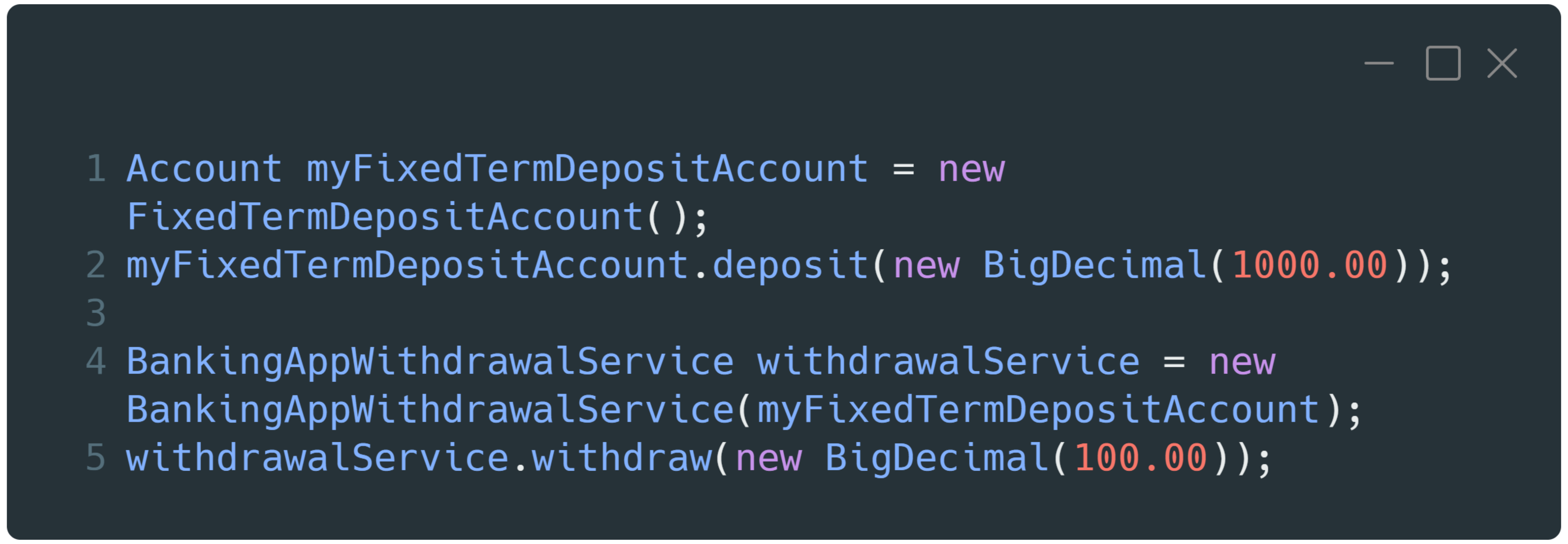
TRubberDucky
Class
→ TDuck


LISKOV SUBSTITUTION PRINCIPLE
If It Looks Like A Duck, Quacks Like A Duck, But Needs Batteries - You Probably Have The Wrong Abstraction

LSP Example: Banking System Problem

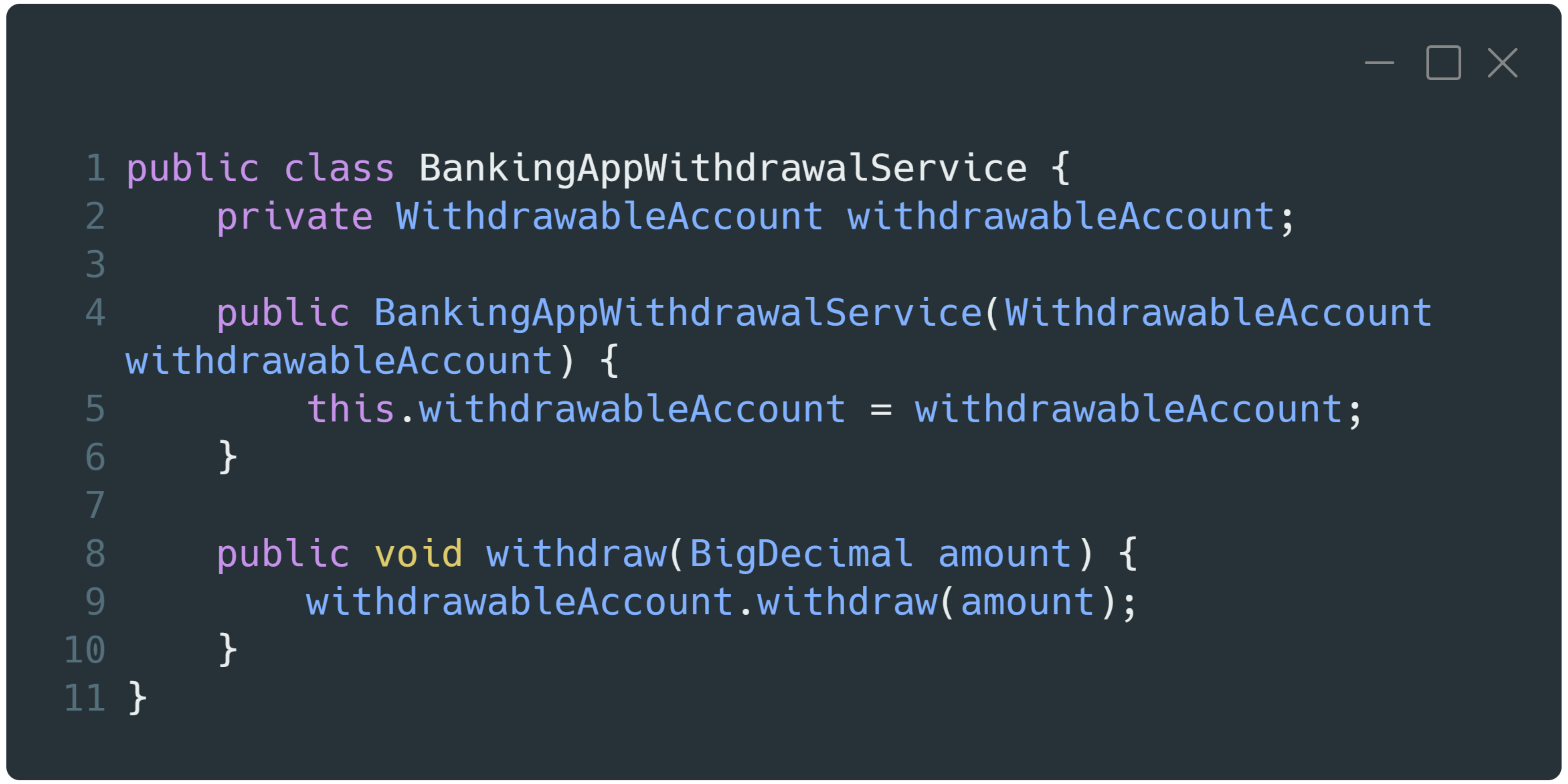


```java
public class FixedTermDepositAccount extends Account {
    @Override
    protected void deposit(BigDecimal amount) {
        // Deposit into this account
    }

    @Override
    protected void withdraw(BigDecimal amount) {
        throw new UnsupportedOperationException("Withdrawals are not supported by FixedTermDepositAccount!!");
    }
}
```

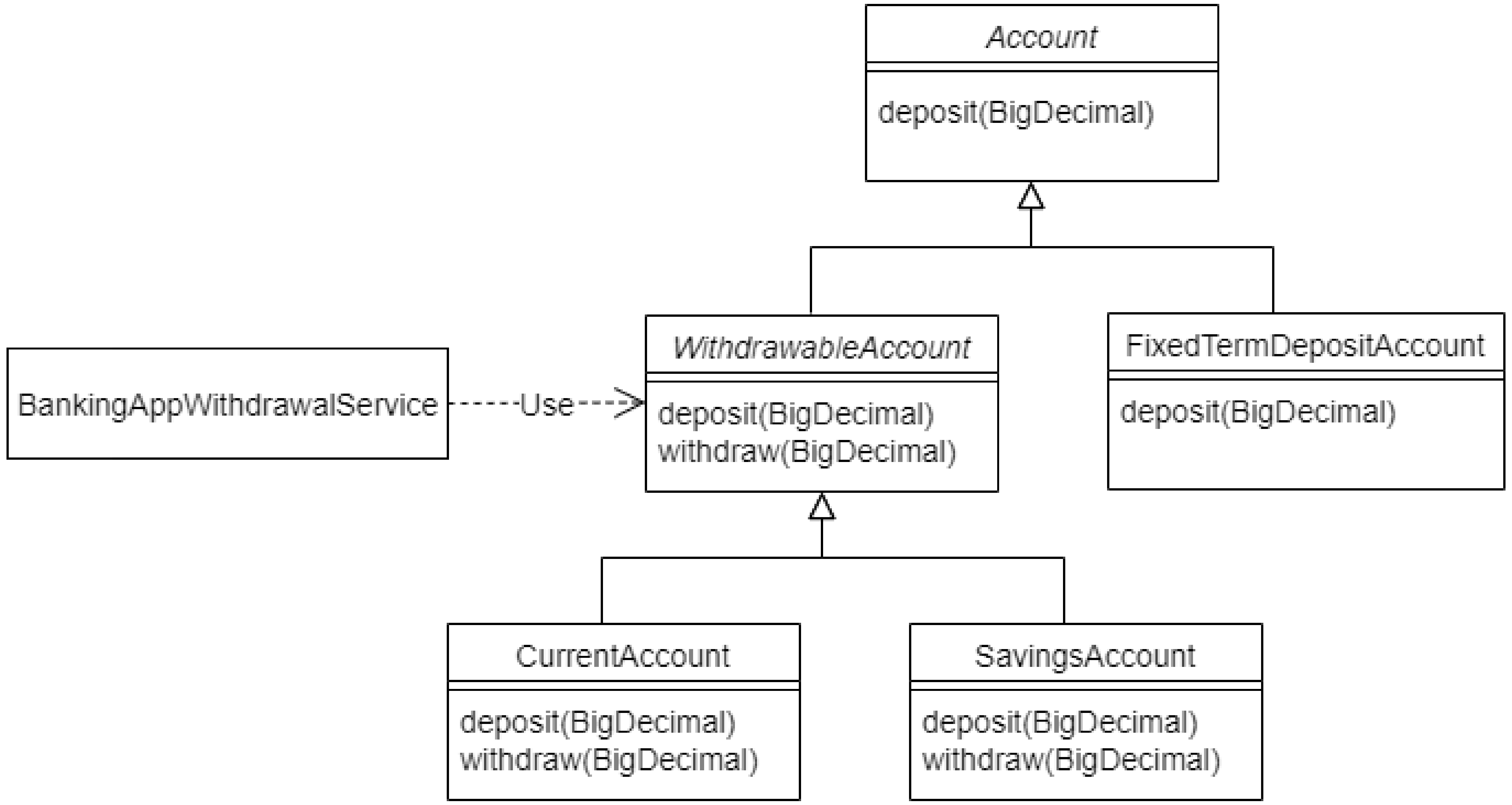


```java
Account myFixedTermDepositAccount = new FixedTermDepositAccount();
myFixedTermDepositAccount.deposit(new BigDecimal(1000.00));

BankingAppWithdrawalService withdrawalService = new BankingAppWithdrawalService(myFixedTermDepositAccount);
withdrawalService.withdraw(new BigDecimal(100.00));
```

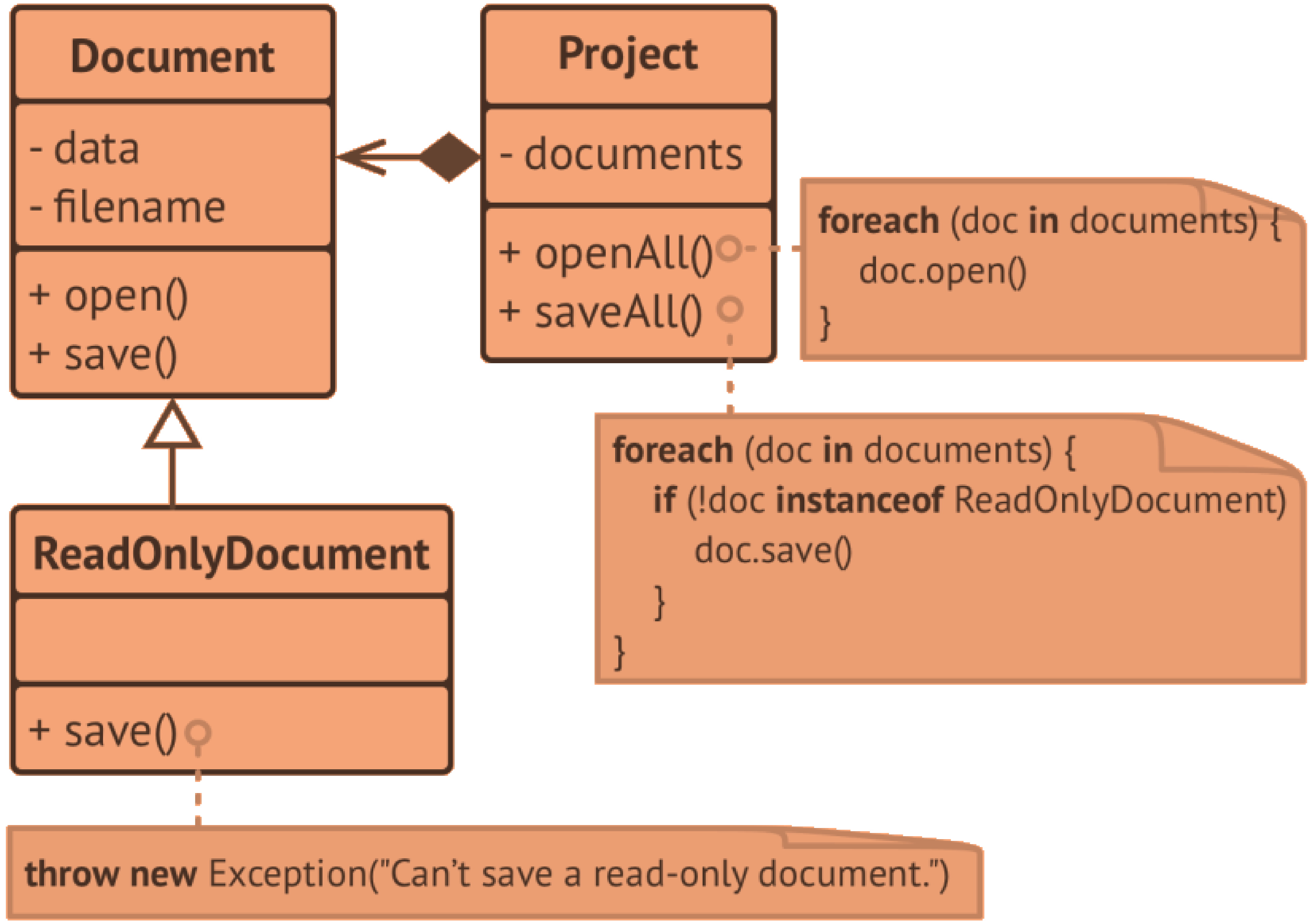
LSP Example: Banking System Solution



```java
public class BankingAppWithdrawalService {
    private WithdrawableAccount withdrawableAccount;

    public BankingAppWithdrawalService(WithdrawableAccount withdrawableAccount) {
        this.withdrawableAccount = withdrawableAccount;
    }

    public void withdraw(BigDecimal amount) {
        withdrawableAccount.withdraw(amount);
    }
}
```



LSP Example: Saving Documents



Document

- data
- filename

+ open()
+ save()

Project

- documents

+ openAll()
+ saveAll()

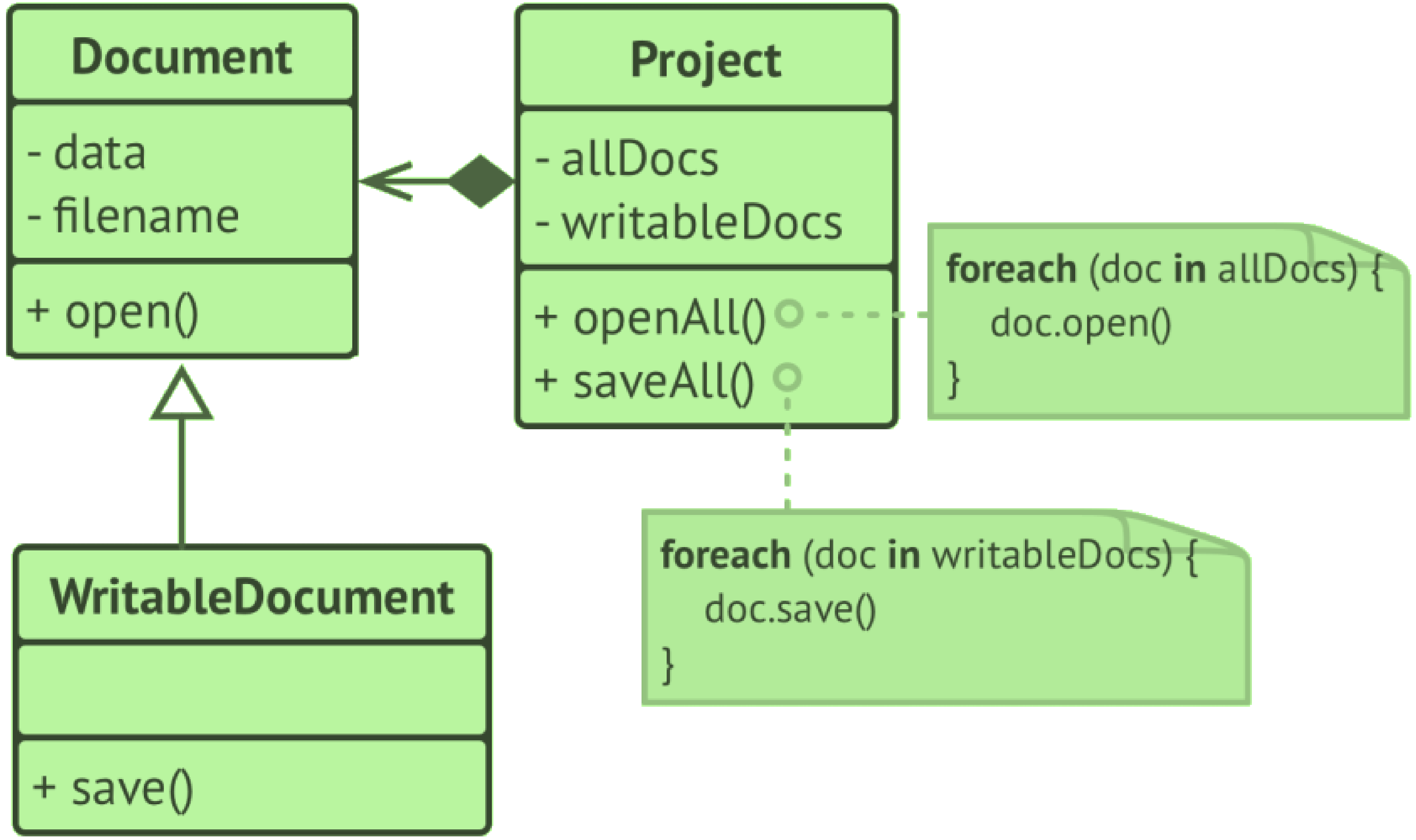
```
foreach (doc in documents) {
    doc.open()
}
```

```
foreach (doc in documents) {
    if (!doc instanceof ReadOnlyDocument)
        doc.save()
    }
}
```

ReadOnlyDocument

+ save()

```
throw new Exception("Can't save a read-only document.")
```



Document

- data
- filename

+ open()

Project

- allDocs
- writableDocs

+ openAll()
+ saveAll()

```
foreach (doc in allDocs) {
    doc.open()
}
```

```
foreach (doc in writableDocs) {
    doc.save()
}
```

WritableDocument

+ save()

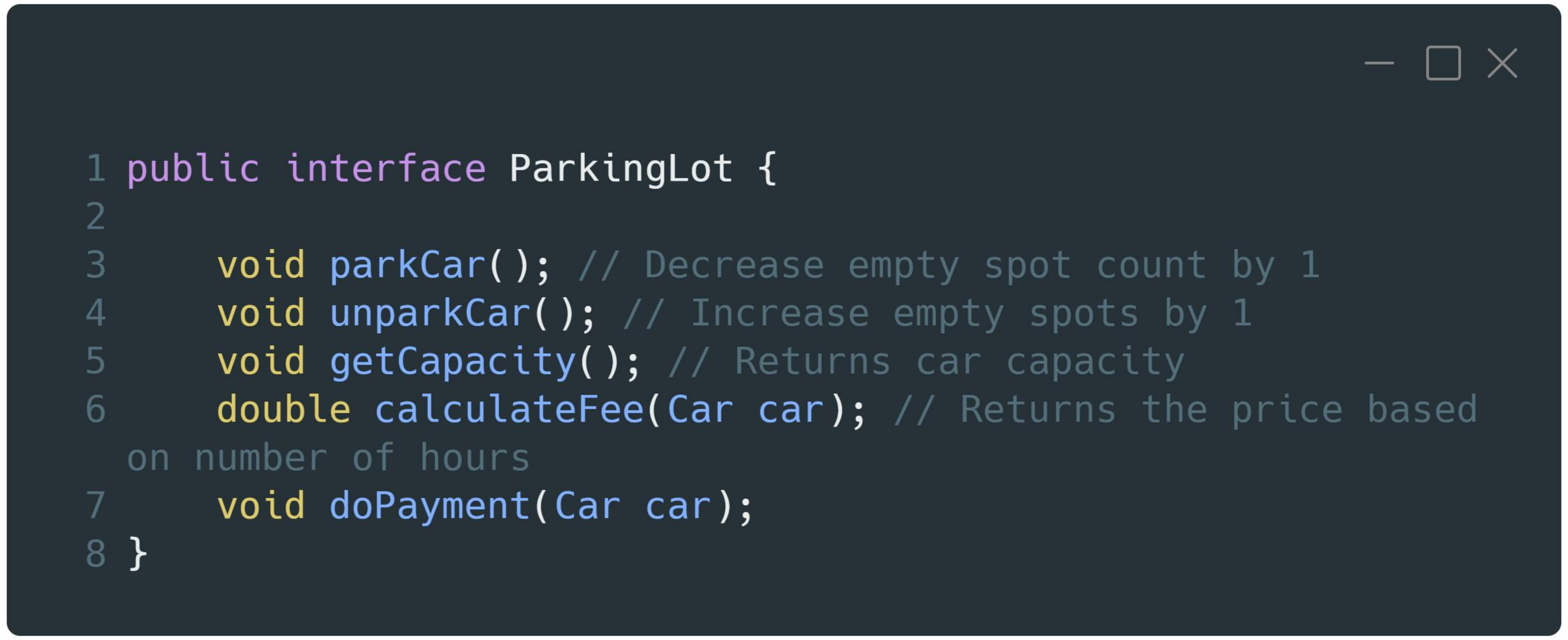
Interface Segregation Principle

- **Many client-specific interfaces are better than one general-purpose interface.**
- **Reduce the side effects of using larger interfaces by breaking application interfaces into smaller ones.**
- **Similar to SRP:**
  - Each class or interface serves a single purpose.
- **It take more time and effort in the design phase .**
- **It increase the code complexity.**
- **It leads to flexible code.**
- **Examples:**
  - logging interface for writing and reading logs – DB vs Console
  - Reportable interface: generateExcel() and generatedPdf().
  - Large Employee class:
    - EmployeeTimeLogController, EmployeeTimeOffController, EmployeeSalaryController
- **How to make sure your code follows the ISP?**

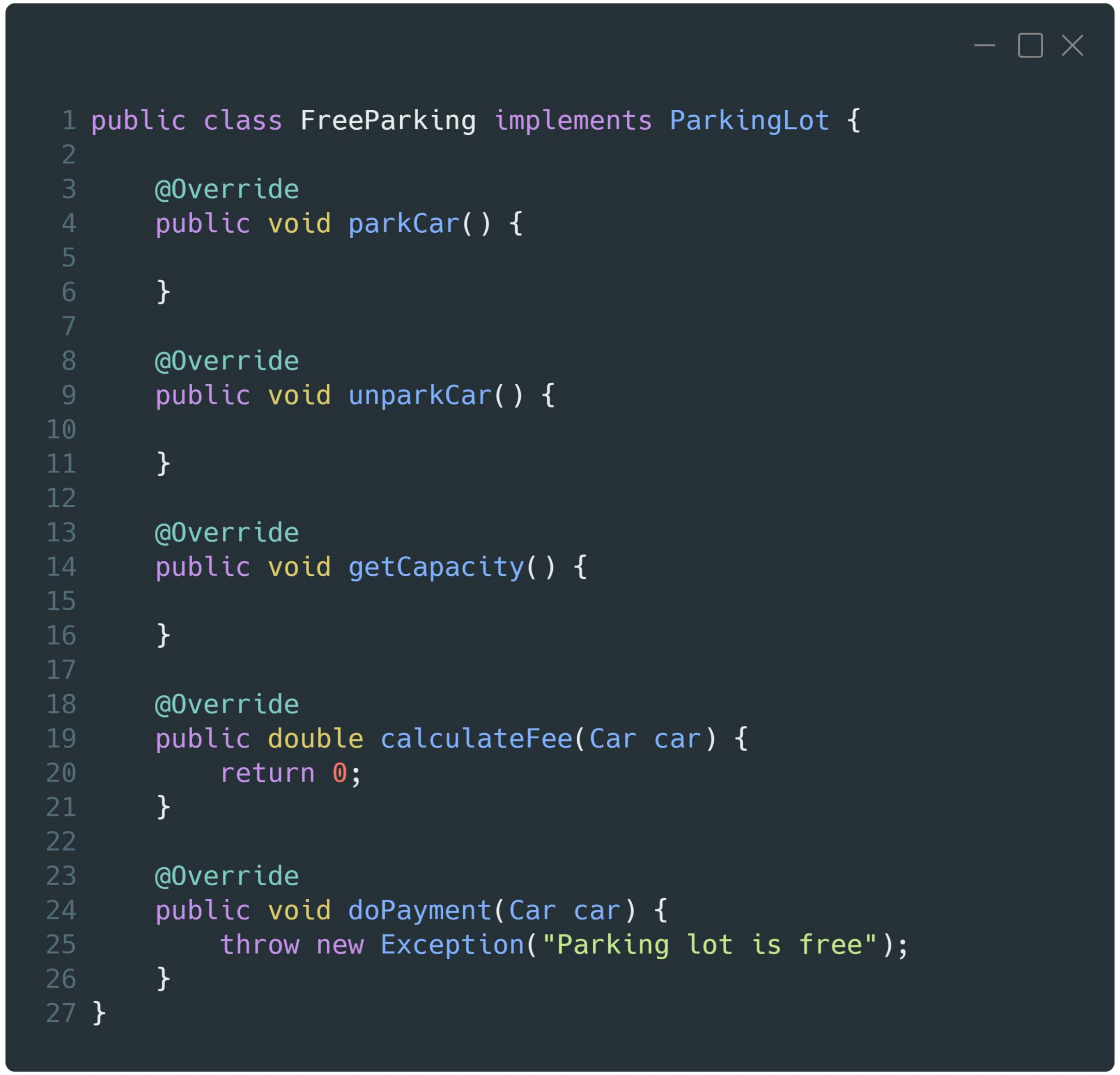
> **“Clients should not be forced to implement unnecessary methods which they will not use”**

INTERFACE SEGREGATION PRINCIPLE
You Want Me To Plug This In, Where?

ISP Example: ParkingLot App Problem

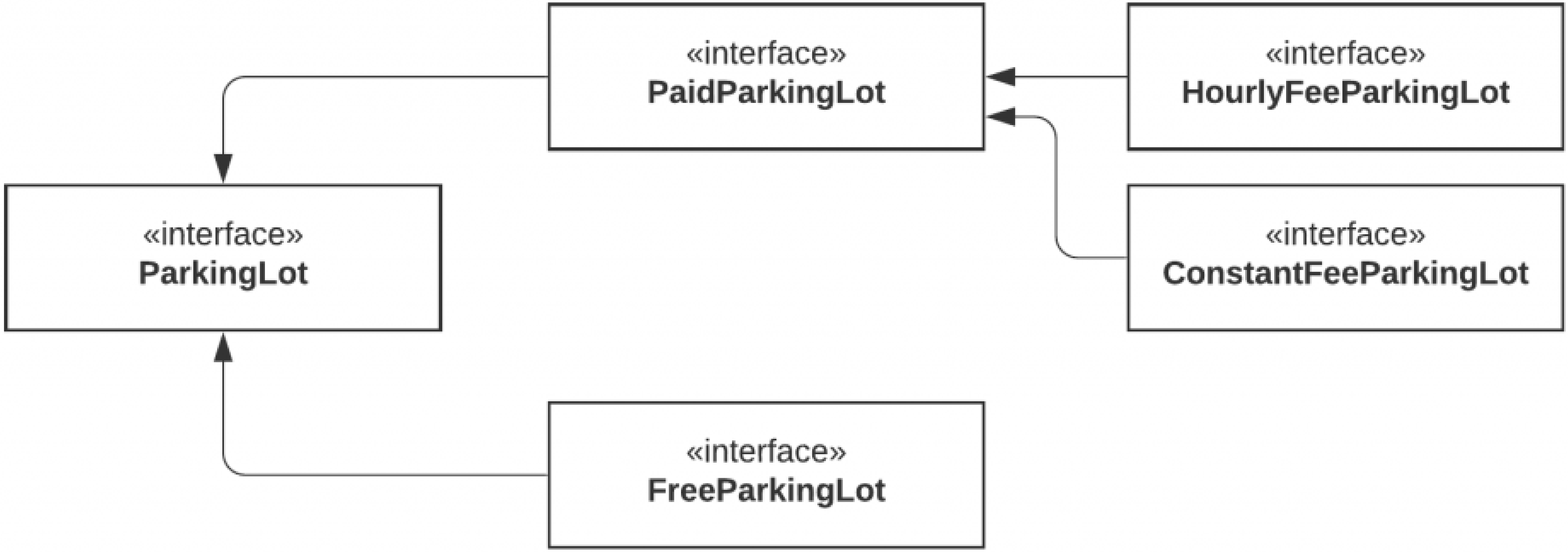


```java
public interface ParkingLot {

    void parkCar(); // Decrease empty spot count by 1
    void unparkCar(); // Increase empty spots by 1
    void getCapacity(); // Returns car capacity
    double calculateFee(Car car); // Returns the price based on number of hours
    void doPayment(Car car);
}
```



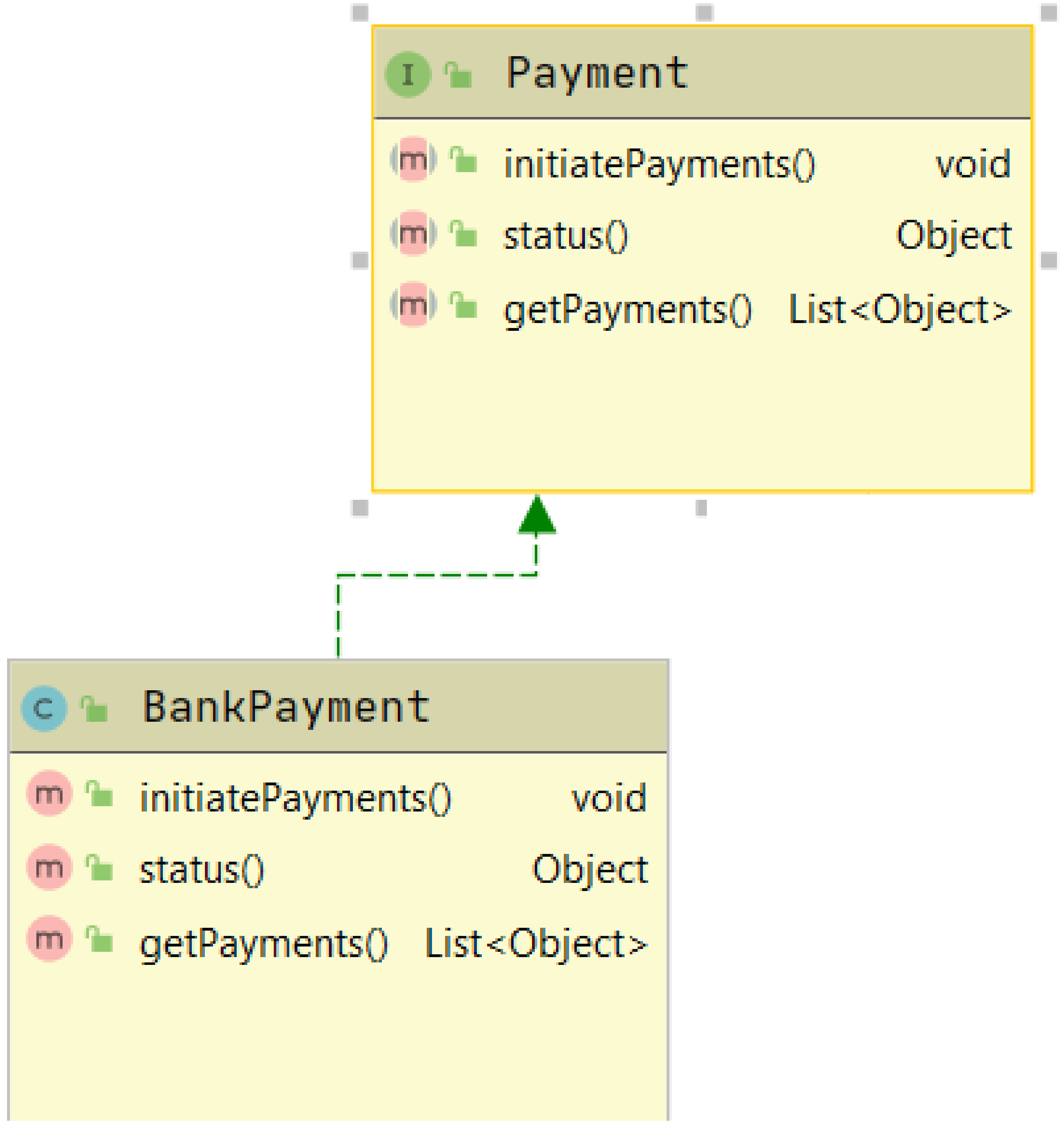
```java
public class FreeParking implements ParkingLot {

    @Override
    public void parkCar() {

    }

    @Override
    public void unparkCar() {

    }

    @Override
    public void getCapacity() {

    }

    @Override
    public double calculateFee(Car car) {
        return 0;
    }

    @Override
    public void doPayment(Car car) {
        throw new Exception("Parking lot is free");
    }
}
```

ISP Example: ParkingLot App Solution



ISP Example: Payment App

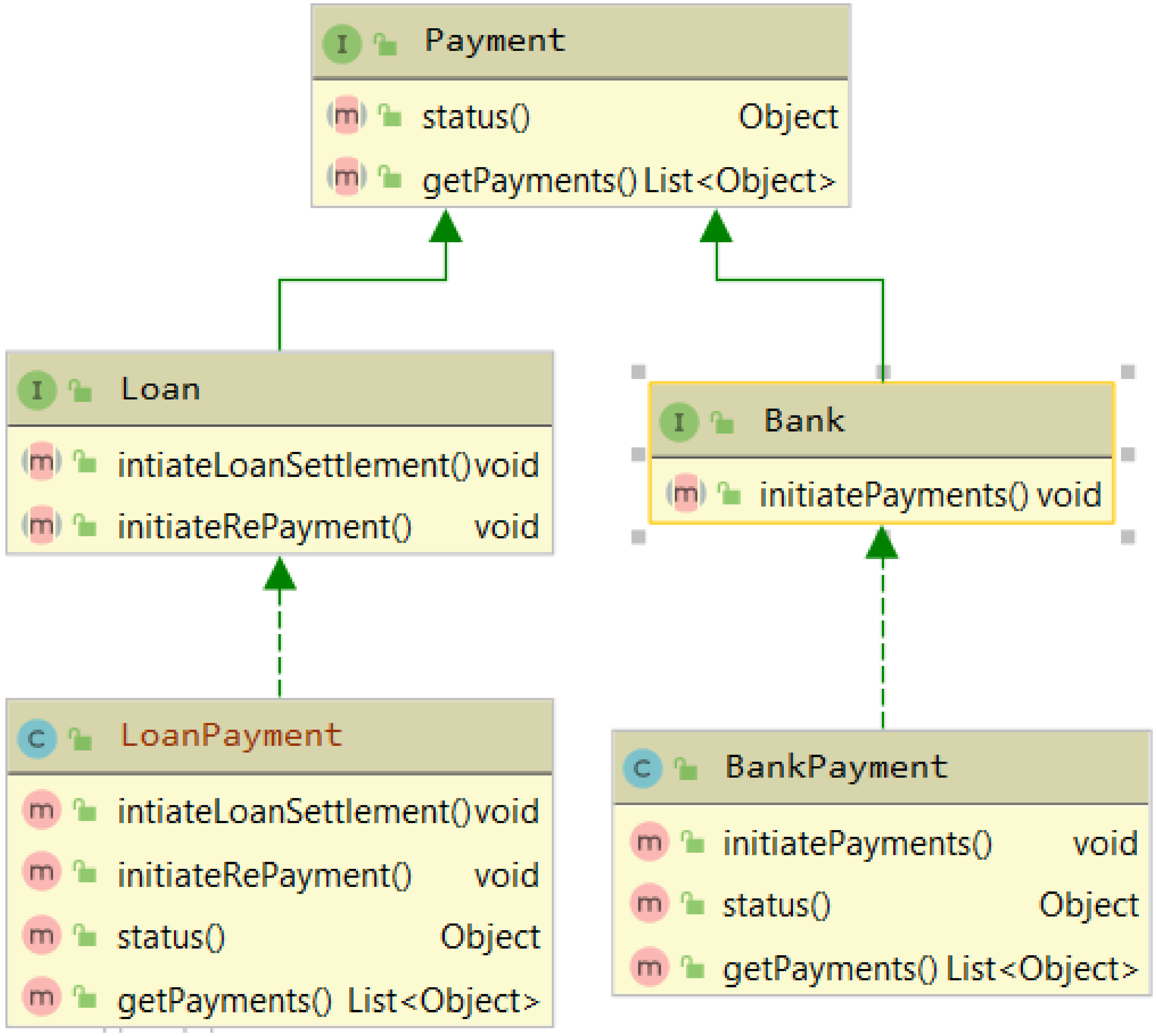
ISP Violation



Payment
initiatePayments() void
status() Object
getPayments() List<Object>

BankPayment
initiatePayments() void
status() Object
getPayments() List<Object>

ISP Compliance



Payment
status() Object
getPayments() List<Object>

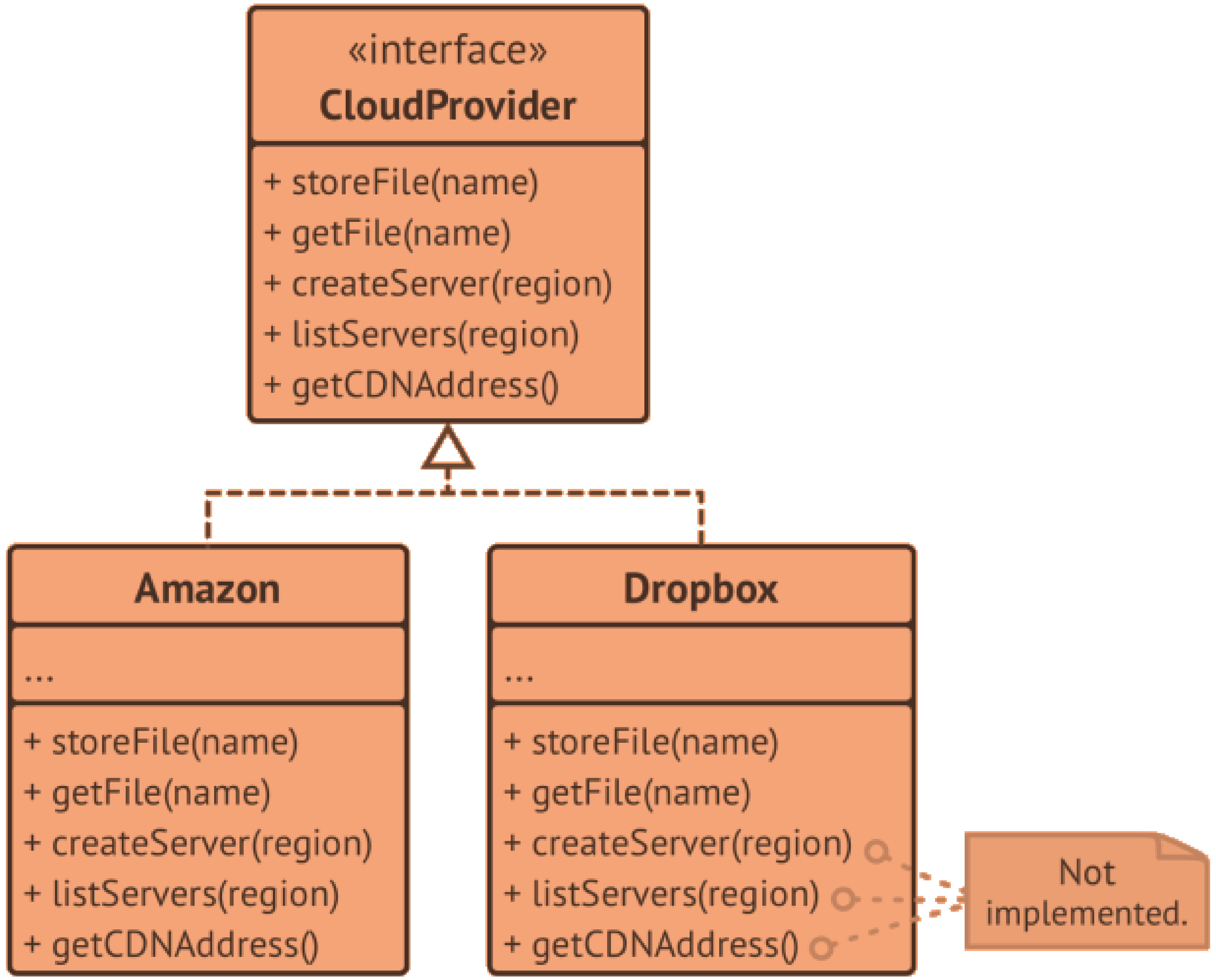
Loan
intiateLoanSettlement() void
initiateRePayment() void

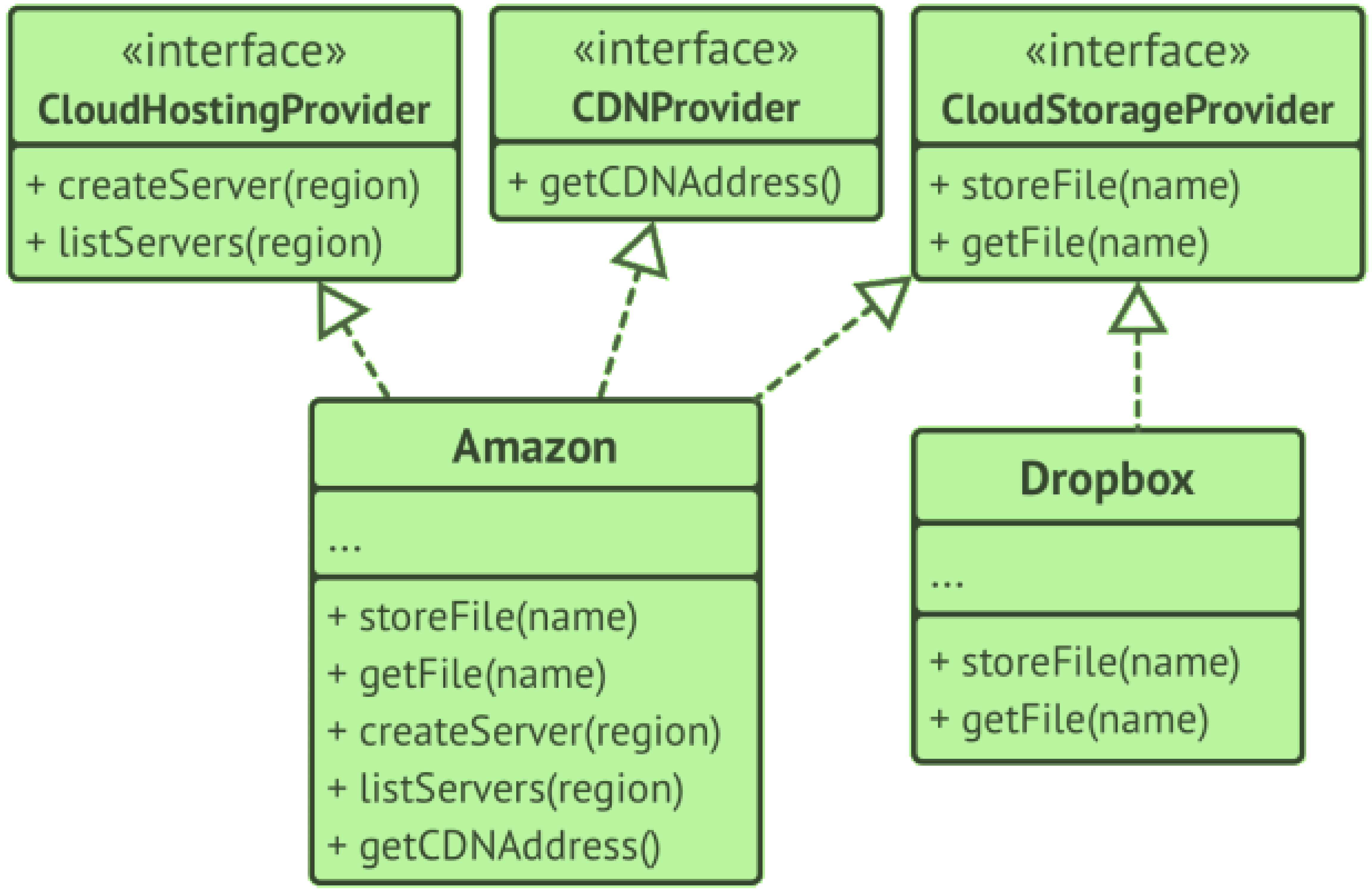
Bank
initiatePayments() void

LoanPayment
intiateLoanSettlement() void
initiateRePayment() void
status() Object
getPayments() List<Object>

BankPayment
initiatePayments() void
status() Object
getPayments() List<Object>

ISP Example: Cloud Provider

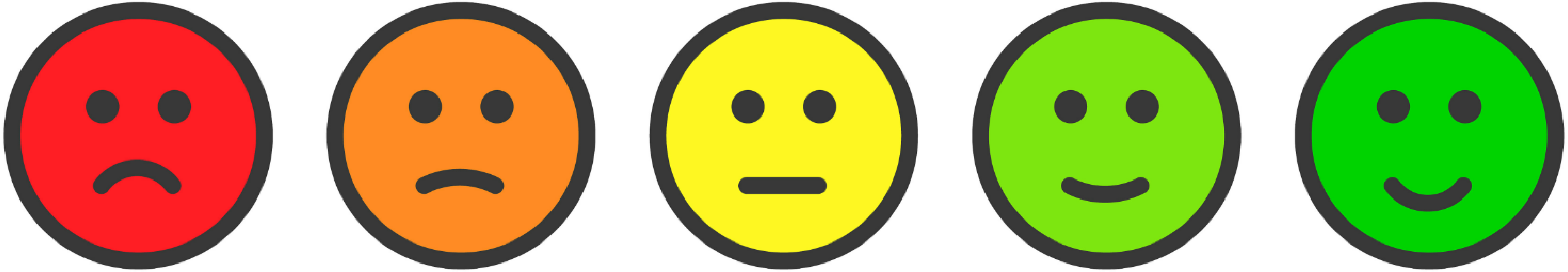




Any Question

????????????????

How do you feel about the course?



Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Please Send Your Question or Feedback...

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**