ULTIMATE DESIGN PATTERNS

MAHMOUD YOUSSEF

CEO & FOUNDER of Eqraatech - اقرأ-تِك
Software Engineer at Booking.com

Social Media Links:

https://www.linkedin.com/in/mahmoud-youssef97/
https://twitter.com/M7moudYoussef97

COURSE OBJECTVE

Reusable, Extensible and Maintainable

Object Oriented Software Design

PREREQUISITES

- 3~6+ months programming experience
- Familiarity with object-oriented programming

DESIGN PATTERNS

CREATIONAL

- Factory Method
- Abstract Factory
- Singleton
- Prototype
- Builder

STRUCTURAL

- Bridge
- Composite
- Decorator
- Facade
- Adapter
- Flyweight
- Proxy

BEHAVIORAL

- Command
- Observer
- State
- Strategy
- Visitor
- Mediator
- Iterator
- Memento
- Chain of Responsibility
- Template Method

2.
Think about solution

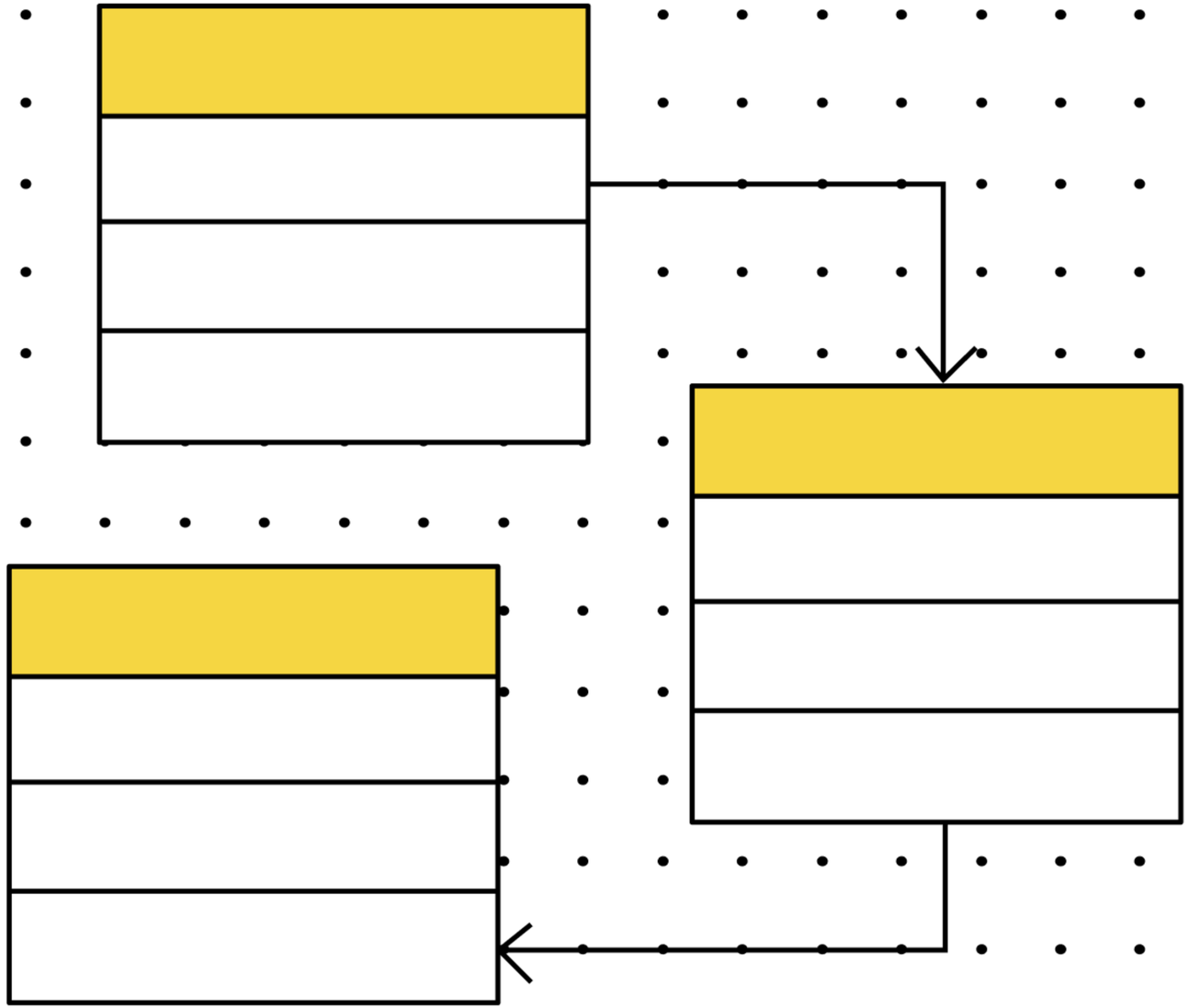
1.
Define the problem

3.
Make your analysis

4.
Enhance it!

FigJam

UML Diagram



COURSE OBJECTVE

Reusable, Extensible and Maintainable

Object Oriented Software Design

MAHMOUD YOUSSEF

CEO & FOUNDER of Eqraatech - اقرأ-تِك
Software Engineer at Booking.com

Social Media Links:

https://www.linkedin.com/in/mahmoud-youssef97/
https://twitter.com/M7moudYoussef97

What Are Design Patterns?

Design Patterns

- Design patterns are ***typical solutions*** to commonly occurring problems in software design.
- It's more of ***general concept*** for solving problems rather than a piece of code.
- Design patterns are a more ***high-level description*** of a solution.

It's a BLUEPRINT

Why Design Patterns?

Tried and ***tested solutions***!

Define ***common language***!

Classification of Design Patterns

CREATIONAL

- Factory Method
- Abstract Factory
- Singleton
- Prototype
- Builder

STRUCTURAL

- Bridge
- Composite
- Decorator
- Facade
- Adapter
- Flyweight
- Proxy

BEHAVIORAL

- Command
- Observer
- State
- Strategy
- Visitor
- Mediator
- Iterator
- Memento
- Chain of Responsibility
- Template Method

COURSE OBJECTIVE

Reusable, Extensible and Maintainable

Object Oriented Software Design

How to Take This Course?

How to Take This Course ?

No Ducks and Pizzas!

© Mahmoud Youssef

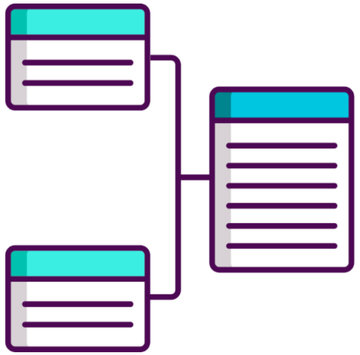
NEW CHALLENGES

© Mahmoud Youssef

COURSE STRUCTURE

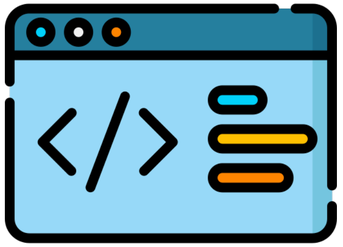
Identify the ***Task***

+



Design the ***UML***

+



Hands on ***Coding***

+

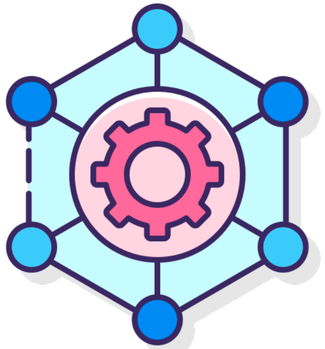
Discussion and ***Enhancements***

© Mahmoud Youssef

COURSE STRUCTURE

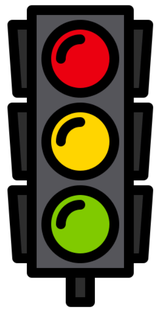
Reading ***Resources***

+



Libraries and ***Frameworks***

+



Real World ***Analogy***

© Mahmoud Youssef

FAQs

Is it mandatory to use them?

FAQs

Is there only one valid pattern for each problem?

COURSE OBJECTIVE

Reusable, Extensible and Maintainable

Object Oriented Software Design

© Mahmoud Youssef

Unified Model Language (UML)

UML CLASS DIAGRAM

Provide a ***standard*** way to ***visualize*** the design of a system!

Student

Attributes

- studentId: int
- name: String
- age: int

Operations

+ displayStudentInfo()

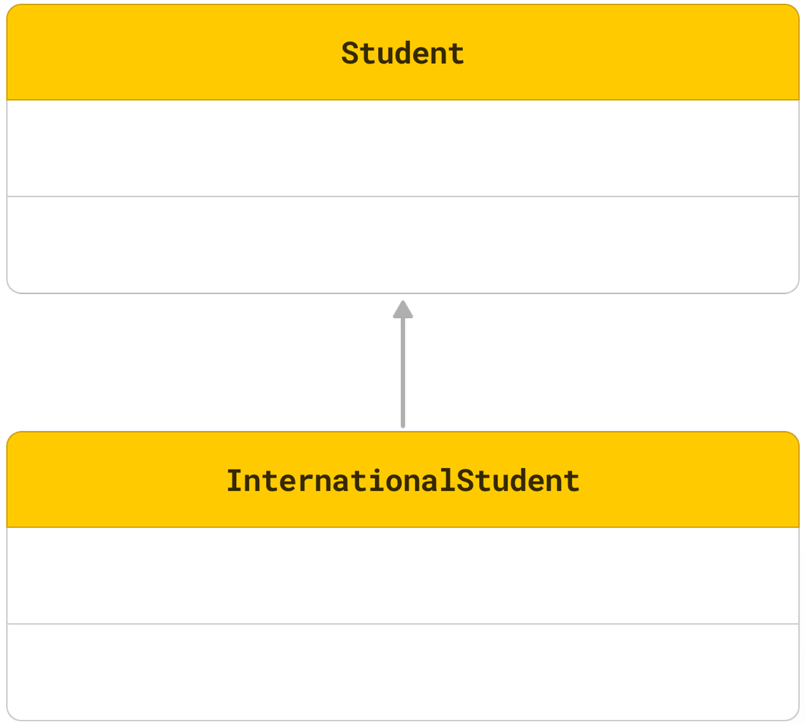
Student

```
public class Student {

    private int studentId;
    private String name;
    private int age;

    public void displayStudentInfo() {
    }
}
```

© Mahmoud Youssef

INHERITANCE

IS-A Relationship

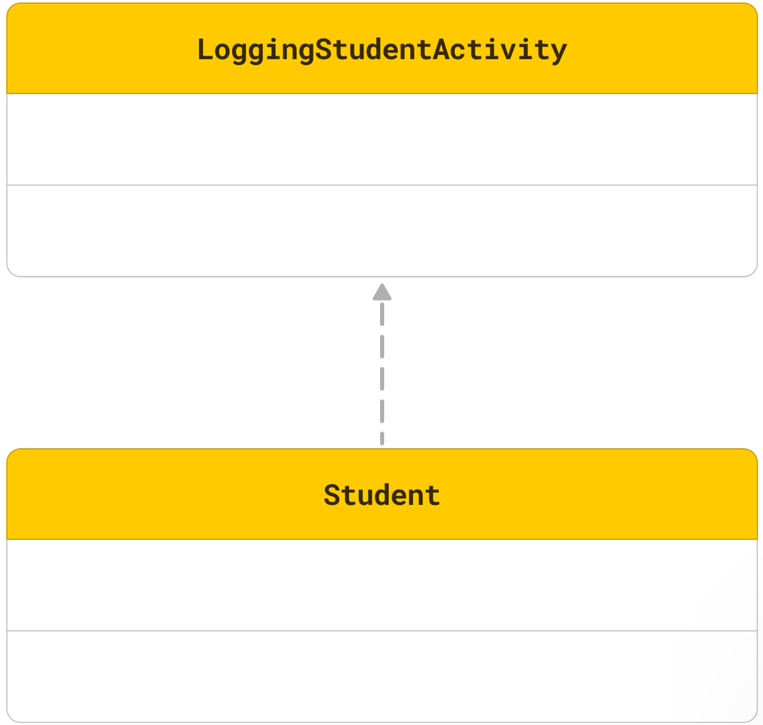


Student

```
public class InternationalStudent extends Student {

    private String countryOfOrigin;
    private String visaStatus;
}
```

© Mahmoud Youssef

IMPLEMENTATION



Student

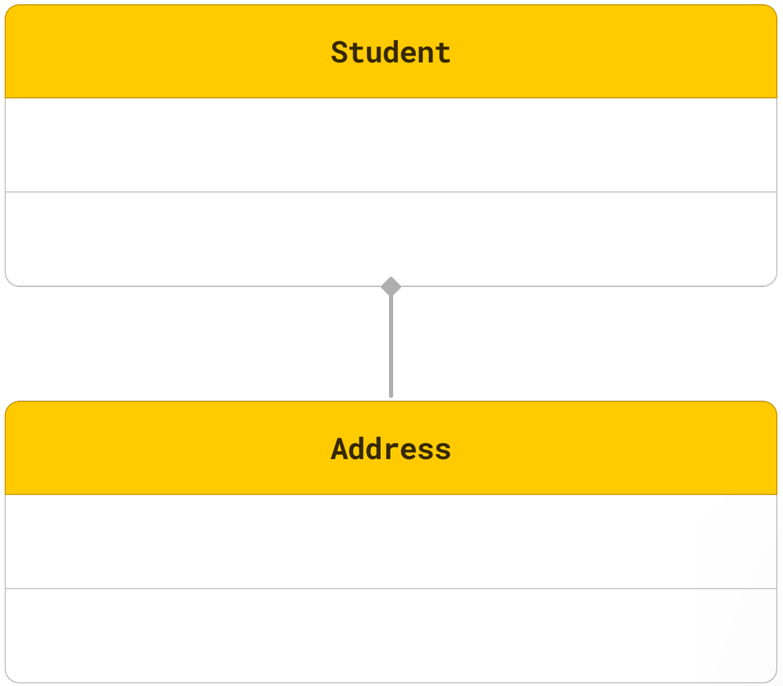
```
public class Student implements LoggingStudentActivity {

    @Override
    public void logStudentActivity() {
    }
}
```

© Mahmoud Youssef

COMPOSITION

Has-A Relationship

Each student has an address

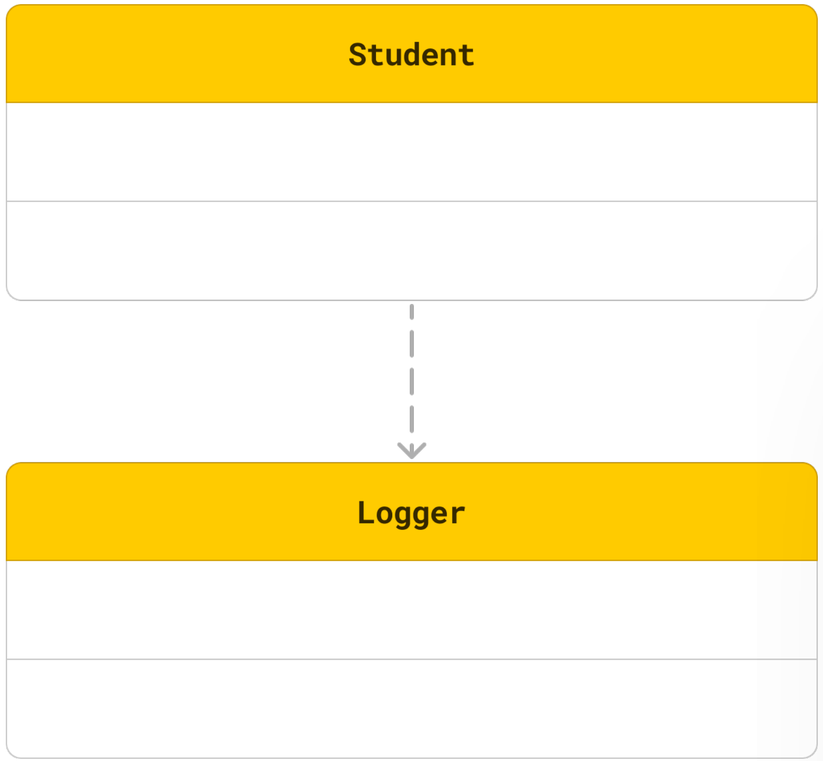


Student

```
public class Student {

    private Address address;
}
```

© Mahmoud Youssef

DEPENDENCY



Student

```
public class Student {

    public void logStudentActivity() {
      Logger logger = new ConsoleLogger();
      logger.log();
    }

    // OR

    public void log(Logger logger, String message) {
    }
}
```

© Mahmoud Youssef

UML - Hands on Coding

CREATIONAL

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

CREATIONAL

- Builder
- Prototype
- Factory Method
- Abstract Factory
- Singleton

© Mahmoud Youssef

BUILDER

BUILDER DESIGN PATTERN

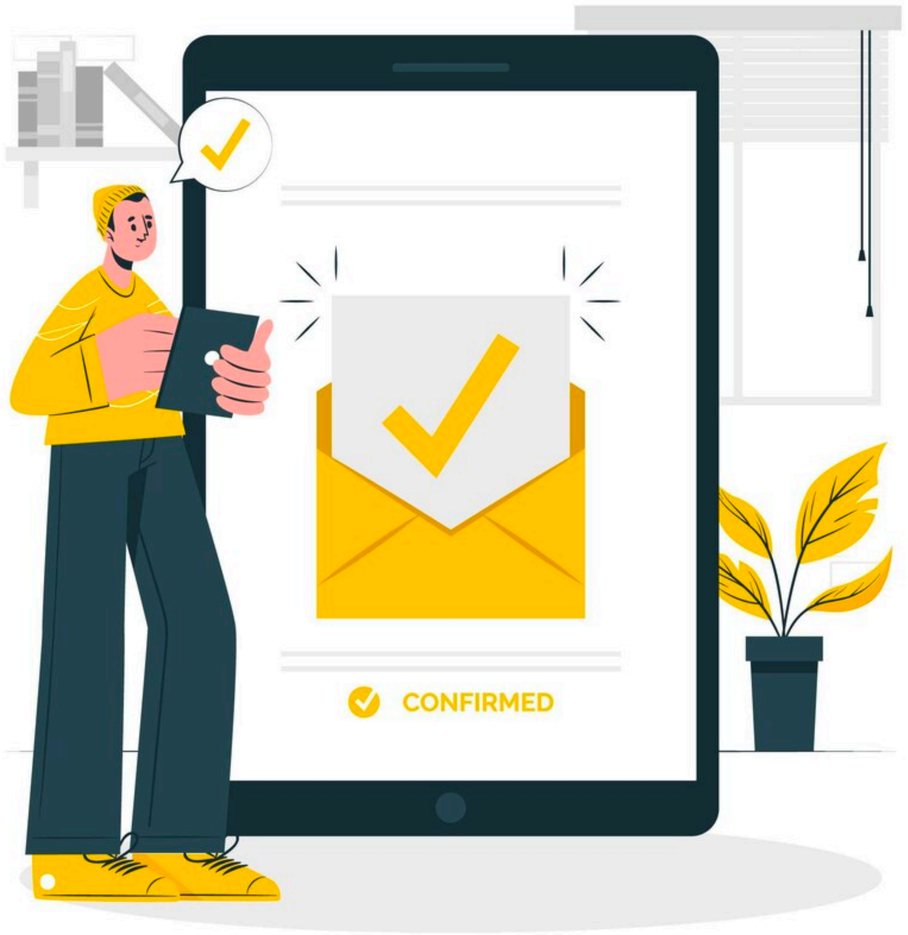
Creational design patterns provide various object creation mechanisms, which ***increase flexibility and reusability.***

© Mahmoud Youssef

WHAT IS THE PROBLEM?

Let's consider a scenario where you need to ***create and configure*** a notification in a messaging application.

Notifications can have ***various attributes*** such as the message content, sender details, timestamp, and ***optional features*** like attachments.



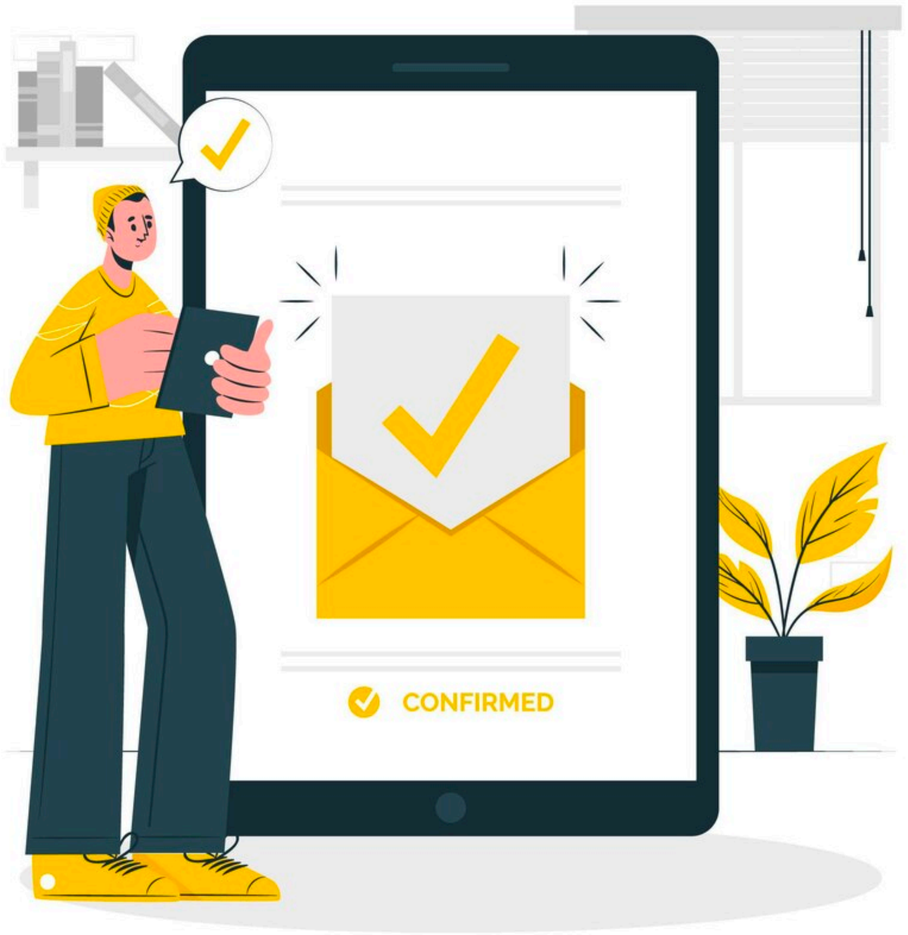
© Mahmoud Youssef

BUILDER

WHAT IS THE PROBLEM?

Let's consider a scenario where you need to ***create and configure*** a notification in a messaging application.

Notifications can have ***various attributes*** such as the message content, sender details, timestamp, and ***optional features*** like attachments.



© Mahmoud Youssef

NAIVE SOLUTION

Notification

Attributes
- content: String
- sender: String
- recipient: String
- timestamp: String
- attachments: List<String>

Operations
+ getContent(): String
+ setContent(String)
+ getSender(): String
+ setSender(String)
+ getRecipient(): String
+ setRecipient(String)
+ getTimestamp(): String
+ setTimestamp(String)
+ getAttachments(): List<String>
+ setAttachments(List<String>)

Mandatory vs. Optional

Immutability

Readability

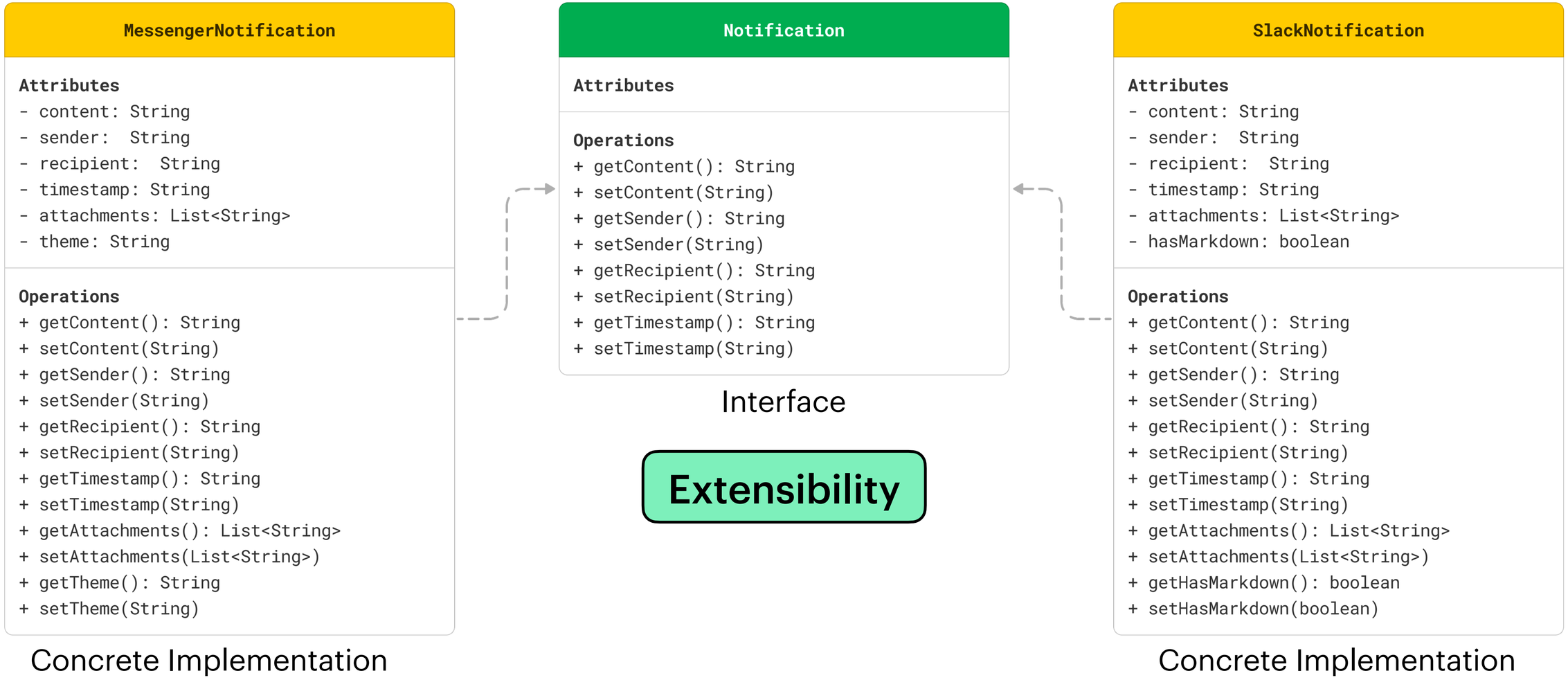
Extensibility

Notification

```java
public class Main {
    public static void main(String[] args) {
        Notification notification = new Notification();
        notification.setContent("Your task is due date!");
        notification.setSender("noreply@example.com");
        notification.setRecipient("mahmoud@example.com");
        notification.setTimestamp("2024-01-01 12:00");
        // other attributes
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



© Mahmoud Youssef

ENHANCED SOLUTION

Concrete Implementation

MessengerNotificationBuilder

Attributes
- content: String

Operations
+ setContent(String)

Implements

Interface

NotificationBuilder

Attributes

Operations
+ setContent(String)
+ setSender(String)
+ setRecipient(String)
+ setTimestamp(String)

Implements

Concrete Implementation

SlackNotificationBuilder

Attributes
- content: String

Operations
+ setContent(String)

Create Product

MessengerNotification

Attributes
- content: String

Operations
+ getContent(): String

Final Notification Product

Create Product

SlackNotification

Attributes
- content: String

Operations
+ getContent(): String

Final Notification Product

Mandatory vs. Optional

Immutability

© Mahmoud Youssef

ENHANCED SOLUTION

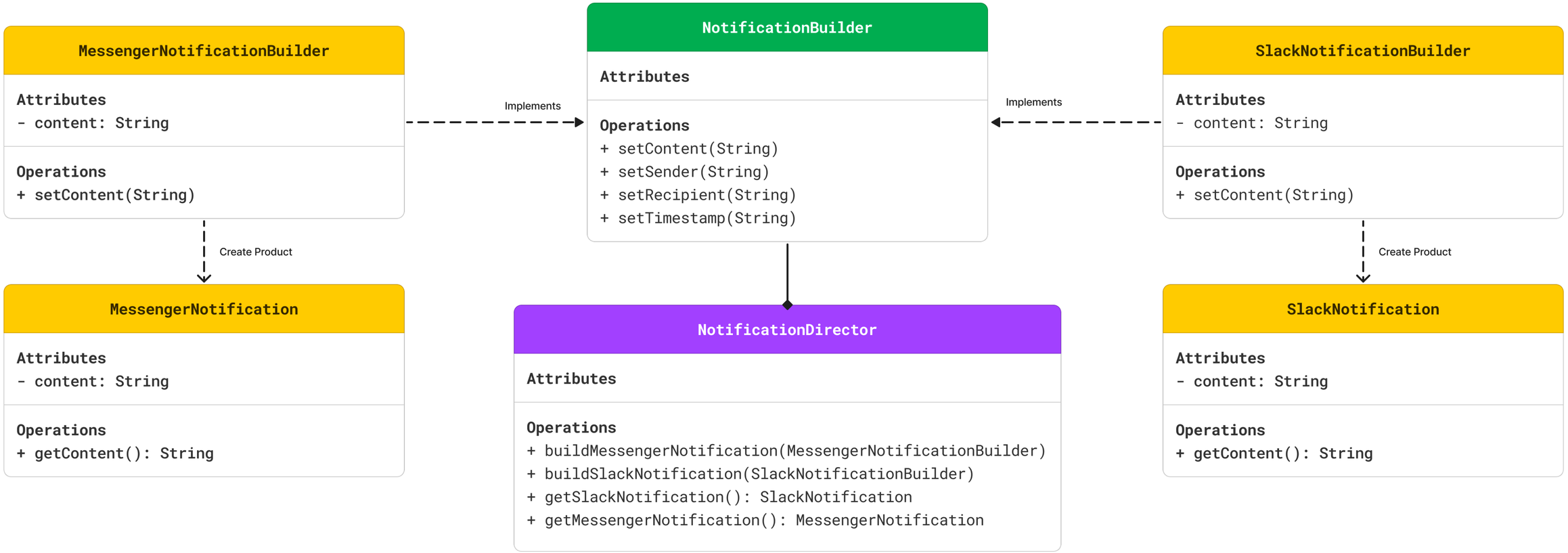
Notification

Readability

```
public class Main {
    public static void main(String[] args) {
        SlackNotification notification = new SlackNotification
        .SlackNotificationBuilder()
        .setContent("Your task is due!")
        .setSender("noreply@example.com")
        .setRecipient("mahmoud@example.com")
        .setTimestamp("2024-01-01 12:00")
        .setAttachments(attachments)
        .setHasMarkdown(true)
        .setIsRead(true)
        .setTheme("Dark")
        // ... other customization methods
        .build();
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



© Mahmoud Youssef

IMPLEMENTATION

SOLUTION

SLACK NOTIFICATION

JAVA BEST PRACTICES

EMAIL NOTIFICATION

SENDING HTTP REQUEST

REAL LIFE EXAMPLE

SINGLETON

LOGGING LEVELS

Least Severity

Most Severity

ALL

TRACE

DEBUG

INFO

WARN

ERROR

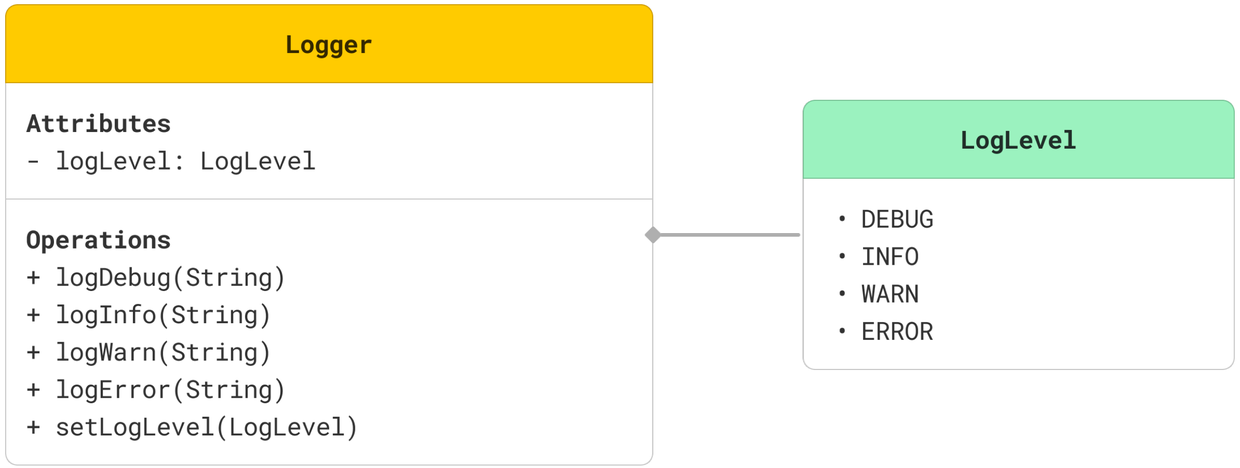
FATAL

Most Logging

Least Logging

© Mahmoud Youssef

NAIVE SOLUTION



No Global Control

Resource Overhead

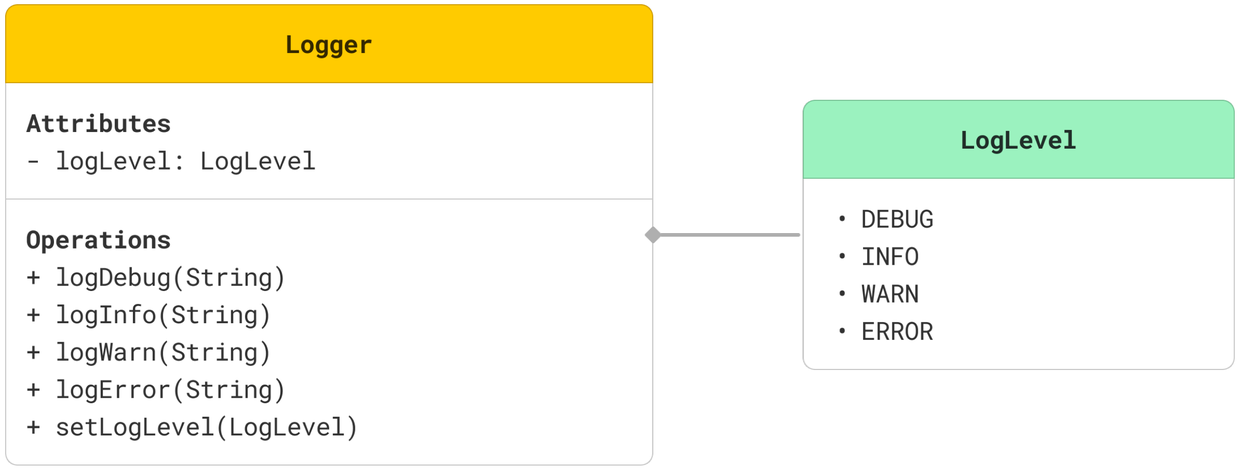
Consistency

Logger

```java
public class Logger {

    private LogLevel logLevel = LogLevel.INFO;

    public void setLogLevel(LogLevel logLevel) {
        this.logLevel = logLevel;
    }

    public void logDebug(String message) {
        if(logLevel.ordinal() <= LogLevel.DEBUG.ordinal()) {
            System.out.println("DEBUG: " + message);
        }
    }
    public void logInfo(String message) {
        if(logLevel.ordinal() <= LogLevel.INFO.ordinal()) {
            System.out.println("INFO: " + message);
        }
    }
    public void logWarn(String message) {
        if(logLevel.ordinal() <= LogLevel.WARN.ordinal()) {
            System.out.println("WARN: " + message);
        }
    }
    public void logError(String message) {
        if(logLevel.ordinal() <= LogLevel.ERROR.ordinal()) {
            System.out.println("ERROR: " + message);
        }
    }
}
```

© Mahmoud Youssef

NAIVE SOLUTION



Student

```java
public class Logger {

    private LogLevel logLevel = LogLevel.INFO;

    public void setLogLevel(LogLevel logLevel) {
        this.logLevel = logLevel;
    }

    public void logDebug(String message) {
        if(logLevel.ordinal() <= LogLevel.DEBUG.ordinal()) {
            System.out.println("DEBUG: " + message);
        }
    }
    public void logInfo(String message) {
        if(logLevel.ordinal() <= LogLevel.INFO.ordinal()) {
            System.out.println("INFO: " + message);
        }
    }
    public void logWarn(String message) {
        if(logLevel.ordinal() <= LogLevel.WARN.ordinal()) {
            System.out.println("WARN: " + message);
        }
    }
    public void logError(String message) {
        if(logLevel.ordinal() <= LogLevel.ERROR.ordinal()) {
            System.out.println("ERROR: " + message);
        }
    }
}
```

No Global Control

Resource Overhead

Consistency

© Mahmoud Youssef

ENHANCED SOLUTION

Eager Initialization

Global Control

Resource Efficient

Consistency

Logger

Attributes

```
- logLevel: LogLevel
- instance: Logger = new Logger()
```

Operations

```
- Logger()
+ logDebug(String)
+ logInfo(String)
+ logWarn(String)
+ logError(String)
+ setLogLevel(LogLevel)
+ getInstance(): Logger
```

LogLevel

- DEBUG
- INFO
- WARN
- ERROR

© Mahmoud Youssef

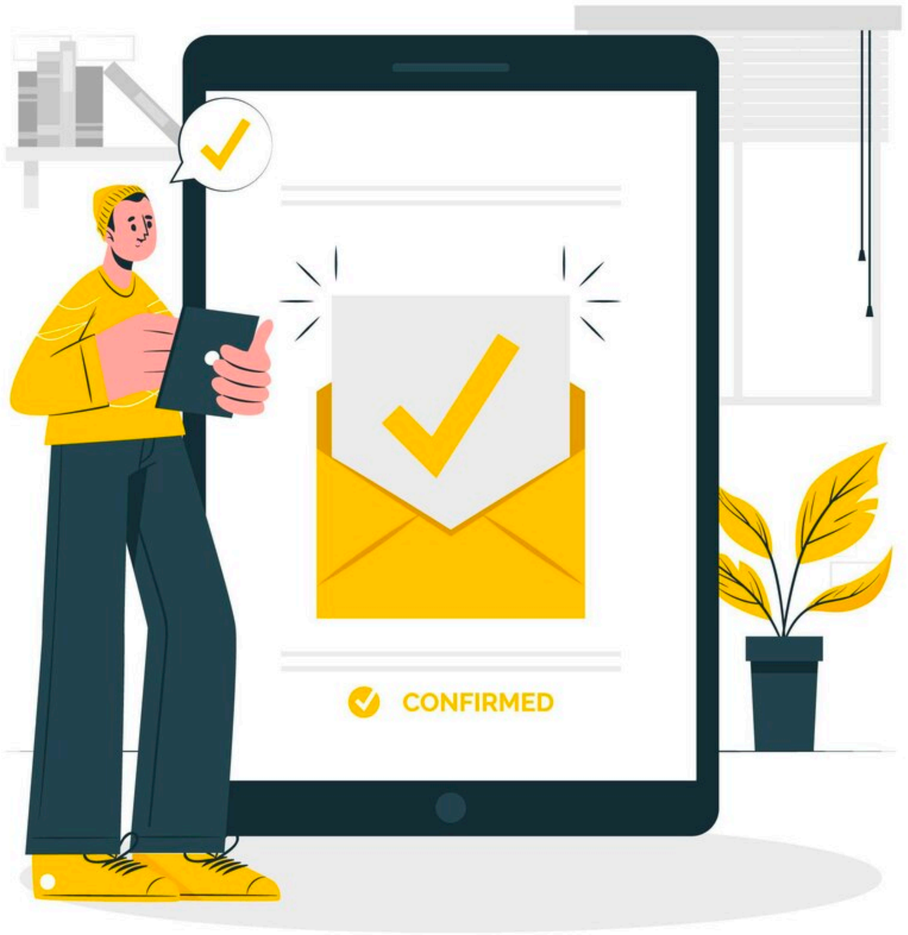
IMPLEMENTATION

PROTOTYPE

WHAT IS THE PROBLEM?

Let's consider a scenario where you need to ***create and configure customized*** brochures, flyers or posters for clients in a marketing company.

Each marketing material can have *different layouts, colors, and content* which makes the creation process a bit ***complex and time-consuming***.



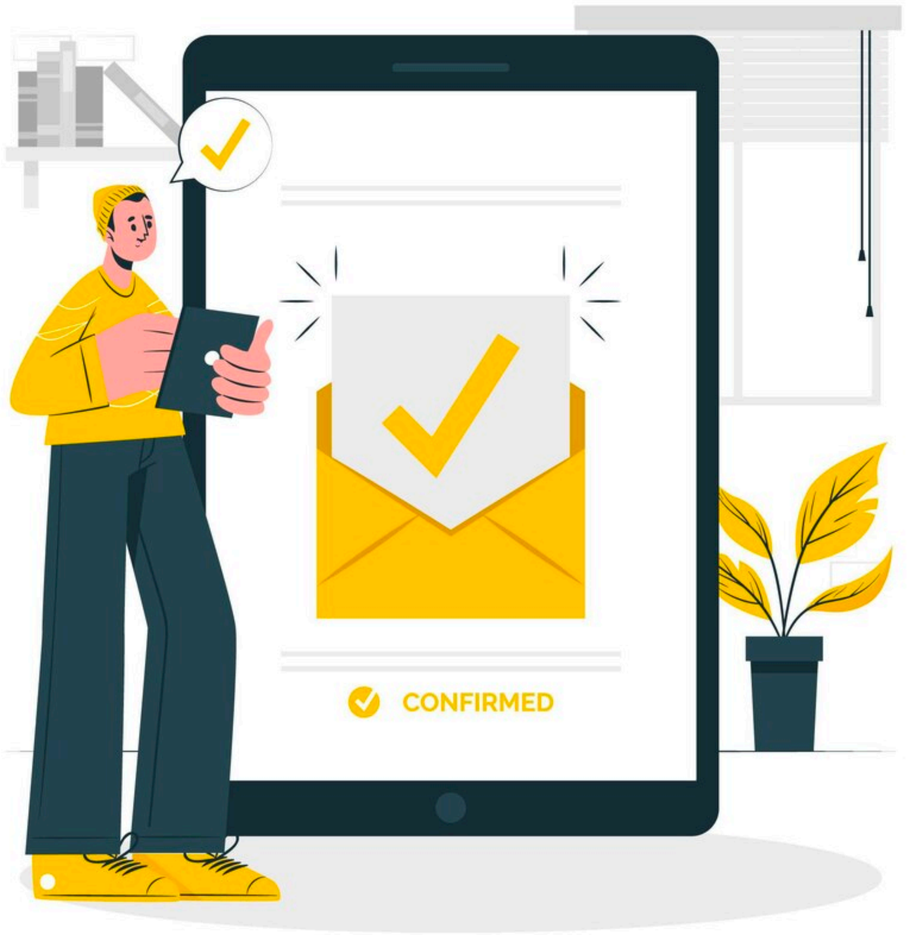
© Mahmoud Youssef

SOLUTION

WHAT IS THE PROBLEM?

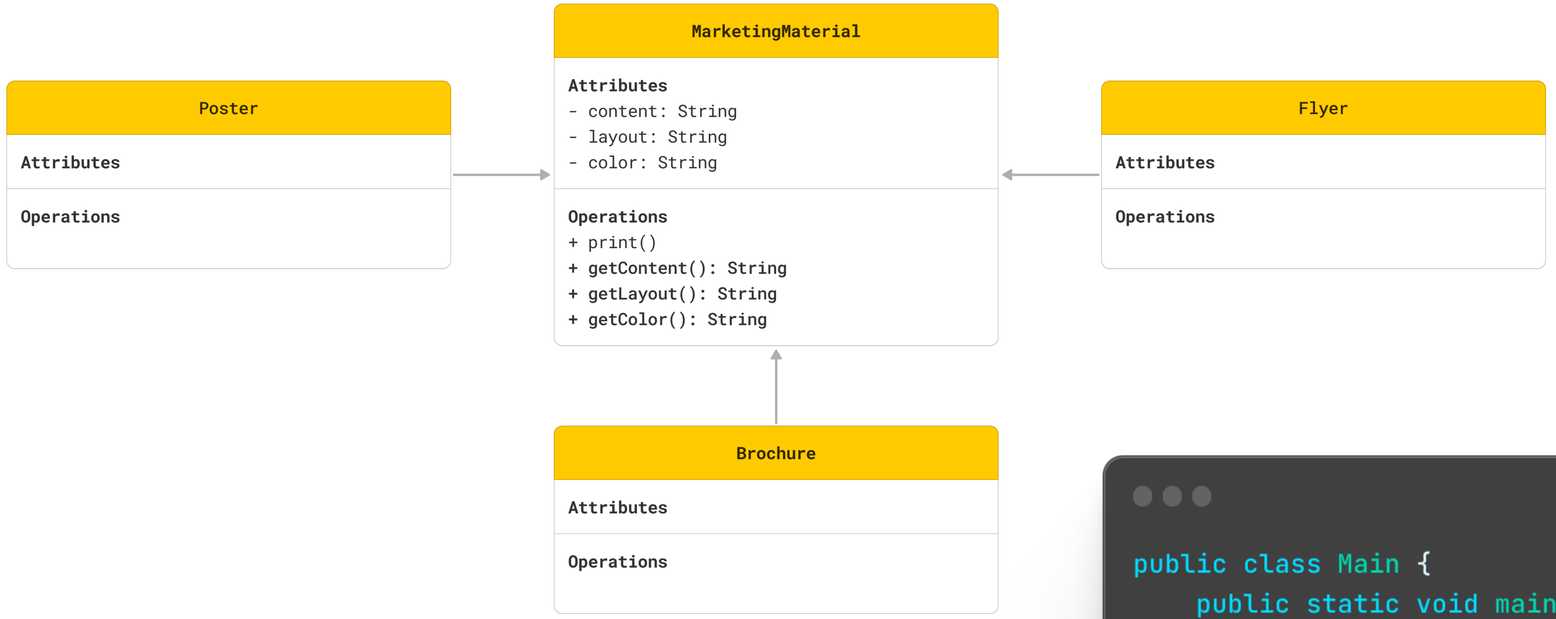
Let's consider a scenario where you need to ***create and configure customized*** brochures, flyers or posters for clients in a marketing company.

Each marketing material can have *different layouts, colors, and content* which makes the creation process a bit ***complex and time-consuming***.



© Mahmoud Youssef

NAIVE SOLUTION



Code Duplication

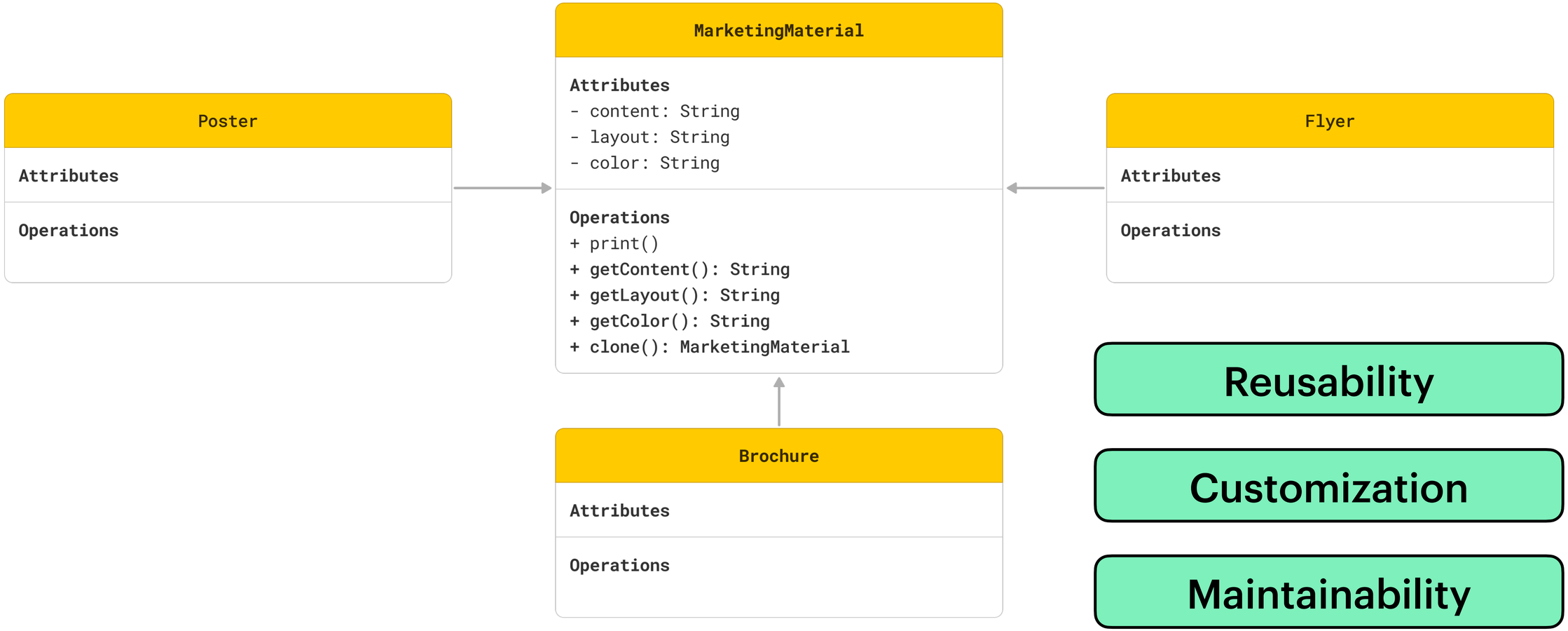
Maintainability

MarketingMaterial

```java
public class Main {
    public static void main(String[] args) {
        Flyer flyer = new Flyer("Layout A4", "Ultimate Design Patterns",
                "dark-blue");
        Flyer flyerA3 = new Flyer("Layout A3", "Ultimate Design Patterns",
                "dark-blue");
        Poster poster = new Poster("Layout A3", "Ultimate Design Patterns",
                "dark-blue");
        flyer.print();
        flyerA3.print();
        poster.print();
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



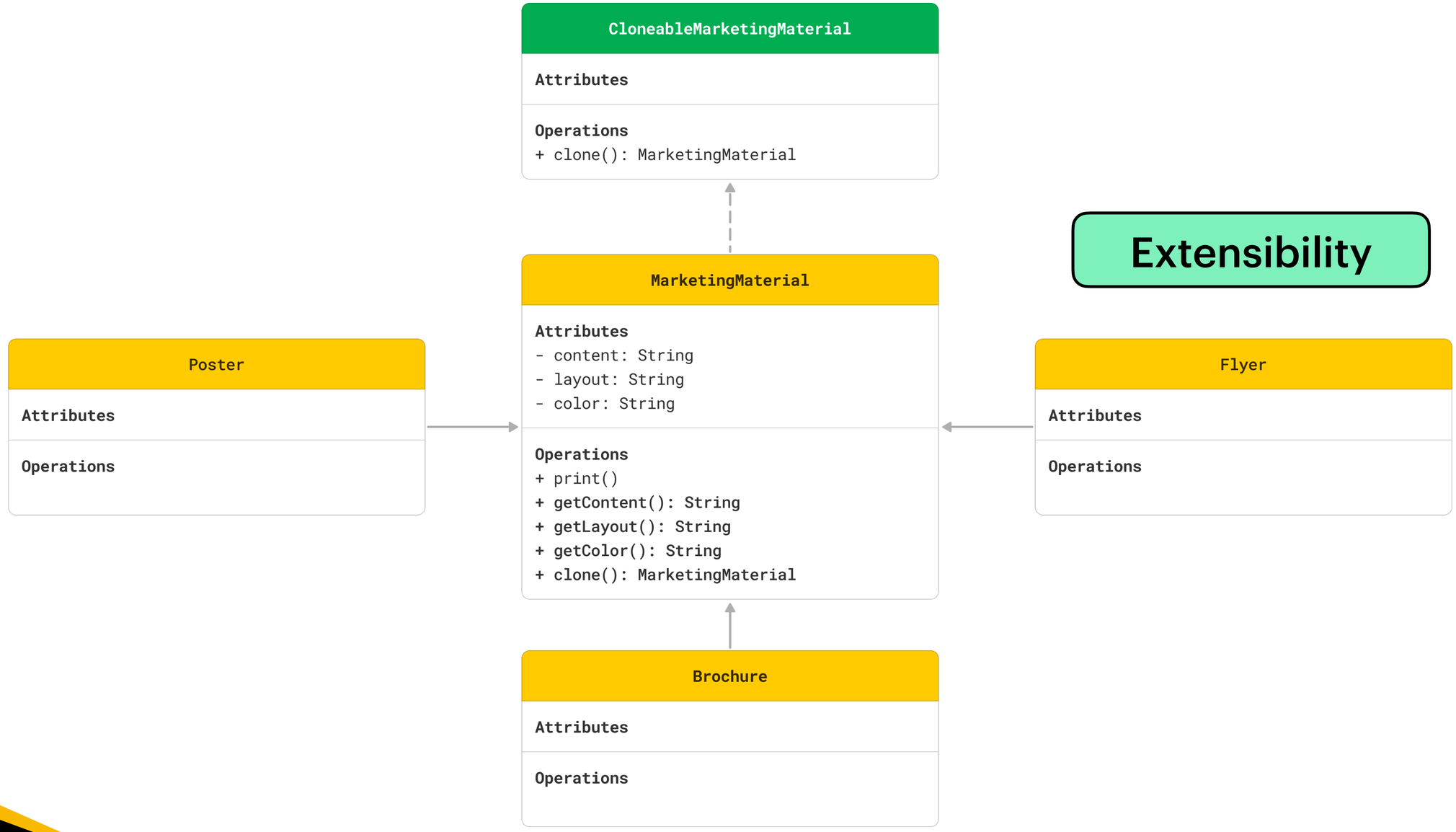
Reusability

Customization

Maintainability

© Mahmoud Youssef

ENHANCED SOLUTION



Extensibility

© Mahmoud Youssef

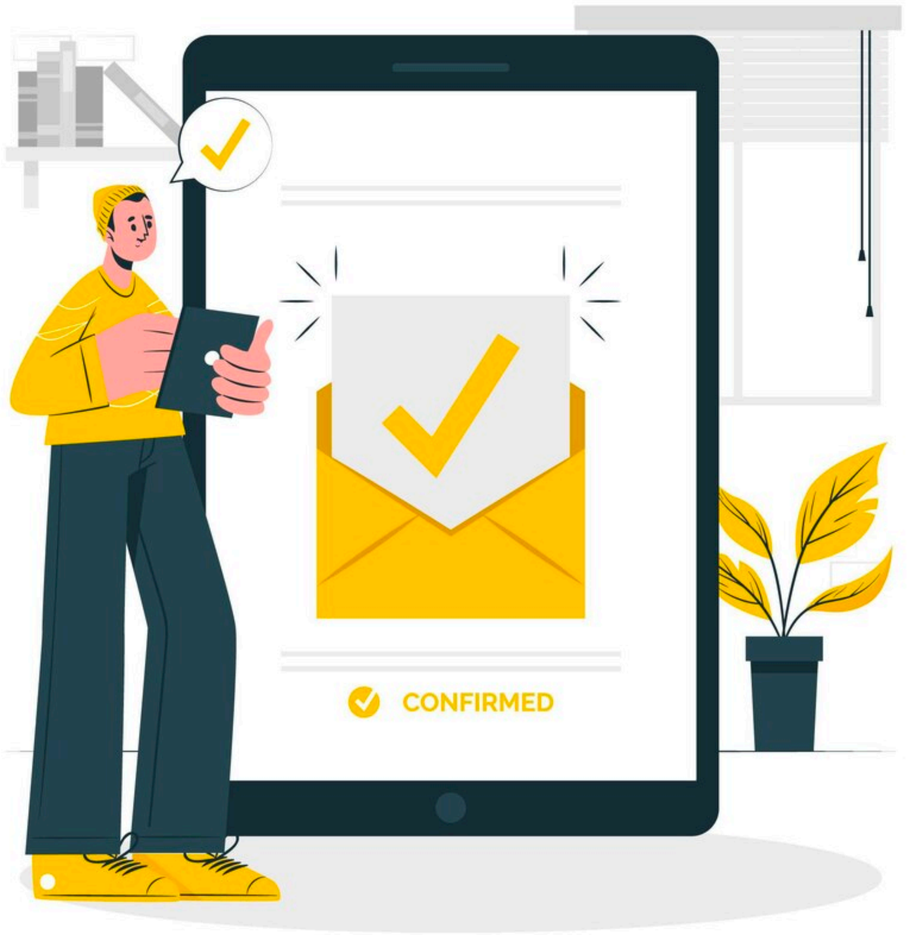
IMPLEMENTATION

FACTORY

WHAT IS THE PROBLEM?

Let's consider a scenario where you need to create a ***payment processor*** in e-commerce application that is responsible for handling various kind of payment methods such as ***Visa, MasterCard and AmericanExpress***.

The processing of payments consists of three steps; ***authorization, startMoneyTransfer, and calculatePaymentFee. Design it as simple as you can!***



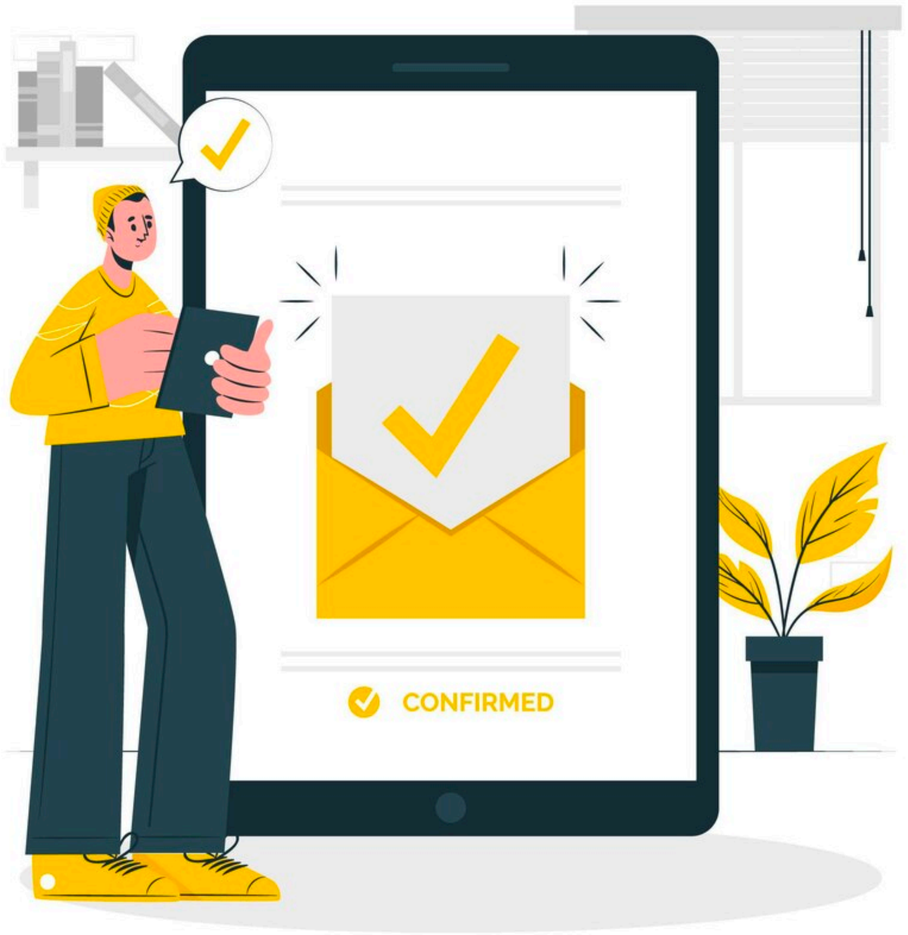
© Mahmoud Youssef

SOLUTION

WHAT IS THE PROBLEM?

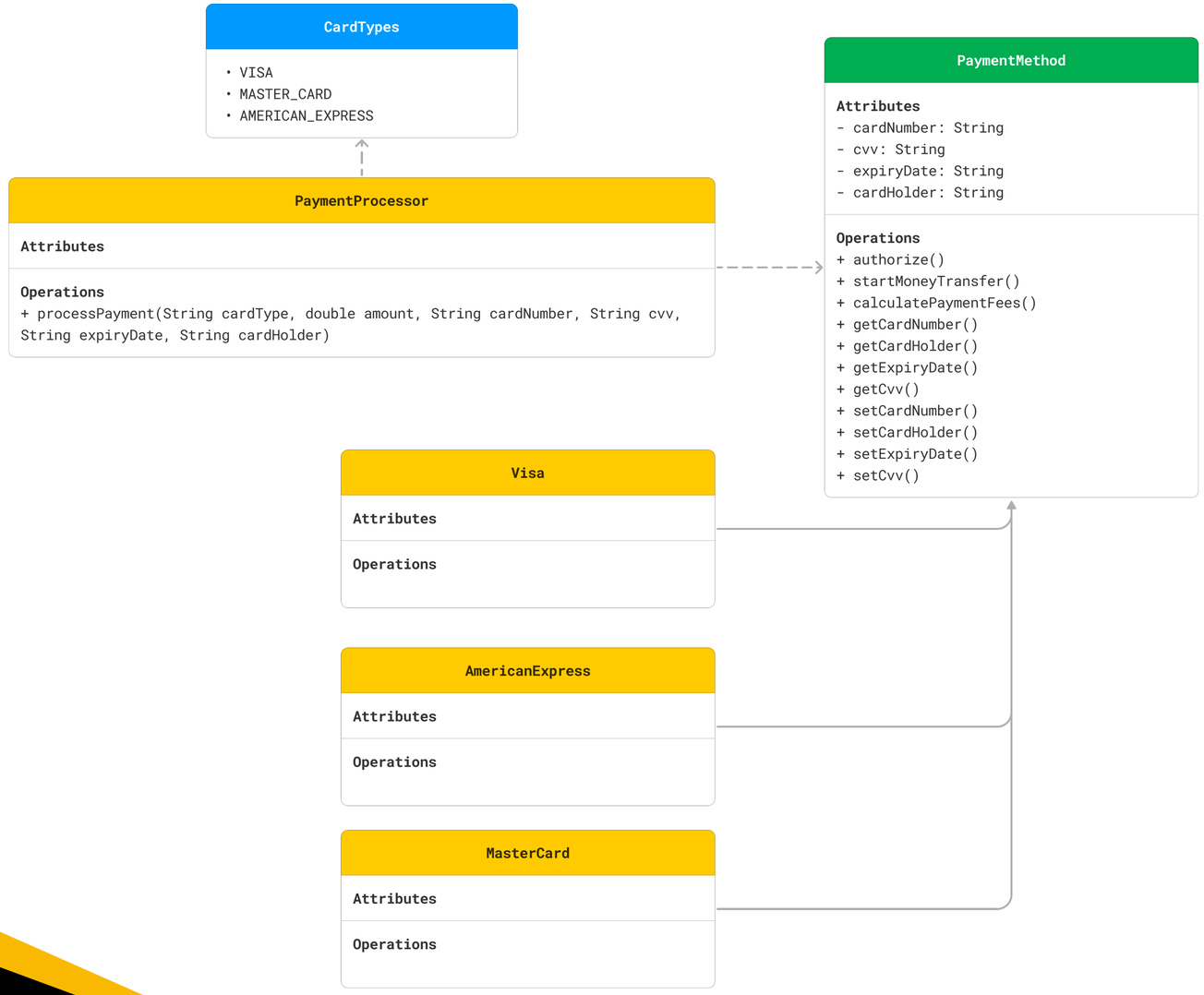
Let's consider a scenario where you need to create a ***payment processor*** in e-commerce application that is responsible for handling various kind of payment methods such as ***Visa, MasterCard and AmericanExpress***.

The processing of payments consists of three steps; ***authorization, startMoneyTransfer, and calculatePaymentFee. Design it as simple as you can!***



© Mahmoud Youssef

NAIVE SOLUTION



Reusability

Duplication

Maintainability

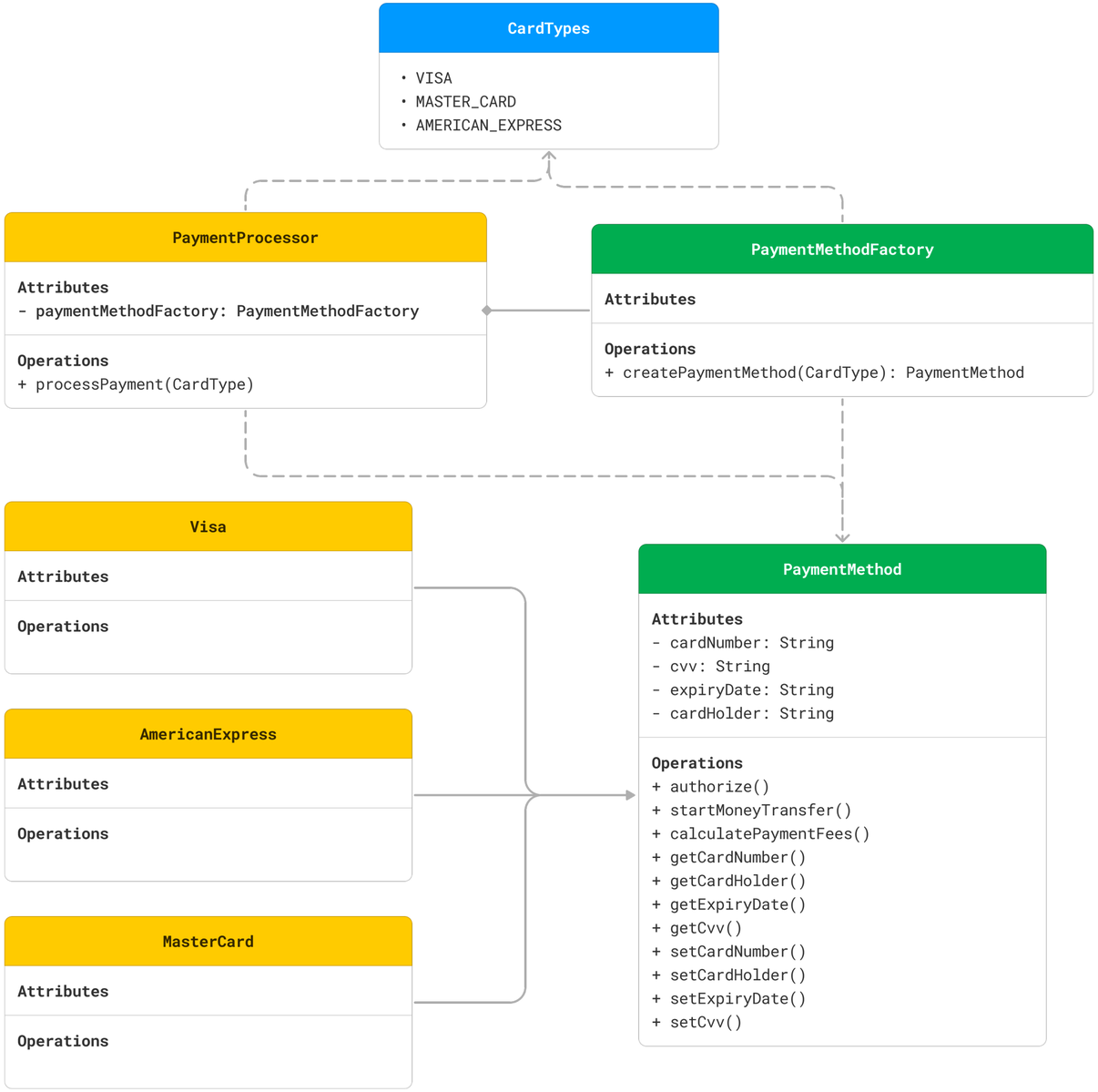
Extensibility

PaymentProcessor

```java
public class PaymentProcessor {

    public void processPayment(String cardType, double amount, String cardHolder, String cardNumber,
                               String cvv, String expiryDate) {
        // logic of create the real card + processing
        PaymentMethod paymentMethod = null;
        if (cardType.equalsIgnoreCase(CardTypes.VISA.toString())) {
            paymentMethod = new Visa(cardHolder, cardNumber, cvv, expiryDate);
        } else if (cardType.equalsIgnoreCase(CardTypes.MASTER_CARD.toString())) {
            paymentMethod = new MasterCard(cardHolder, cardNumber, cvv, expiryDate);
        } else if (cardType.equalsIgnoreCase(CardTypes.AMERICAN_EXPRESS.toString())) {
            paymentMethod = new AmericanExpress(cardHolder, cardNumber, cvv, expiryDate);
        } else {
            throw new IllegalArgumentException("Card type is not supported...");
        }

        // process payment
        paymentMethod.authorizePayment();
        paymentMethod.startMoneyTransfer();
        paymentMethod.calculatePaymentFees(amount);
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Reusability

Maintainability

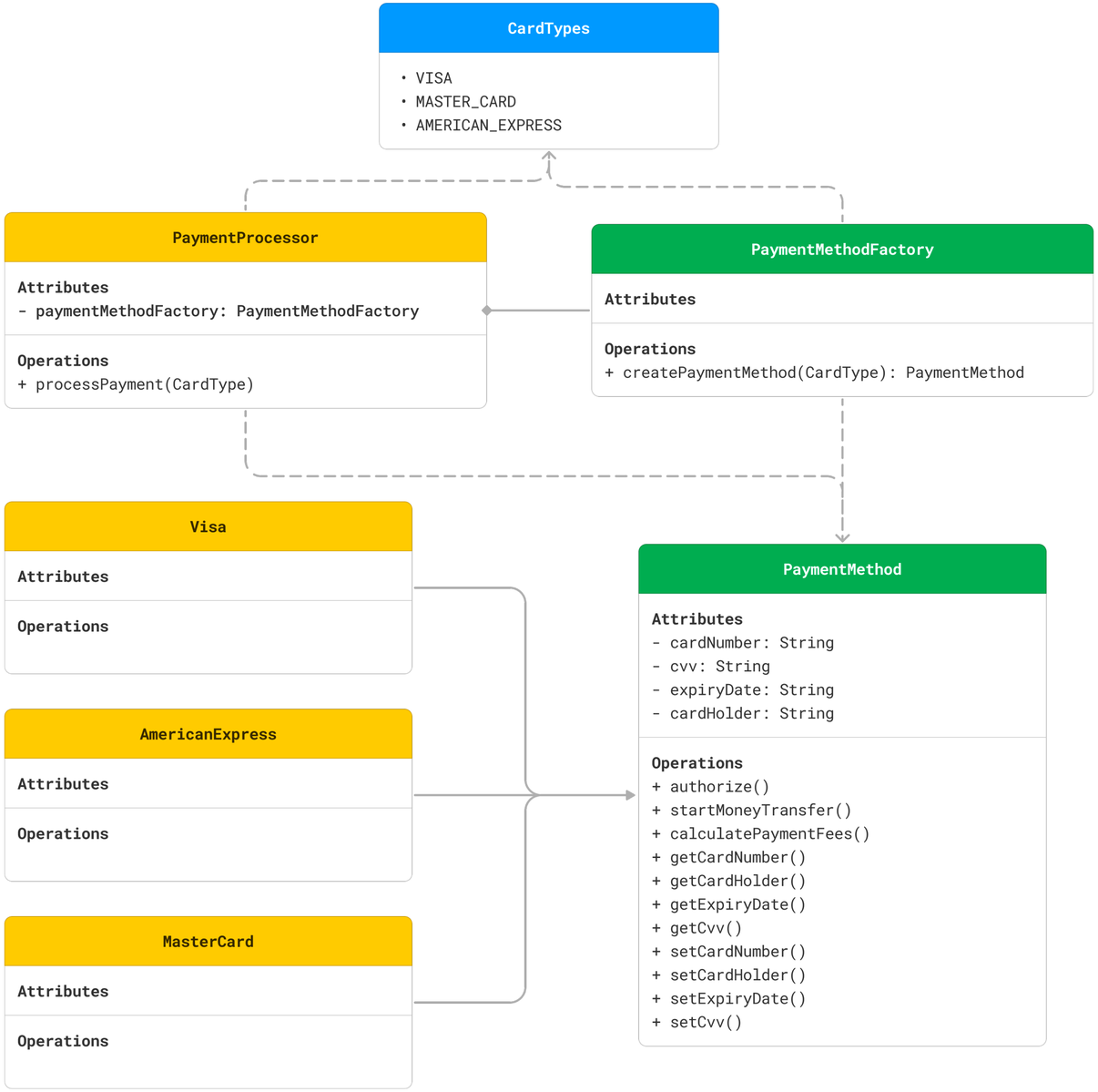
Extensibility

© Mahmoud Youssef

IMPLEMENTATION

ABSTRACT FACTORY

ENHANCED SOLUTION



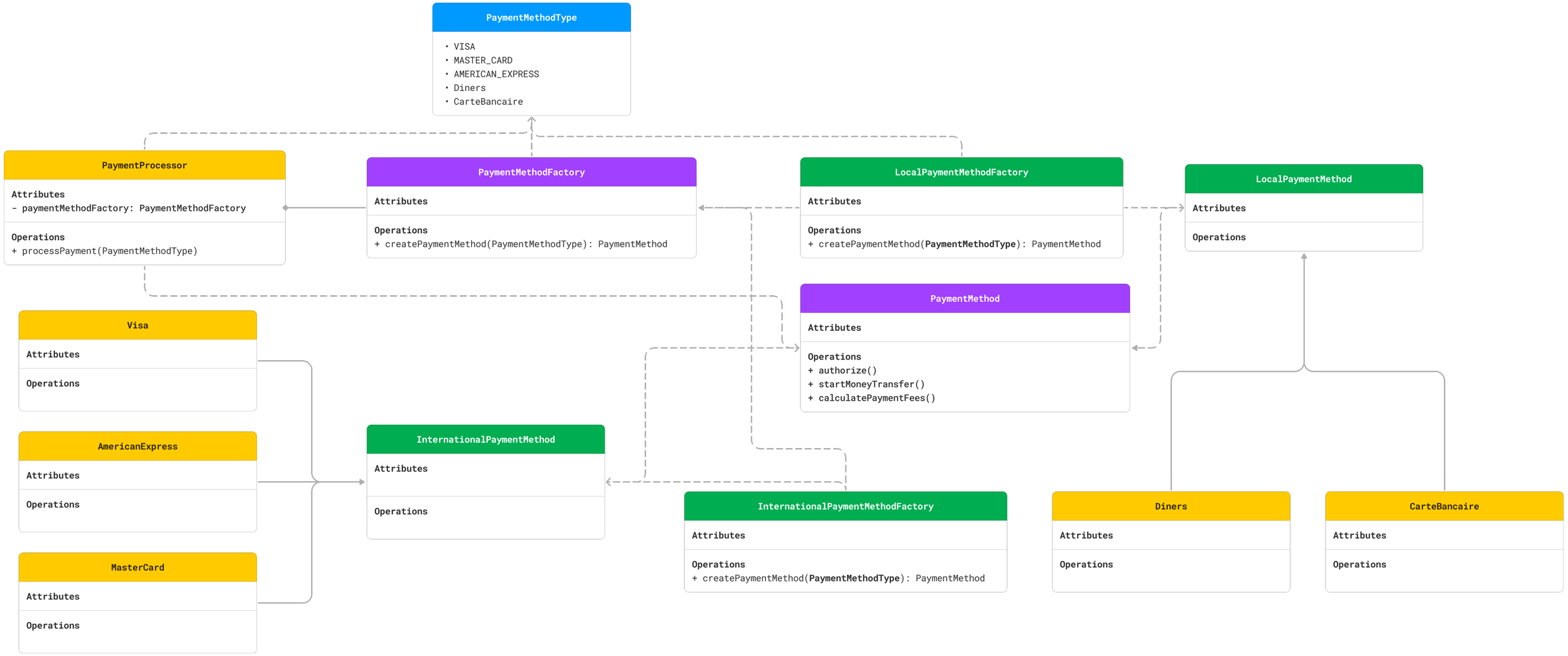
Reusability

Maintainability

Extensibility

© Mahmoud Youssef

ENHANCED SOLUTION



© Mahmoud Youssef

STRUCTURAL

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

STRUCTURAL

Proxy

Facade

Adapter

Bridge

Composite

Decorator

Flyweight

© Mahmoud Youssef

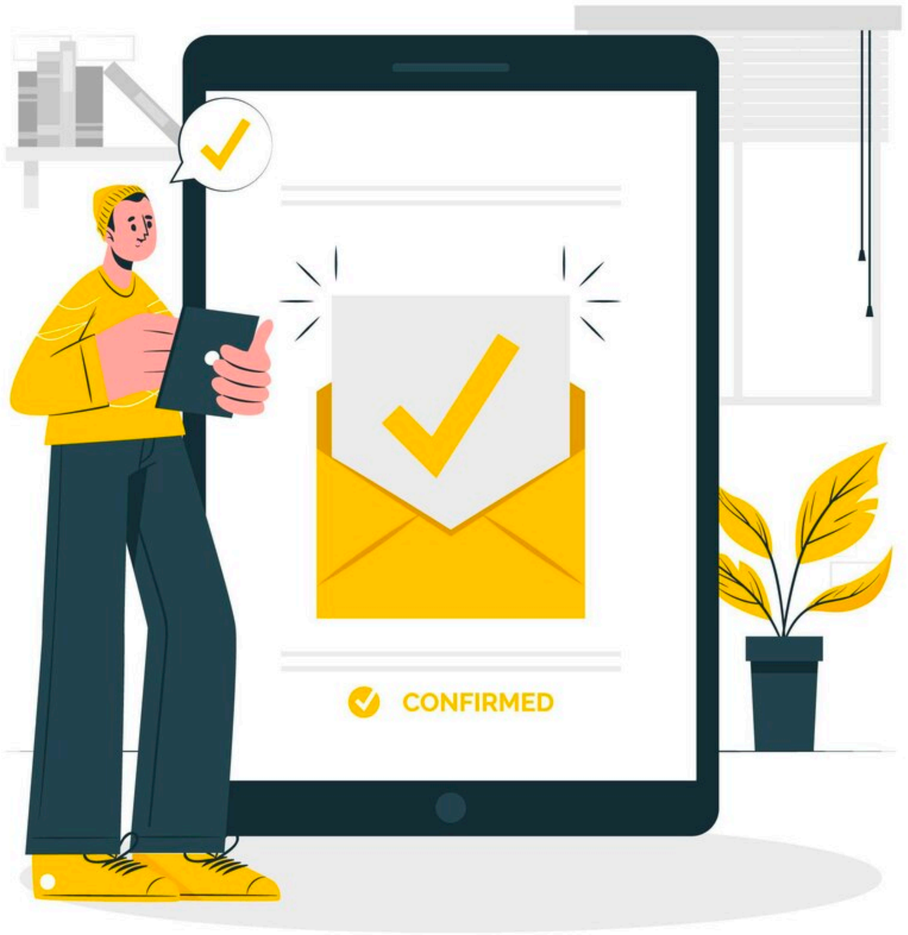
ADAPTER

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

Let's consider a scenario where we have a weather application that expects to receive city and country to show users weather information in XML format.

It's required to ***change the current structure of Weather Application ,*** to receive a ***location of longitude and latitude*** then extract the city and country from it and use the current third-party which takes city and country. Also we need to show the information in ***JSON format***.



© Mahmoud Youssef

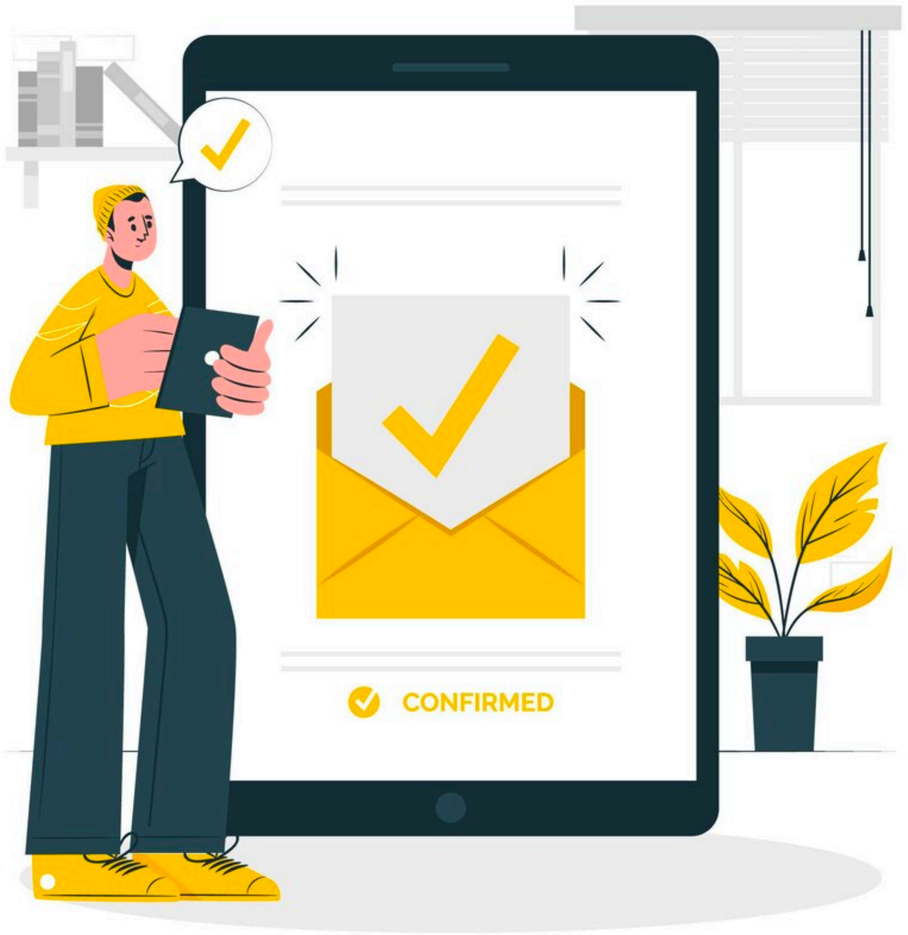
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

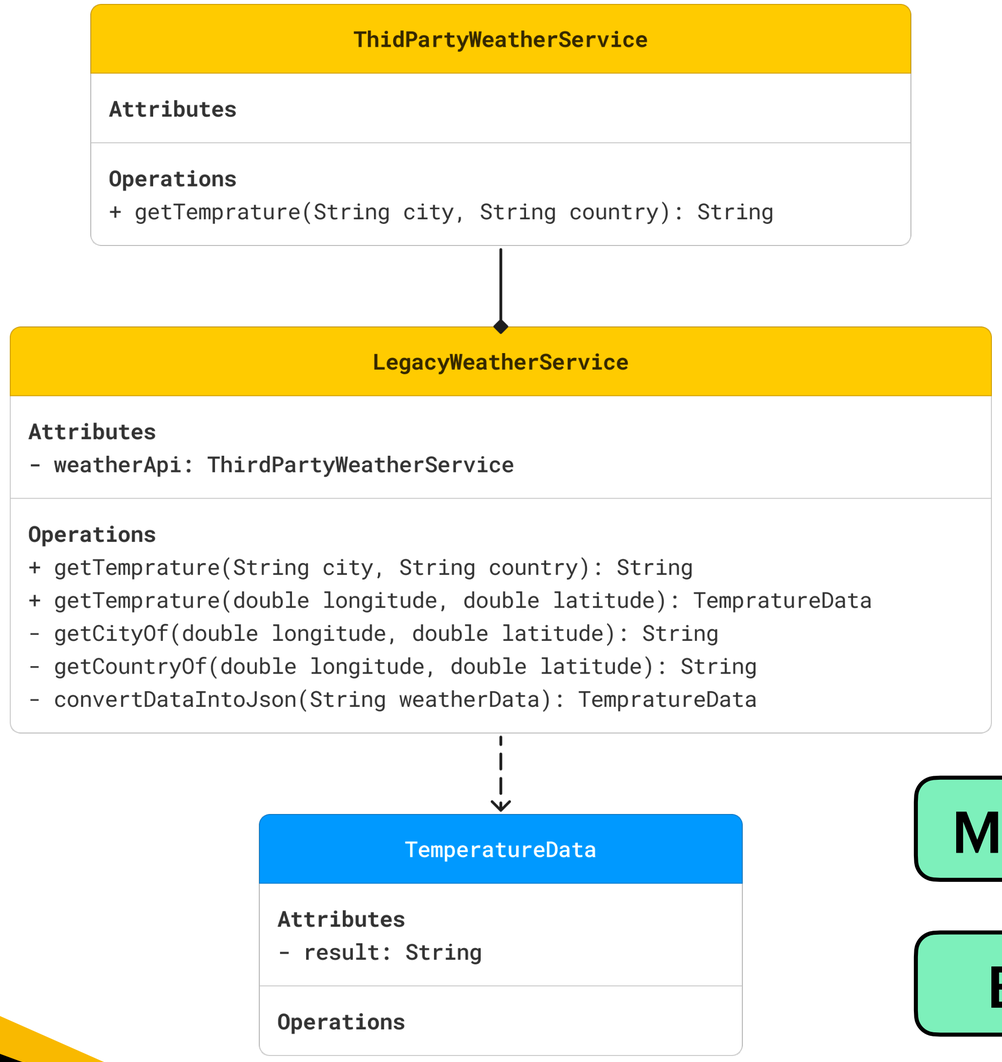
Let's consider a scenario where we have a weather application that expects to receive city and country to show users weather information in XML format.

It's required to ***change the current structure of Weather Application ,*** to receive a ***location of longitude and latitude*** then extract the city and country from it and use the current third-party which takes city and country. Also we need to show the information in ***JSON format***.



© Mahmoud Youssef

NAIVE SOLUTION



Maintainability

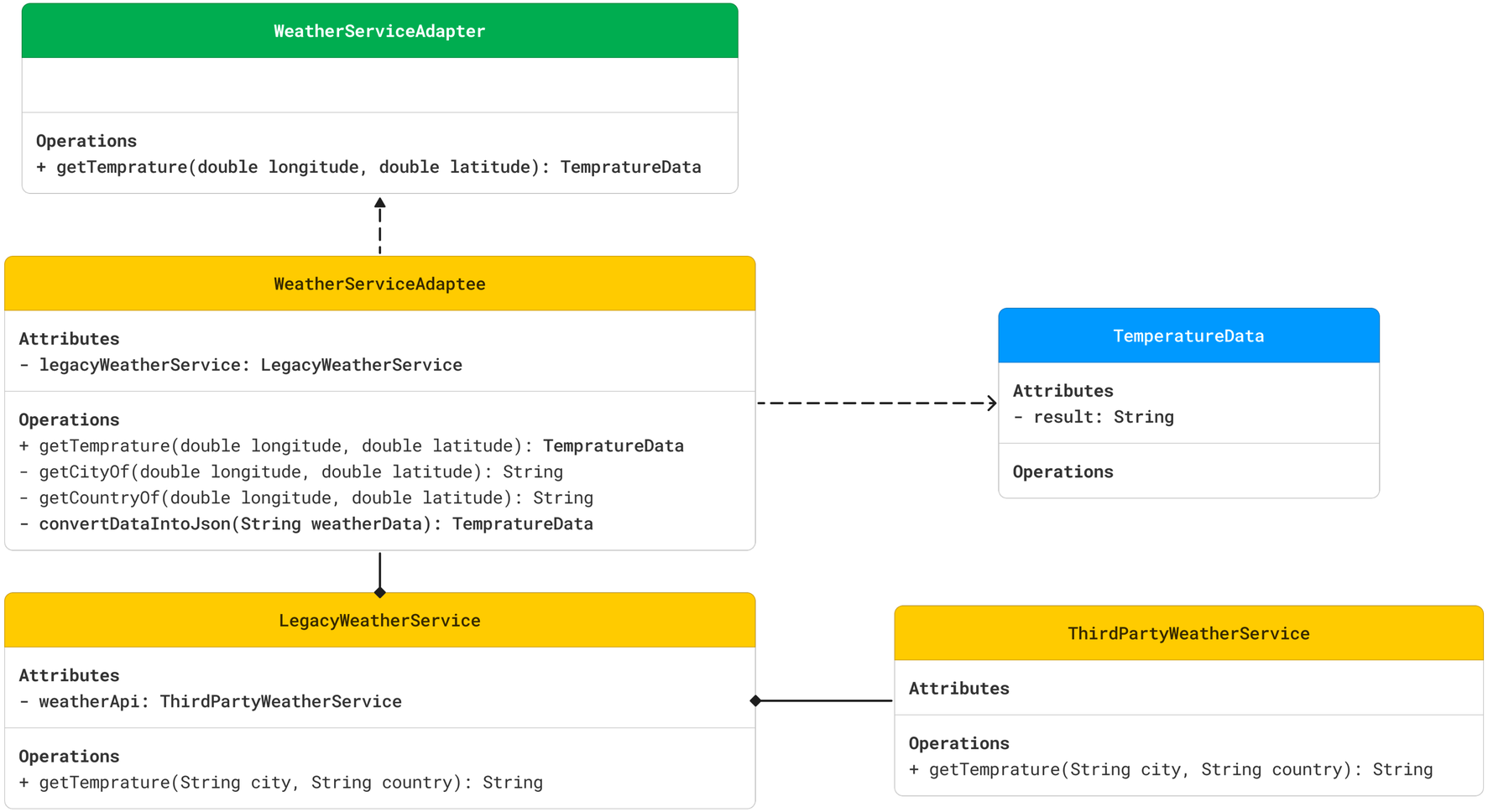
Extensibility

LegacyWeatherService

```java
public class LegacyWeatherService {

    private final ThirdPartyWeatherService weatherApi;

    public LegacyWeatherService(ThirdPartyWeatherService weatherApi) {
        this.weatherApi = weatherApi;
    }

    public String getTemperature(String city, String country) {
        return weatherApi.getTemperature(city, country);
    }

    public TemperatureDate getTemperature(double longitude, double latitude) {
        String temperatureDataInXML = getTemperature(getCityOf(longitude, latitude),
getCountryOf(longitude, latitude));
        return convertXMLDataToJson(temperatureDataInXML);
    }

    private TemperatureDate convertXMLDataToJson(String xmlData) {
        System.out.println("Converting...");
        return new TemperatureDate("Converted Data from XML into JSON");
    }

    private String getCityOf(double longitude, double latitude) {
        System.out.println("Extracting city of longitude: " + longitude + " and
latitude: " + latitude);
        return "City";
    }

    private String getCountryOf(double longitude, double latitude) {
        System.out.println("Extracting country of longitude: " + longitude + " and
latitude: " + latitude);
        return "Country";
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Maintainability

Extensibility

© Mahmoud Youssef

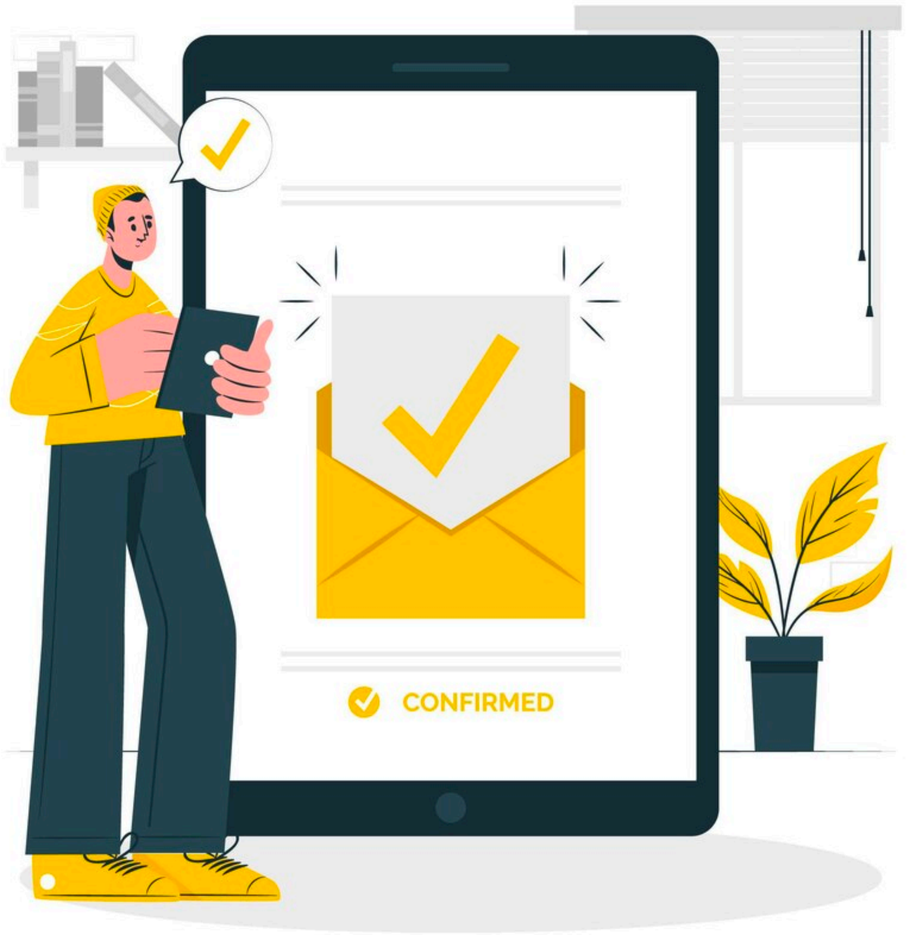
BRIDGE

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

Let's consider a scenario where we need to build a streaming video platform that supports different video providers (YouTube, Twitch...etc) with playback function.

It's required to also have the ability to set the ***quality of the video rendering (HD, SD..etc)***, considering that different platforms might have different video quality which is not supported by others.



© Mahmoud Youssef

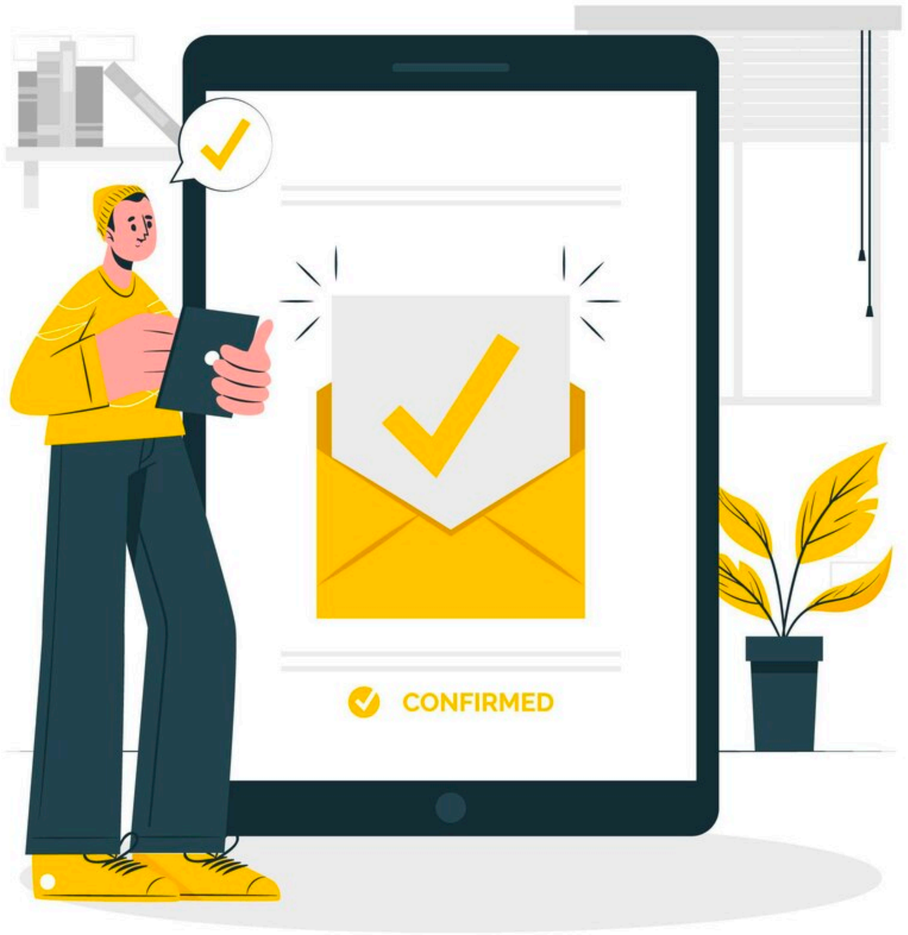
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

Let's consider a scenario where we need to build a streaming video platform that supports different video providers (YouTube, Twitch...etc) with playback function.

It's required to also have the ability to set the ***quality of the video rendering (HD, SD..etc)***, considering that different platforms might have different video quality which is not supported by others.



© Mahmoud Youssef

NAIVE SOLUTION

Code Duplication

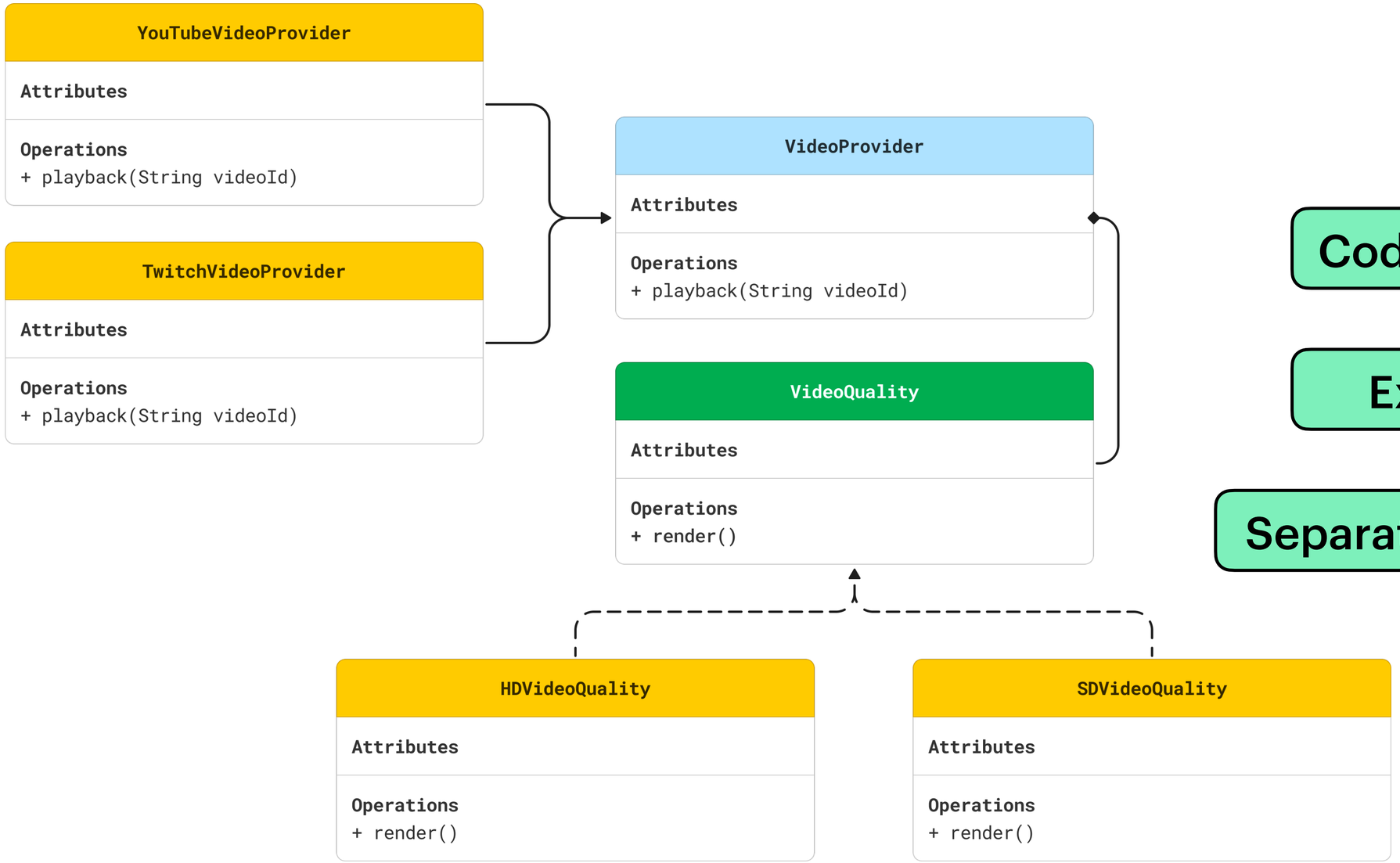
Tightly Coupling

Bridge

```java
public class Main {
    public static void main(String[] args) {
        VideoProvider youTubeVideoProvider = new YouTubeVideoProvider();
        youTubeVideoProvider.playback("1234.mp4");
        HDVideoQuality hdVideoQuality = new HDVideoQuality();
        hdVideoQuality.render();

        VideoProvider twitchVideoProvider = new TwitchVideoProvider();
        twitchVideoProvider.playback("test.mp4");
        SDVideoQuality sdVideoQuality = new SDVideoQuality();
        sdVideoQuality.render();
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Code Reusability

Extensibility

Separation of Concerns

© Mahmoud Youssef

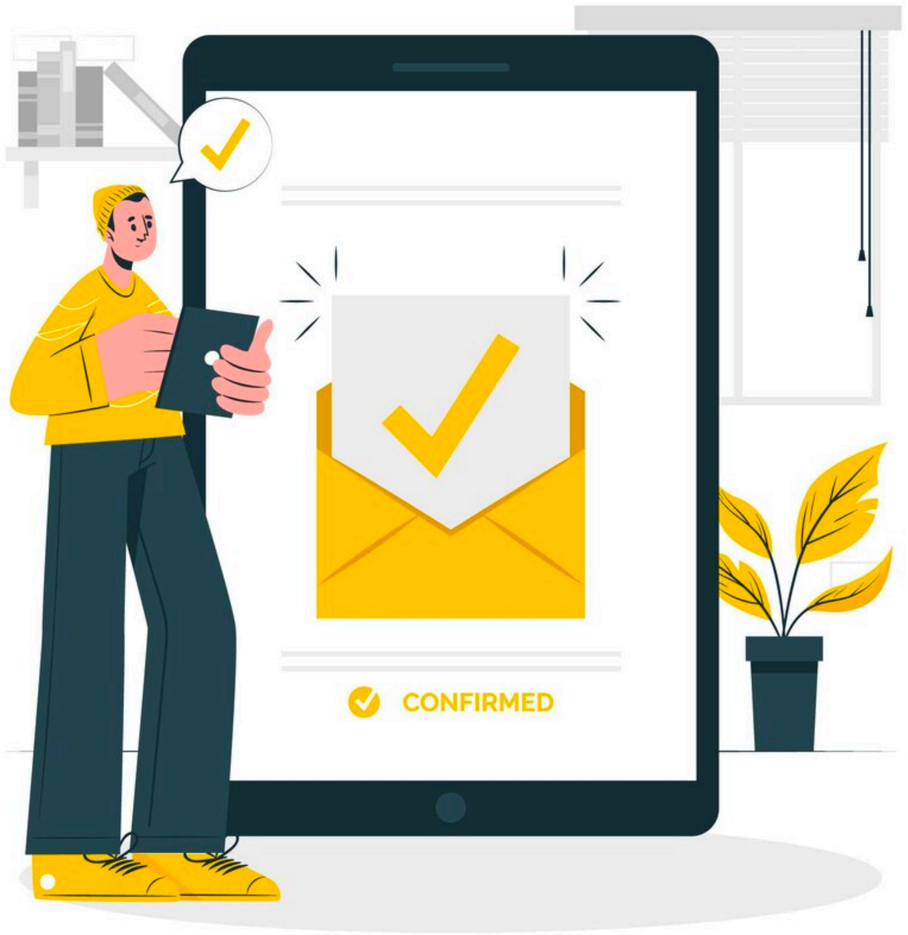
COMPOSITE

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

In a large organization, employees are organized into ***departments***, and each department may have both ***individual employees and sub-departments***.

The company wants to ***generate reports that include*** information about the hierarchy, such as the total salary of a department or its sub-departments.



© Mahmoud Youssef

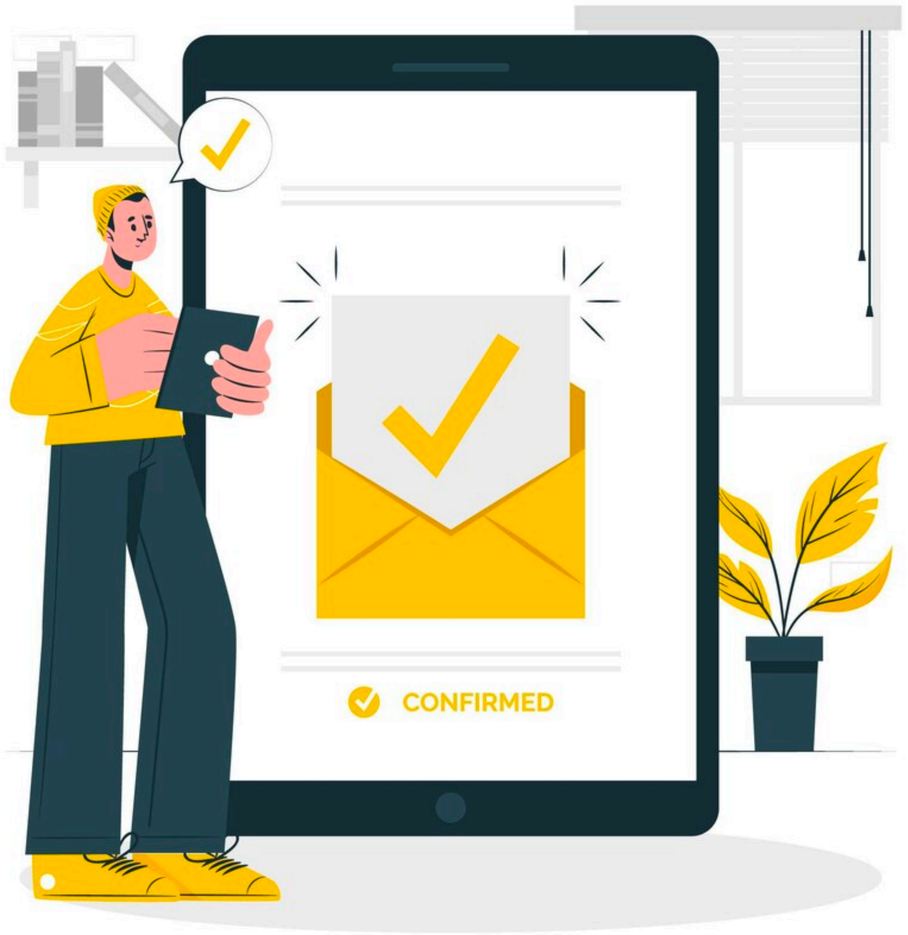
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

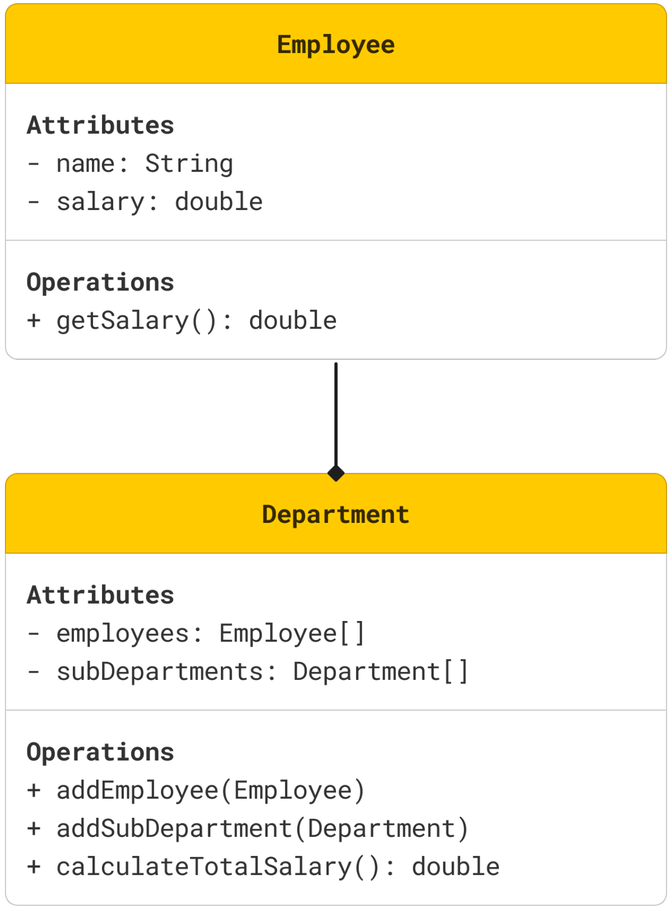
In a large organization, employees are organized into ***departments***, and each department may have both ***individual employees and sub-departments***.

The company wants to ***generate reports that include*** information about the hierarchy, such as the total salary of a department or its sub-departments.



© Mahmoud Youssef

NAIVE SOLUTION



Code Duplication

Complexity

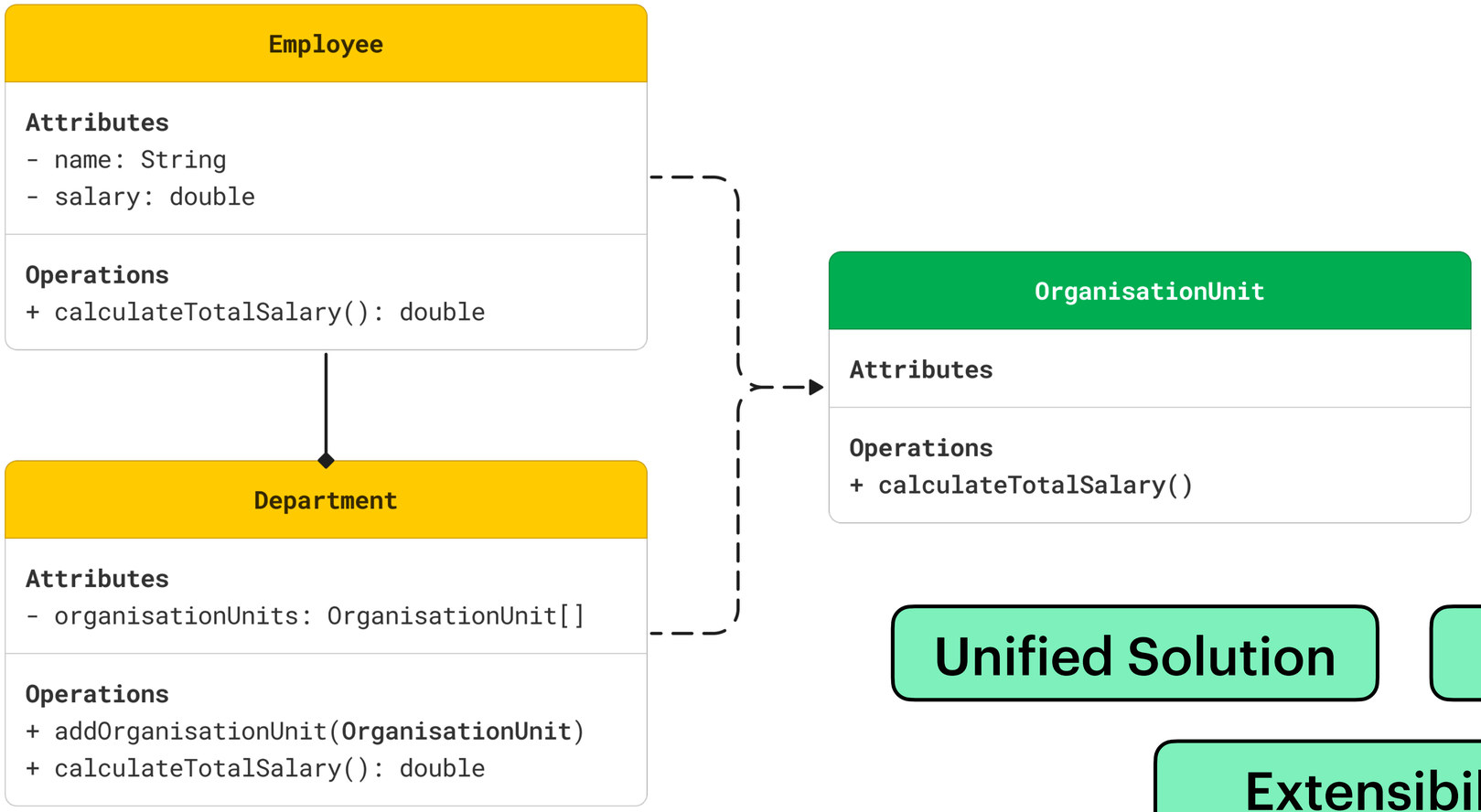
Extensibility

Composite

```java
public class Main {
    public static void main(String[] args) {
        Employee mahmoud = new Employee("Mahmoud", 1000.0);
        Employee ahmed = new Employee("Ahmed", 5000.0);
        Department department = new Department();
        department.addEmployee(mahmoud);
        department.addEmployee(ahmed);

        Department subDepartment = new Department();
        subDepartment.addEmployee(new Employee("Youssef", 5000.0));

        department.addSubDepartment(subDepartment);
        System.out.println("Total Salary: " + department.calculateTotalSalary());
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Unified Solution

Simplicity

Extensibility

© Mahmoud Youssef

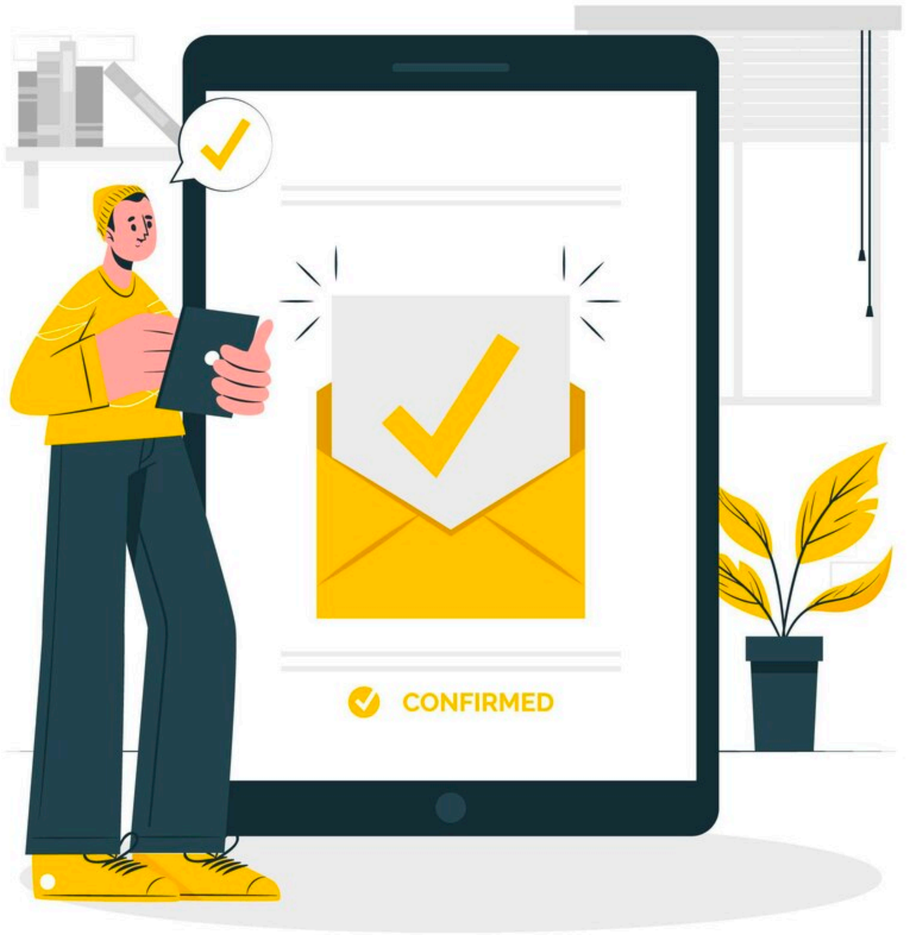
DECORATOR

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing a system for ordering beverages. Customers can order ***different types of beverages*** (e.g., coffee, tea), and they can customize their orders with various condiments (e.g., milk, sugar, whipped cream).

The challenge is to allow ***dynamic customization*** of beverages with different condiments (e.g. coffee with milk, tea with lemon and coffee with sugar and milk).



© Mahmoud Youssef

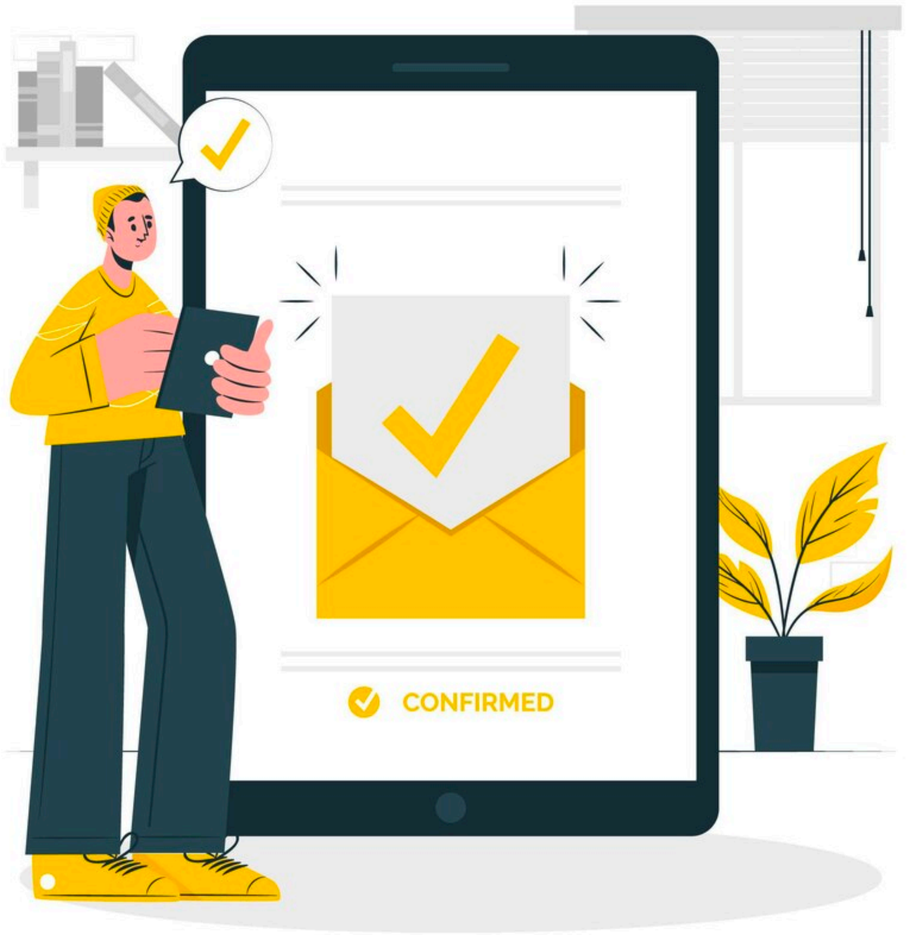
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing a system for ordering beverages. Customers can order ***different types of beverages*** (e.g., coffee, tea), and they can customize their orders with various condiments (e.g., milk, sugar, whipped cream).

The challenge is to allow ***dynamic customization*** of beverages with different condiments (e.g. coffee with milk, tea with lemon and coffee with sugar and milk).



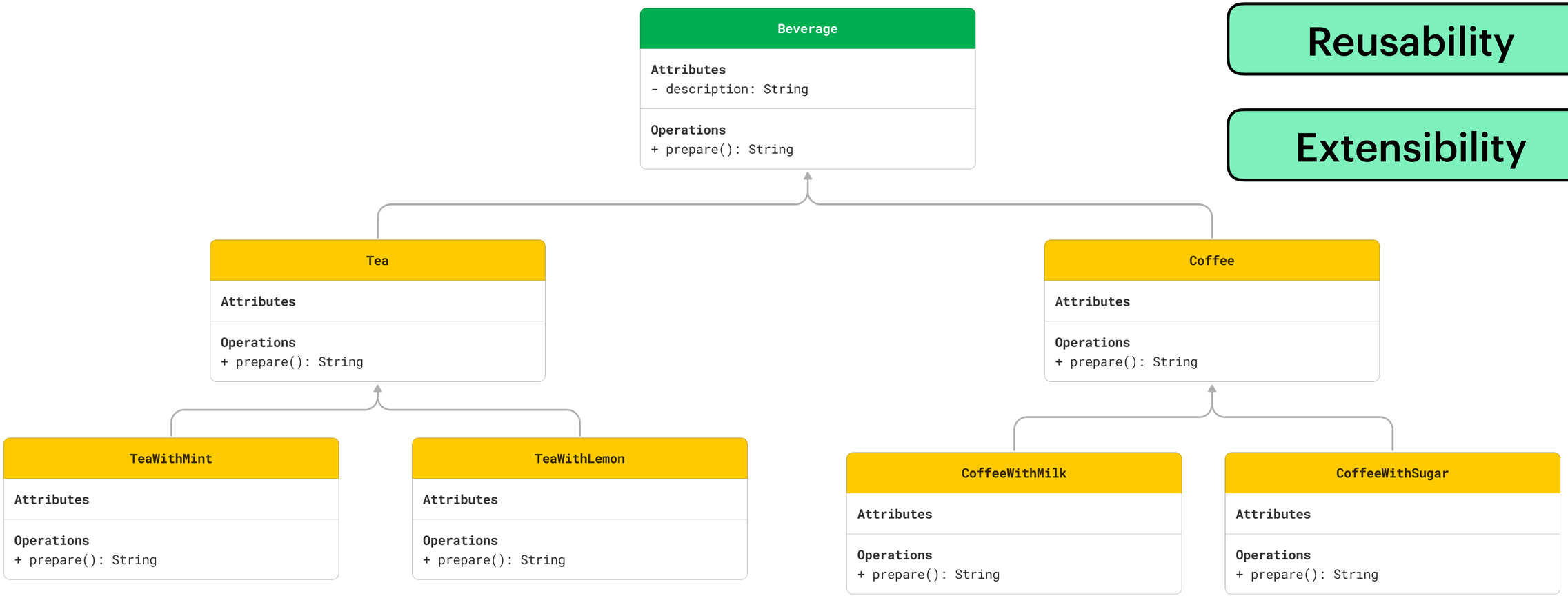
© Mahmoud Youssef

NAIVE SOLUTION

Complexity

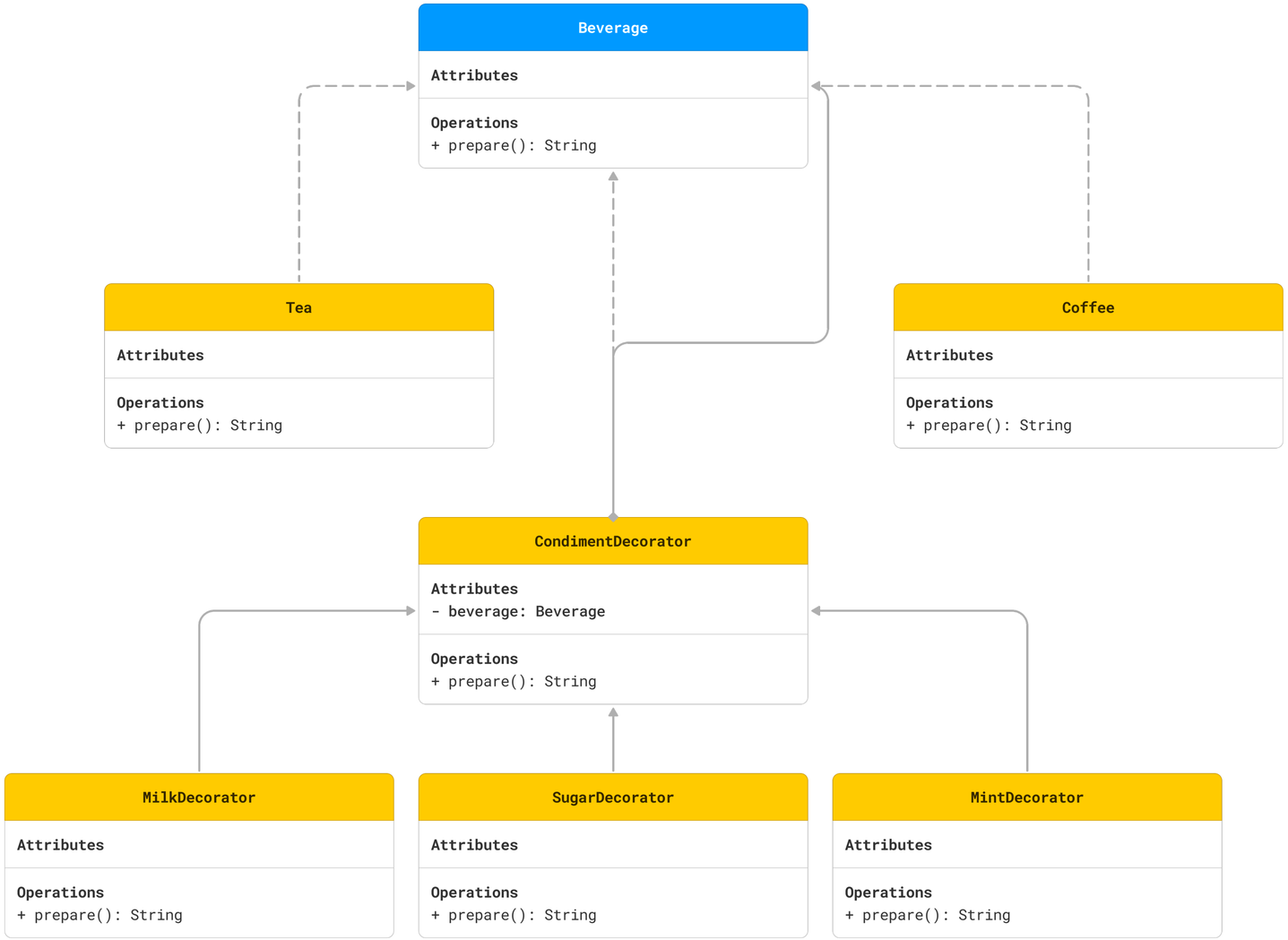
Reusability

Extensibility



© Mahmoud Youssef

ENHANCED SOLUTION



Flexibility

Reusability

Extensibility

© Mahmoud Youssef

IMPLEMENTATION

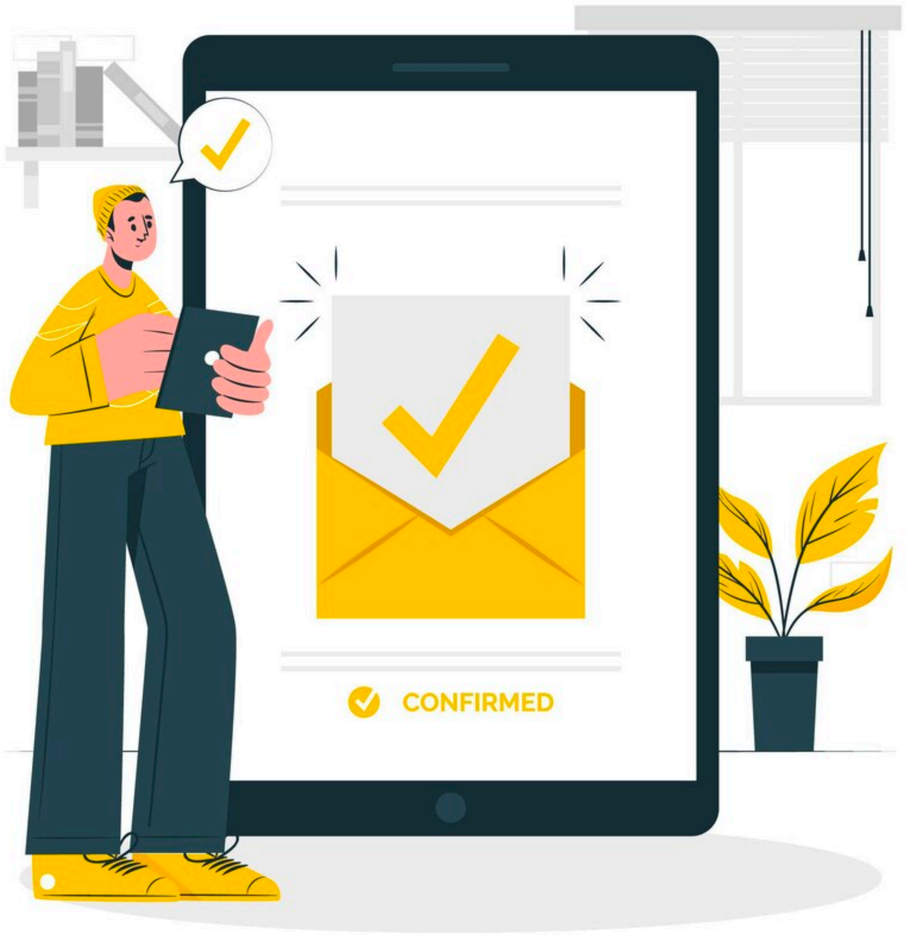
FACADE

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing an ***online travel booking system*** where customers can book flights, reserve hotel rooms, and rent cars for their trips.

The booking system involves multiple subsystems, including ***flight management, hotel reservation, car rental, and payment processing***.



© Mahmoud Youssef

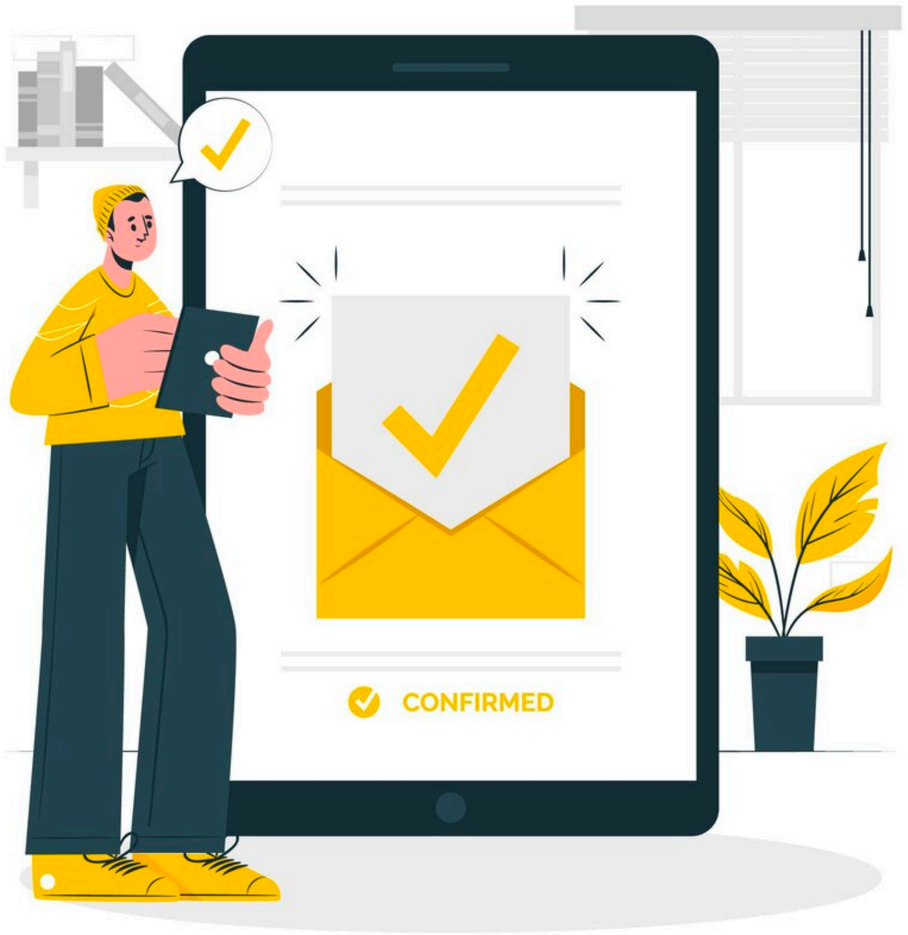
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing an ***online travel booking system*** where customers can book flights, reserve hotel rooms, and rent cars for their trips.

The booking system involves multiple subsystems, including ***flight management, hotel reservation, car rental, and payment processing***.



© Mahmoud Youssef

NAIVE SOLUTION

FlightManager

Attributes

Operations
+ bookFlight(String departure, String destination, String date): void

HotelReservation

Attributes

Operations
+ reserveRoom(String roomId, String checkInDate, String checkOutDate): void

CarRental

Attributes

Operations
+ rentCar(String location, String startDate, String endDate): void

PaymentProcessor

Attributes

Operations
+ processPayment(double amount, String paymentMethod): void

Complexity

Tightly Coupled

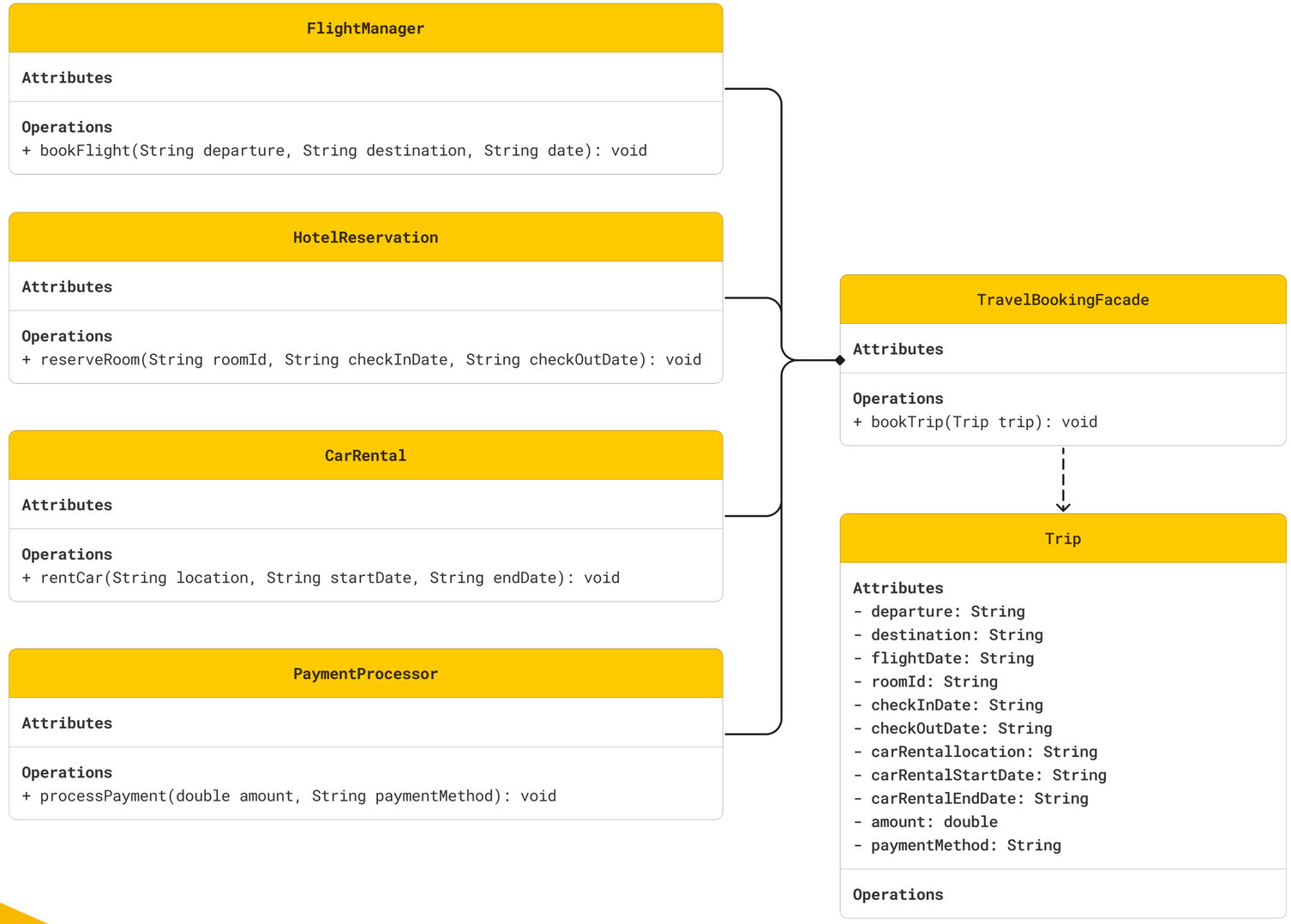
Lack of Encapsulation

Facade

```java
public class Main {
    public static void main(String[] args) {
        FlightManager flightManager = new FlightManager();
        flightManager.bookFlight("Amsterdam", "Spain", "2024-01-01T08:00:00");

        HotelReservation hotelReservation = new HotelReservation();
        hotelReservation.reserveRoom("12345", "2024-01-01T14:00:00",
                "2024-01-04T14:00:00");

        CarRental carRental = new CarRental();
        carRental.rentCar("Airport", "2024-01-01T11:00:00", "2024-01-04T15:00:00");

        PaymentProcessor paymentProcessor = new PaymentProcessor();
        paymentProcessor.processPayment(1000.0, "VISA");
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Simplicity

Encapsulation

Abstraction

© Mahmoud Youssef

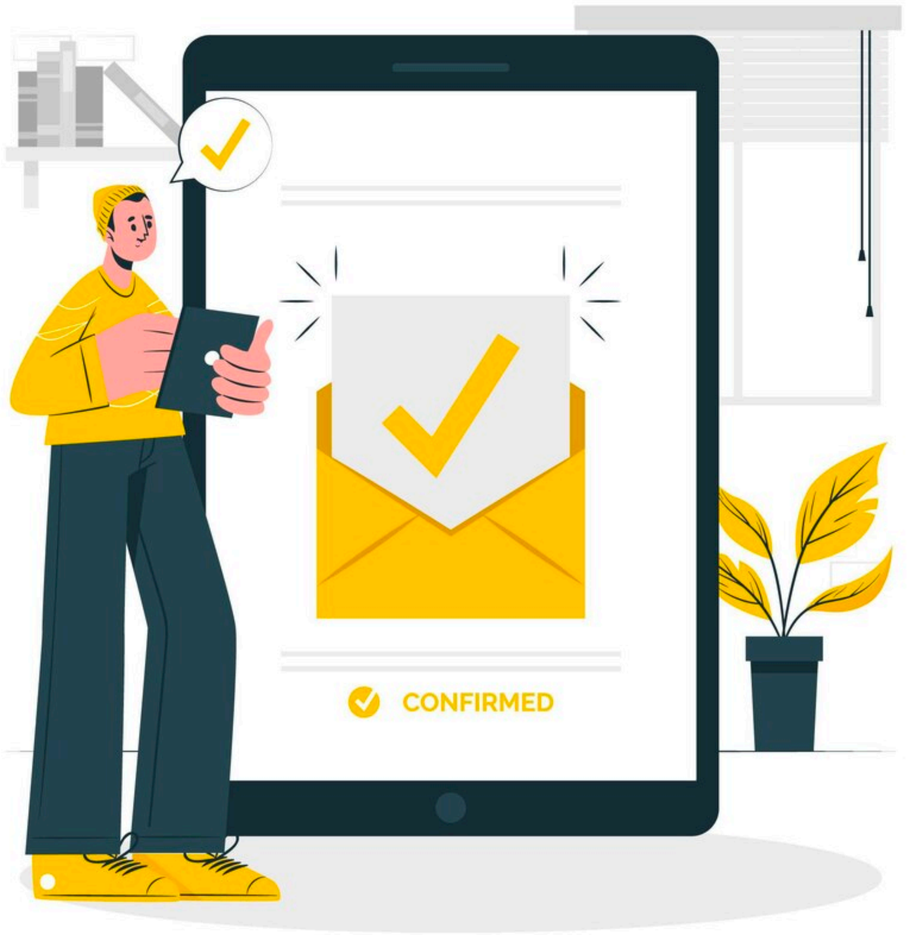
PROXY

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a web application that ***interacts with various remote APIs to fetch data***. However, fetching data from remote APIs can be time-consuming and resource-intensive, especially for frequently accessed data.

Additionally, you want to ***reduce the load on the remote APIs*** and improve the responsiveness of your application.



© Mahmoud Youssef

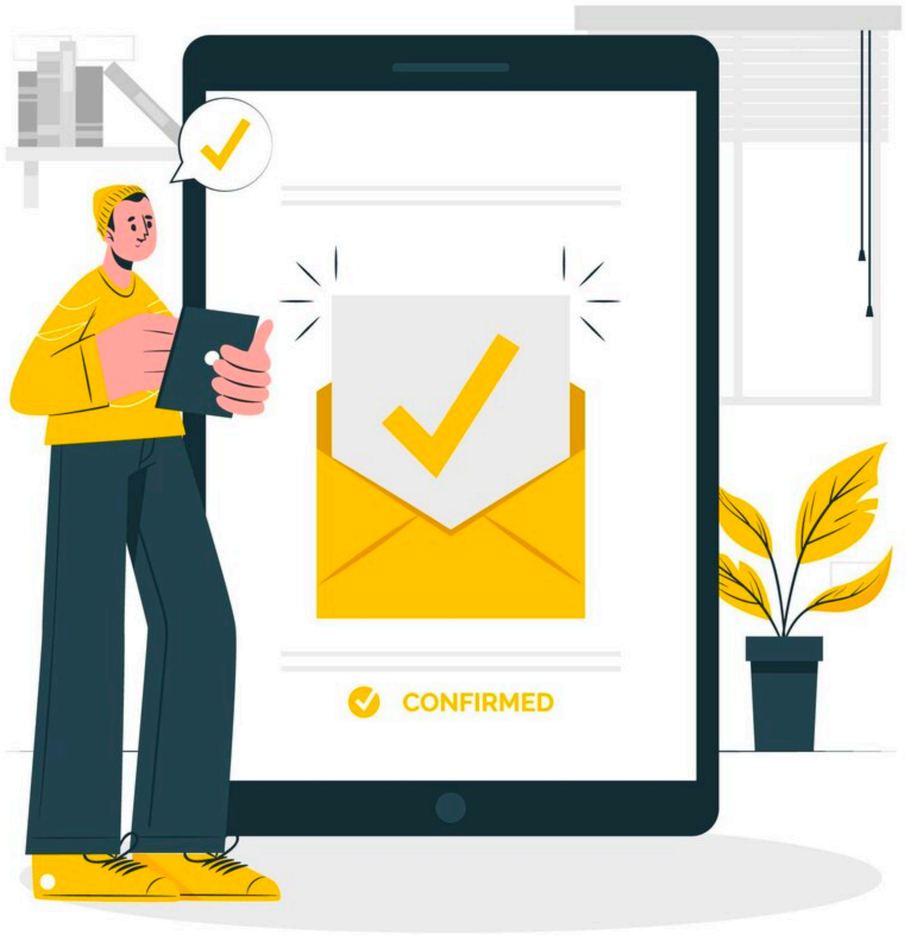
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a web application that ***interacts with various remote APIs to fetch data***. However, fetching data from remote APIs can be time-consuming and resource-intensive, especially for frequently accessed data.

Additionally, you want to ***reduce the load on the remote APIs*** and improve the responsiveness of your application.



© Mahmoud Youssef

NAIVE SOLUTION

DummyThirdPartyApiService

Attributes
- cachingLayer: Map<String, String>

Operations
+ getAllProducts(): String
+ getAllRecipes(): String

Tightly Coupled

Reusability

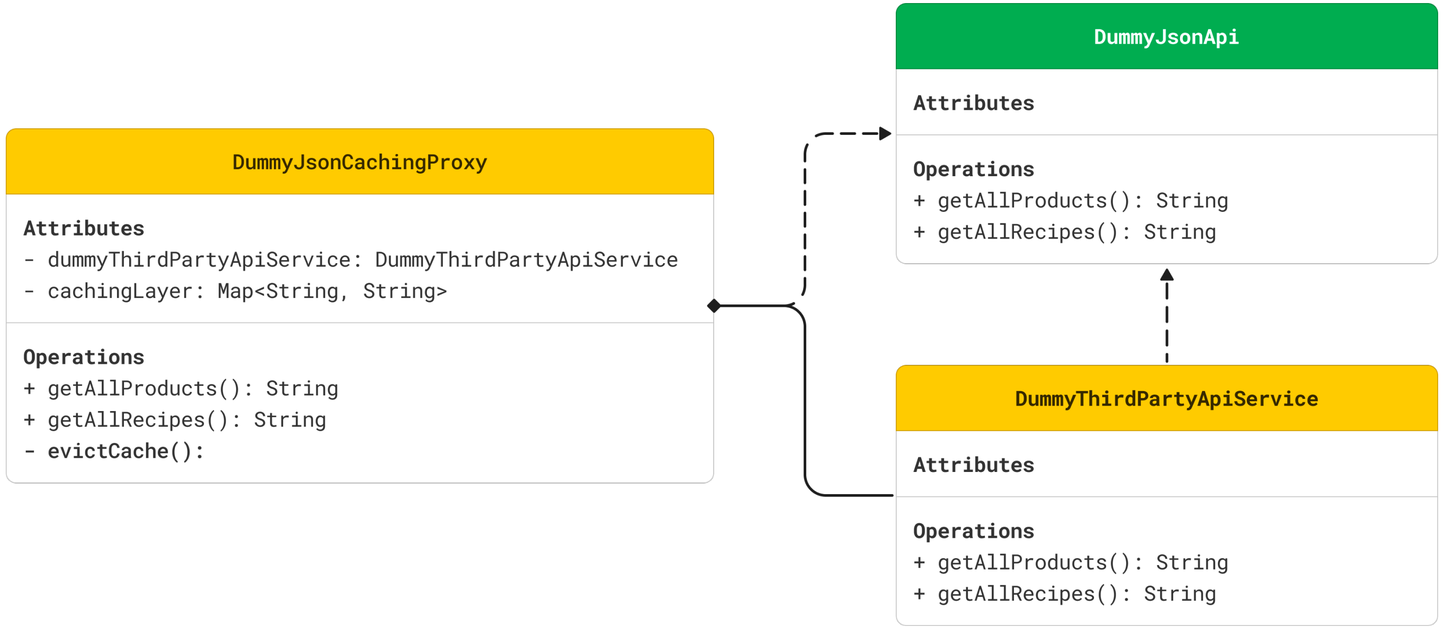
Maintainability

Student

```
public static void main(String[] args) {
        DummyThirdPartyApiService dummyService = new DummyThirdPartyApiService();
        String productsResponse = dummyService.getAllProducts();
        String recipesResponse = dummyService.getAllRecipes();
        System.out.println(productsResponse);
        System.out.println(recipesResponse);
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Separation of Concerns

Extensibility

Dynamic Configuration

© Mahmoud Youssef

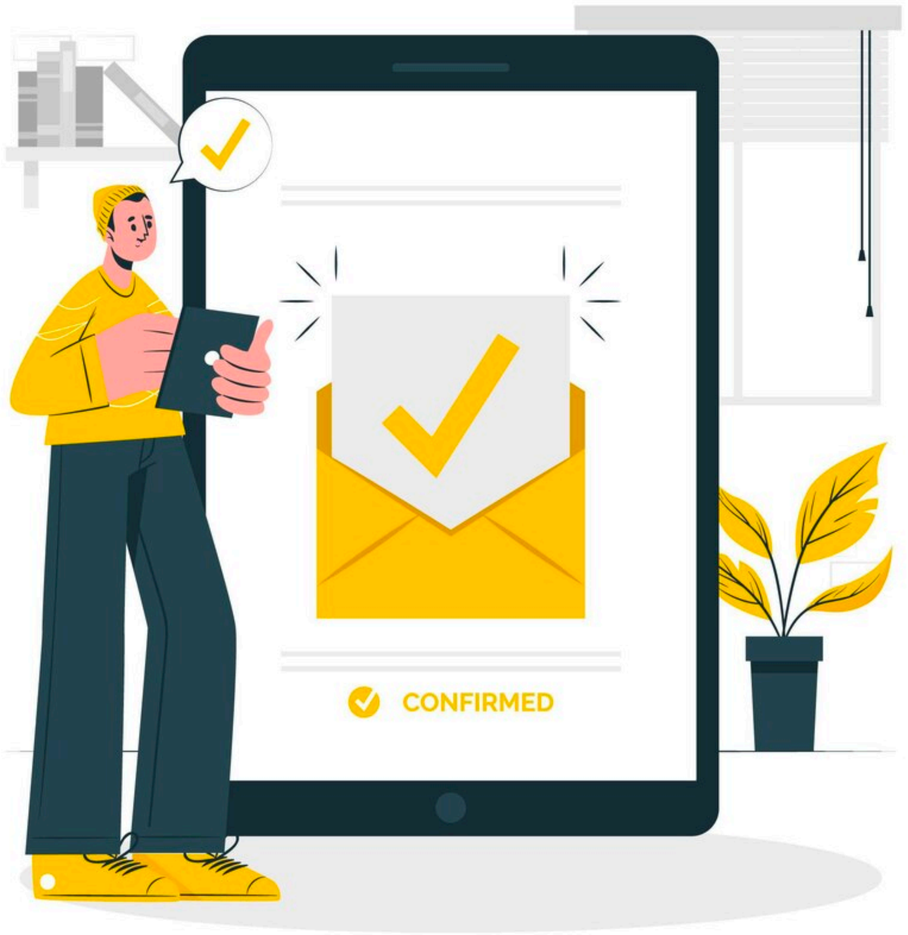
FLYWEIGHT

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a game which consists of a ***single player and enemies***. Both of them have some common attributes like ***health bar, weapon and attack power***.

There is two types of enemies (Weak, Strong) each type has different attack power and health bar. Weapons can be also changed from both sides (player, enemies) and some of them ***provide extra attack power points***.



© Mahmoud Youssef

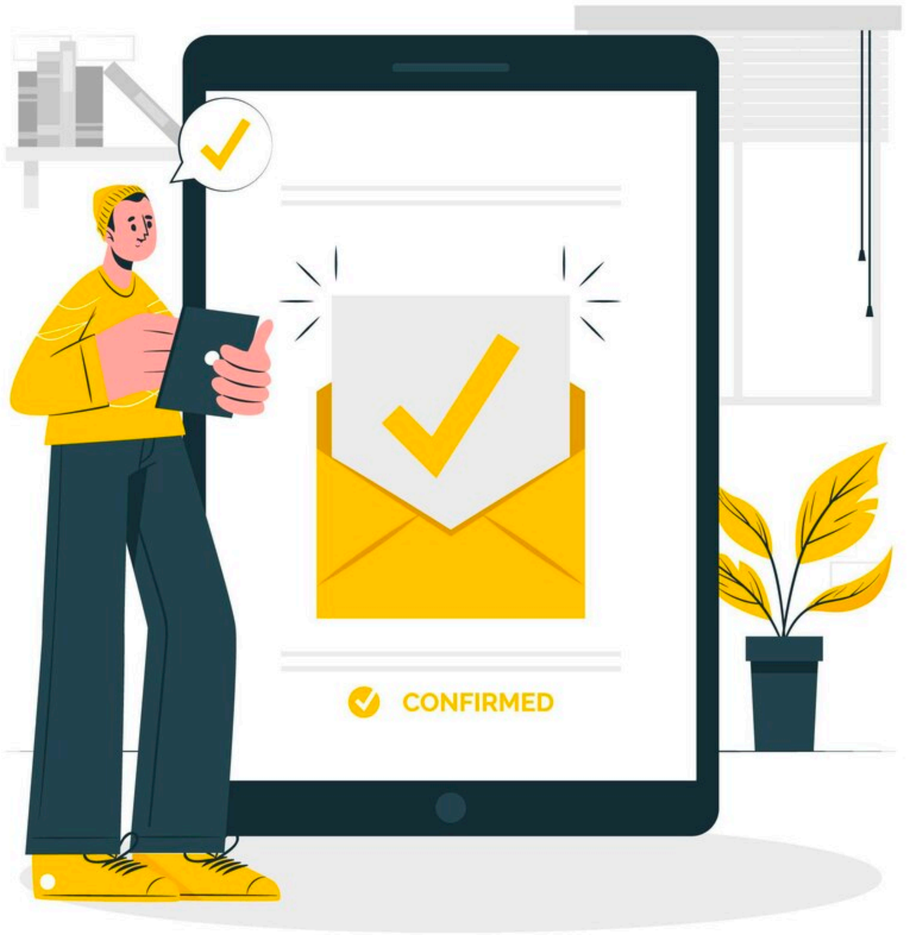
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

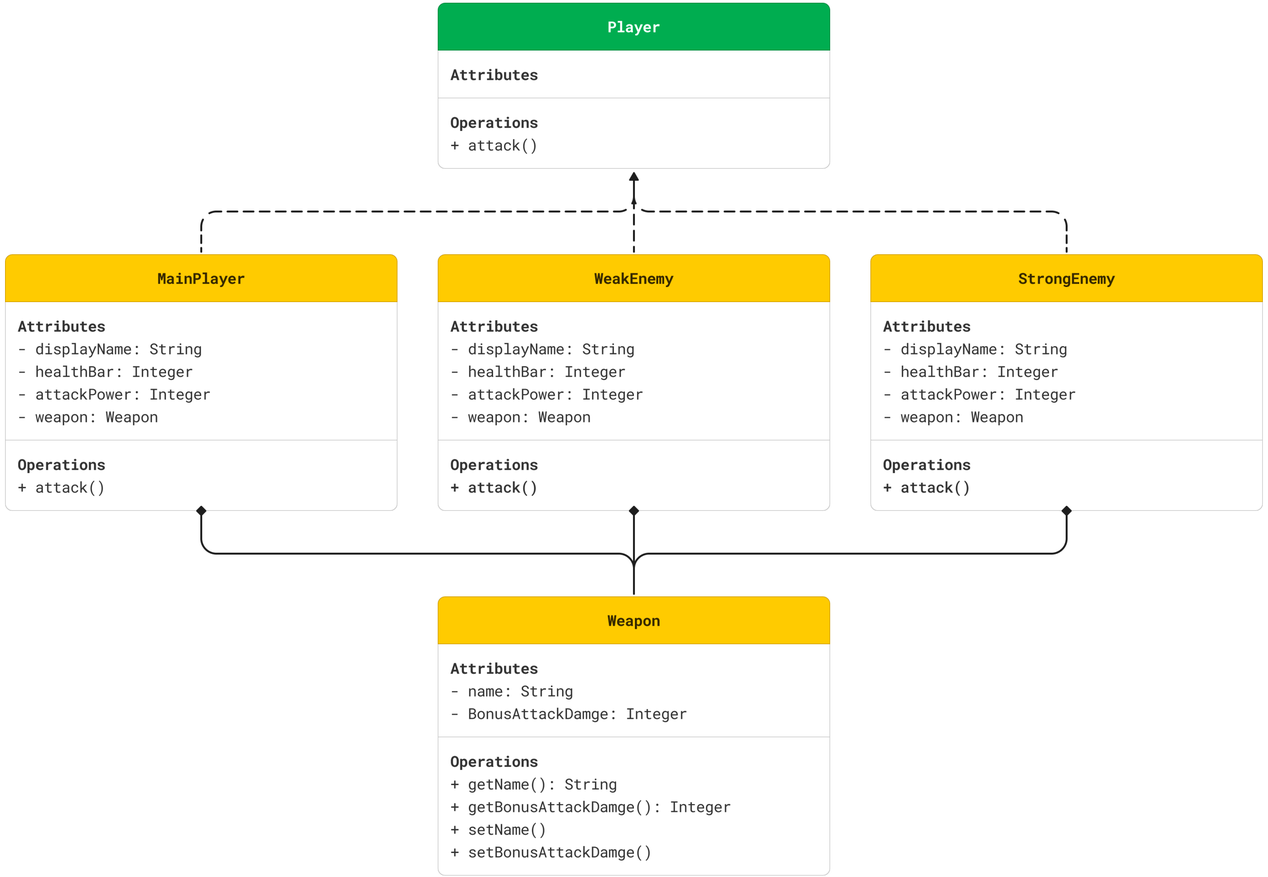
You're developing a game which consists of a ***single player and enemies***. Both of them have some common attributes like ***health bar, weapon and attack power***.

There is two types of enemies (Weak, Strong) each type has different attack power and health bar. Weapons can be also changed from both sides (player, enemies) and some of them ***provide extra attack power points***.



© Mahmoud Youssef

NAIVE SOLUTION



High Memory Usage

Redundant Data

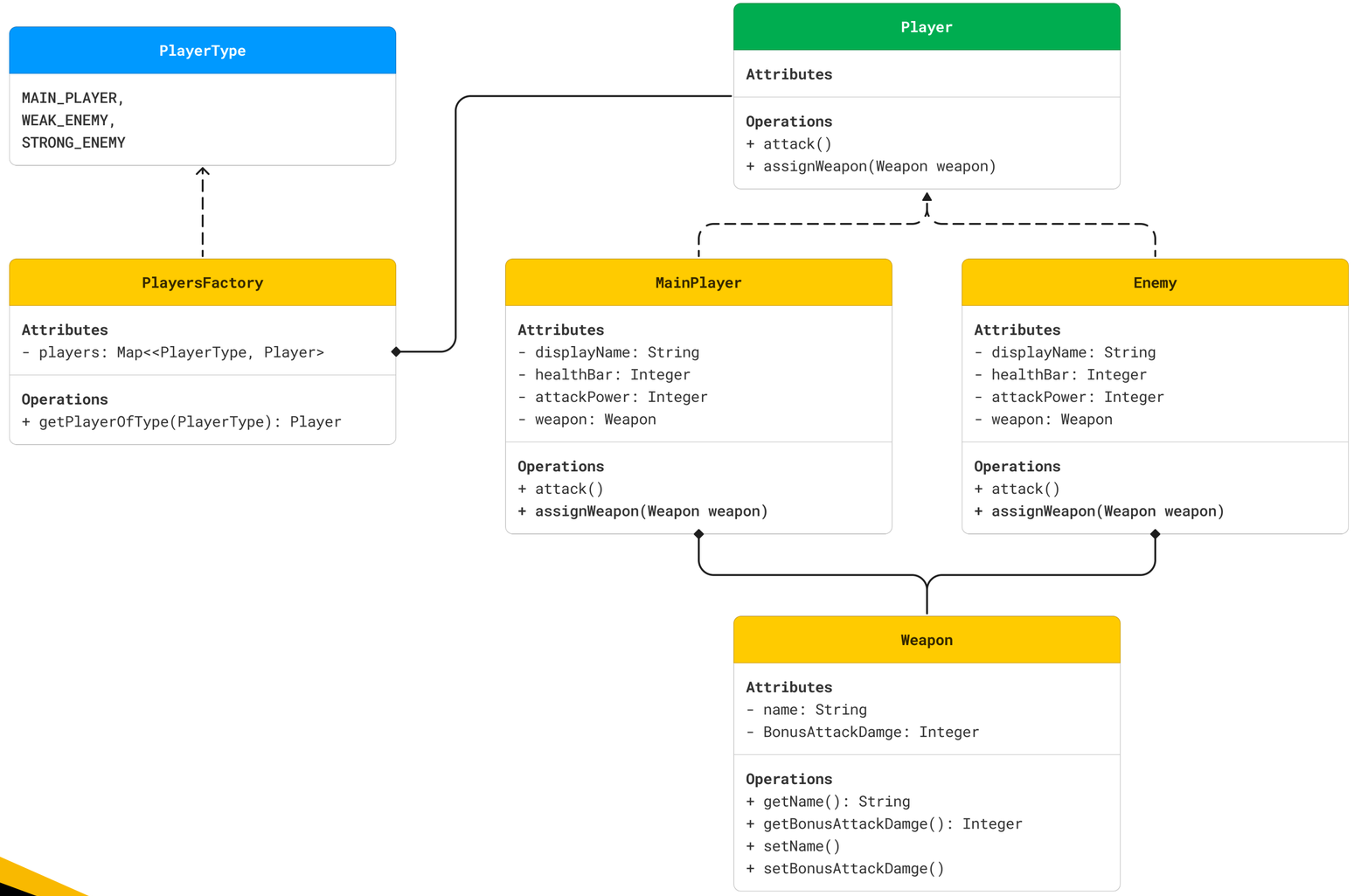
Performance Impact

Flyweight

```java
public class Main {
    public static void main(String[] args) {
        Player mainPlayer = new MainPlayer(new Weapon("AK47", 0));
        mainPlayer.attack();

        Player weakEnemy = new WeakEnemy();
        weakEnemy.attack();

        Player weakEnemy2 = new WeakEnemy();
        weakEnemy2.attack();

        Player weakEnemy3 = new WeakEnemy();
        weakEnemy3.attack();
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Memory Efficiency

Flexibility

Scalability

© Mahmoud Youssef

BEHAVIORAL

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

BEHAVIORAL

- State
- Observer
- Command
- Strategy
- Visitor
- Mediator
- Iterator
- Memento
- Chain of Responsibility
- Template Method

© Mahmoud Youssef

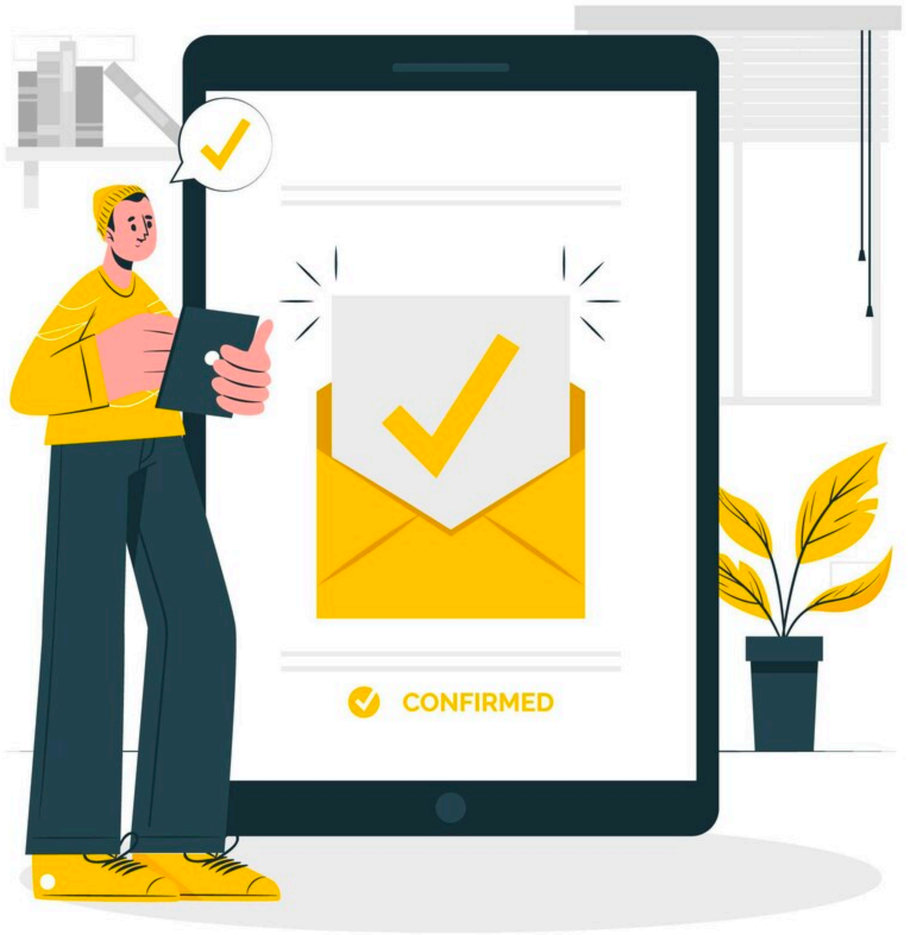
STRATEGY

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

you're developing an e-commerce platform where different products may have varying pricing logic based on factors ***such as customer segment***. We have regular, gold and premium membership.

Once product price is calculated, then at the checkout the customer could ***pay with different methods*** like PayPal, VisaCard and BankTransfer. Each payment method ***vary in it's logic of processing the payment and conducting required fees***.



© Mahmoud Youssef

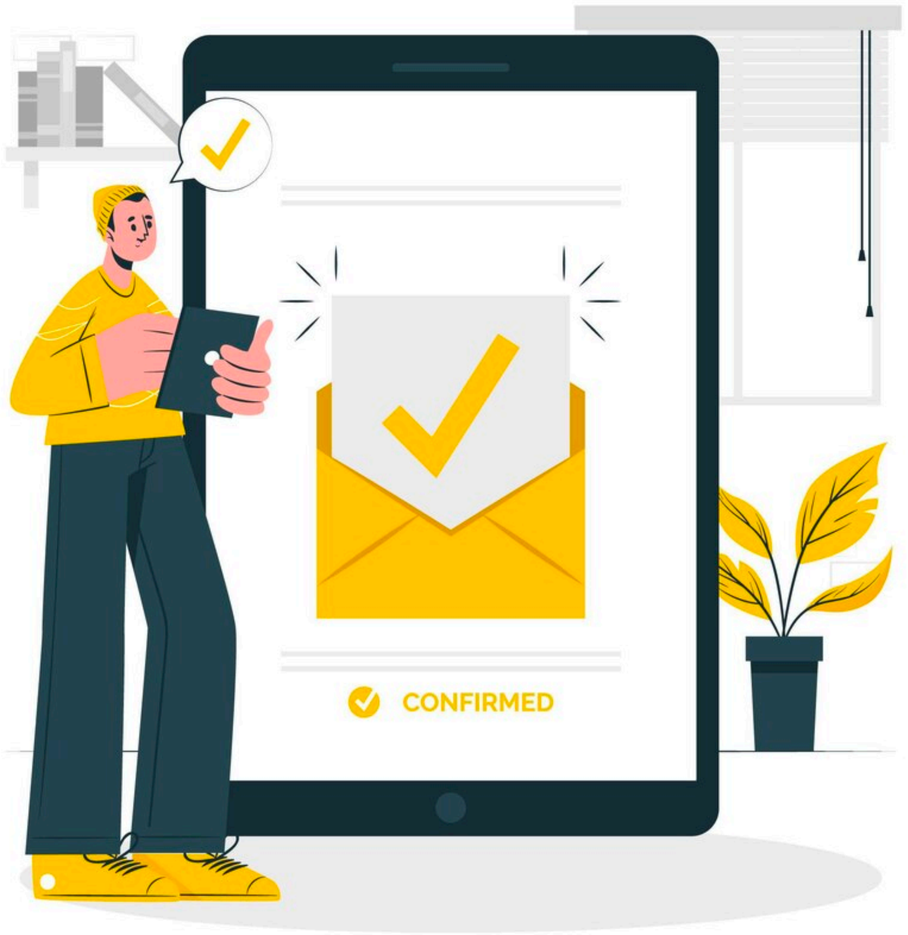
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

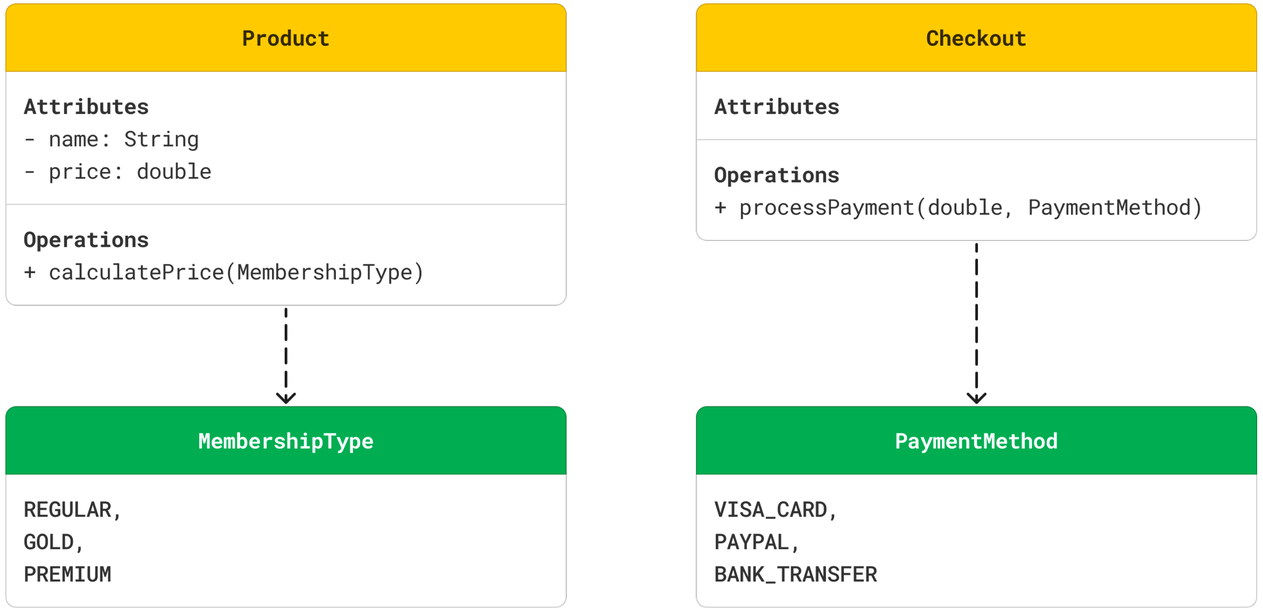
you're developing an e-commerce platform where different products may have varying pricing logic based on factors ***such as customer segment***. We have regular, gold and premium membership.

Once product price is calculated, then at the checkout the customer could ***pay with different methods*** like PayPal, VisaCard and BankTransfer. Each payment method ***vary in it's logic of processing the payment and conducting required fees***.



© Mahmoud Youssef

NAIVE SOLUTION



Reusability

Maintenance

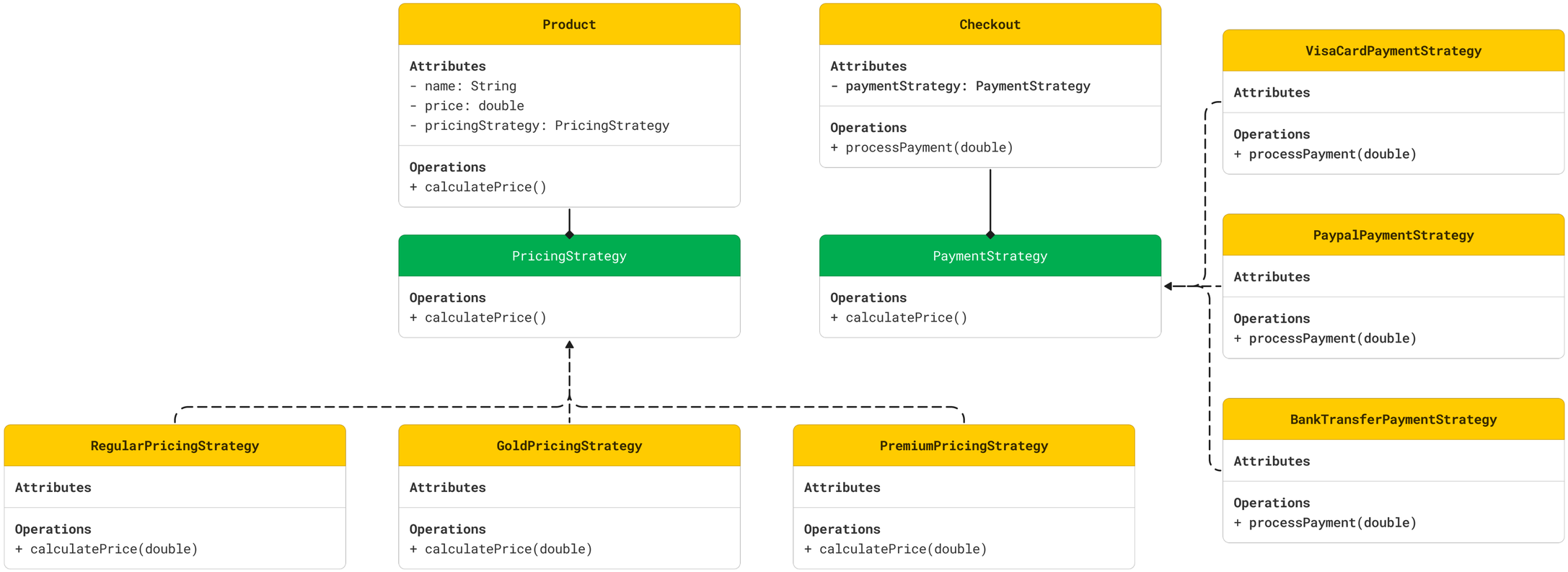
Extensibility

Strategy

```java
public class Main {
    public static void main(String[] args) {
        Product wallet = new Product("Wallet", 200.0);
        double walletPrice = wallet.calculatePrice(MembershipType.REGULAR);
        System.out.println(walletPrice);

        Product jacket = new Product("Jacket", 100.0);
        double jacketPrice = jacket.calculatePrice(MembershipType.GOLD);
        System.out.println(jacketPrice);

        Product mobile = new Product("Mobile", 1000.0);
        double mobilePrice = mobile.calculatePrice(MembershipType.PREMIUM);
        System.out.println(mobilePrice);

        Checkout checkout = new Checkout();
        checkout.processPayment(walletPrice, PaymentMethod.VISA_CARD);
        checkout.processPayment(jacketPrice, PaymentMethod.PAYPAL);
        checkout.processPayment(mobilePrice, PaymentMethod.BANK_TRANSFER);
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Reusability **Maintenance** **Extensibility**

© Mahmoud Youssef

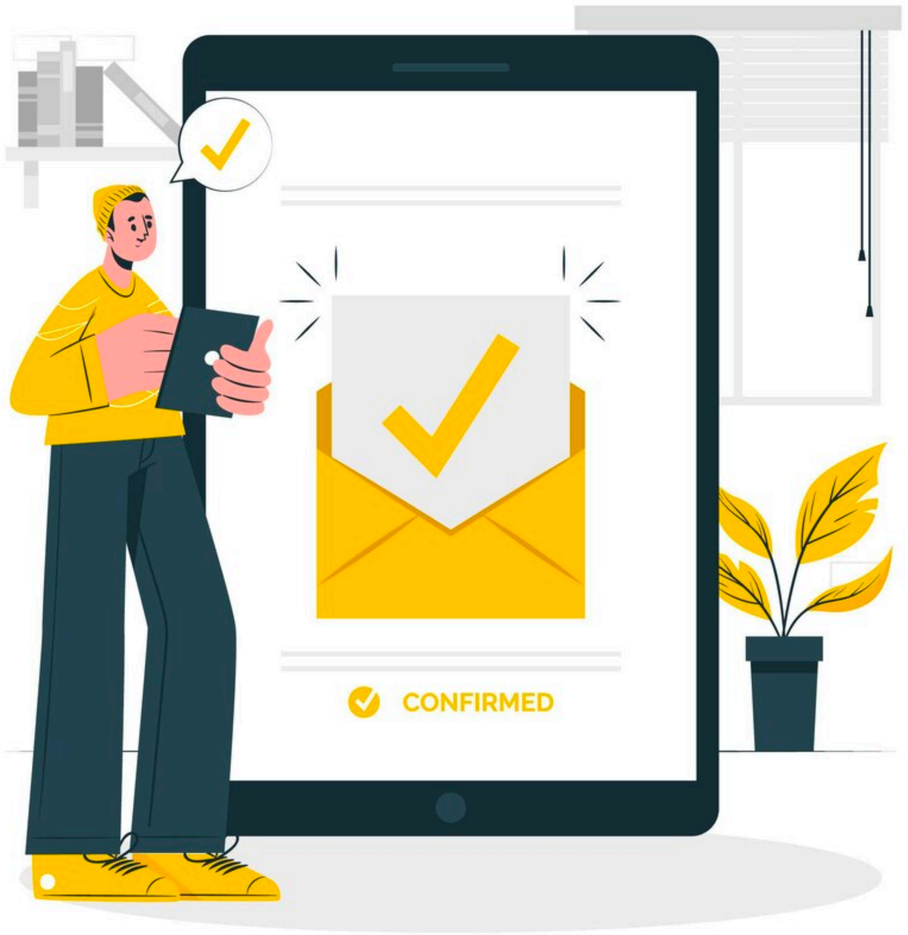
TEMPLATE METHOD

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

you're developing an a Recruiting Application, which involves ***uploading CV files in different formats*** like PDF, image and .doc. We need to generate a report based on those CVs to know which ones to filter them out.

Generating a report requires some ***common functionality as reading files, extracting data, analyzing data and finally generating*** the report.



© Mahmoud Youssef

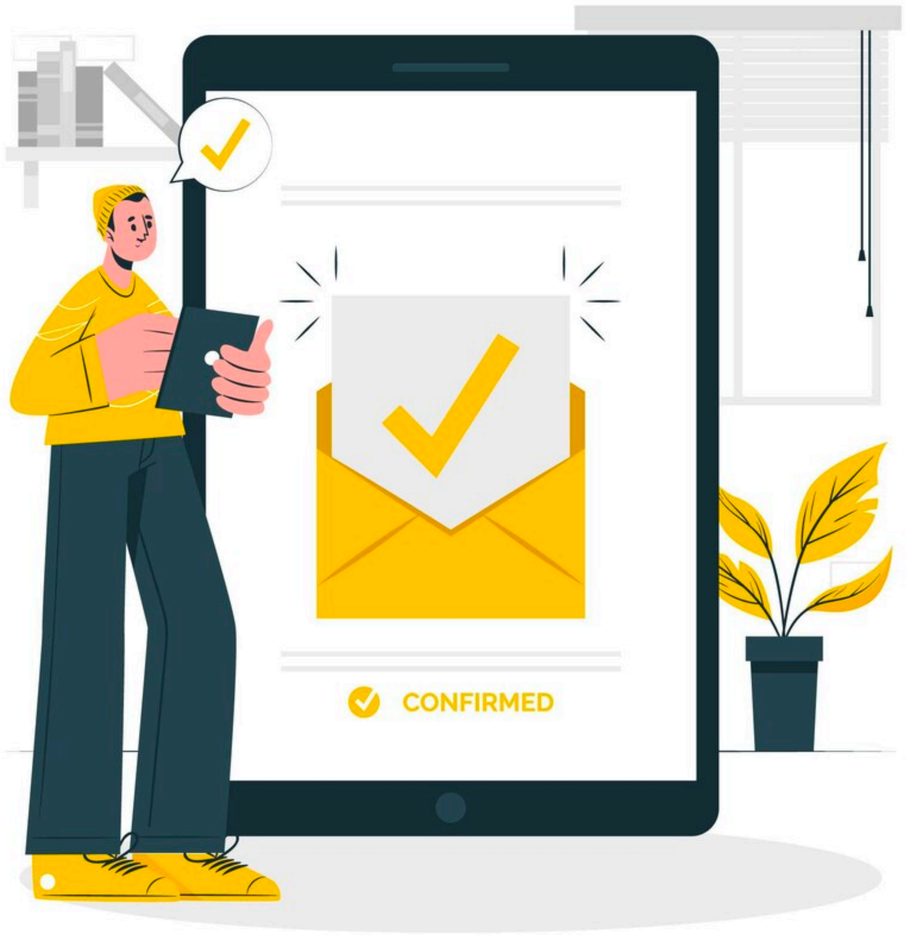
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

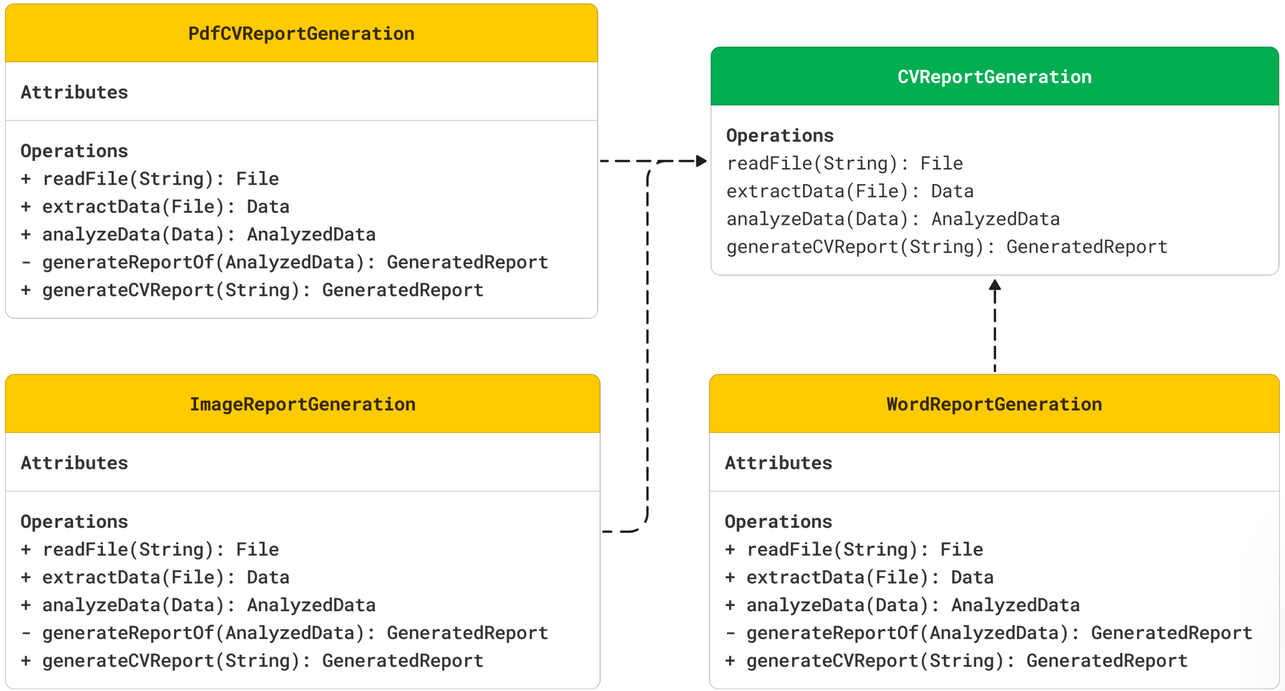
you're developing an a Recruiting Application, which involves ***uploading CV files in different formats*** like PDF, image and .doc. We need to generate a report based on those CVs to know which ones to filter them out.

Generating a report requires some ***common functionality as reading files, extracting data, analyzing data and finally generating*** the report.



© Mahmoud Youssef

NAIVE SOLUTION



Maintenance

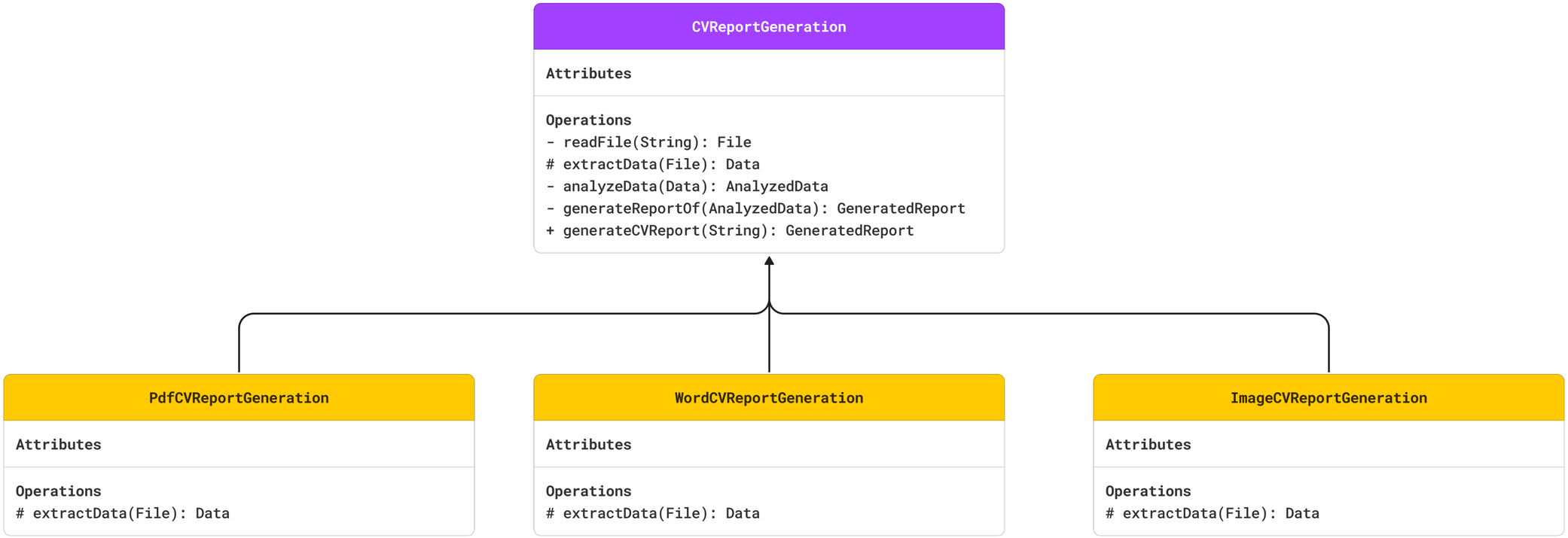
Extensibility

TemplateMethod

```java
public class Main {
    public static void main(String[] args) {
        CVReportGeneration pdfCvReportGeneration = new PdfCVReportGeneration();
        GeneratedReport pdfGeneratedReport =
pdfCvReportGeneration.generateCVReport("/tmp/cv-001.pdf");
        System.out.println(pdfGeneratedReport.isPassed());

        CVReportGeneration wordCvReportGeneration = new WordCVReportGeneration();
        GeneratedReport wordGeneratedReport =
wordCvReportGeneration.generateCVReport("/tmp/cv-001.docx");
        System.out.println(wordGeneratedReport.isPassed());

        CVReportGeneration imageCvReportGeneration = new ImageCVReportGeneration();
        GeneratedReport imageGeneratedReport =
imageCvReportGeneration.generateCVReport("/tmp/cv-001.jpeg");
        System.out.println(imageGeneratedReport.isPassed());
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Reusability **Maintenance** **Extensibility**

© Mahmoud Youssef

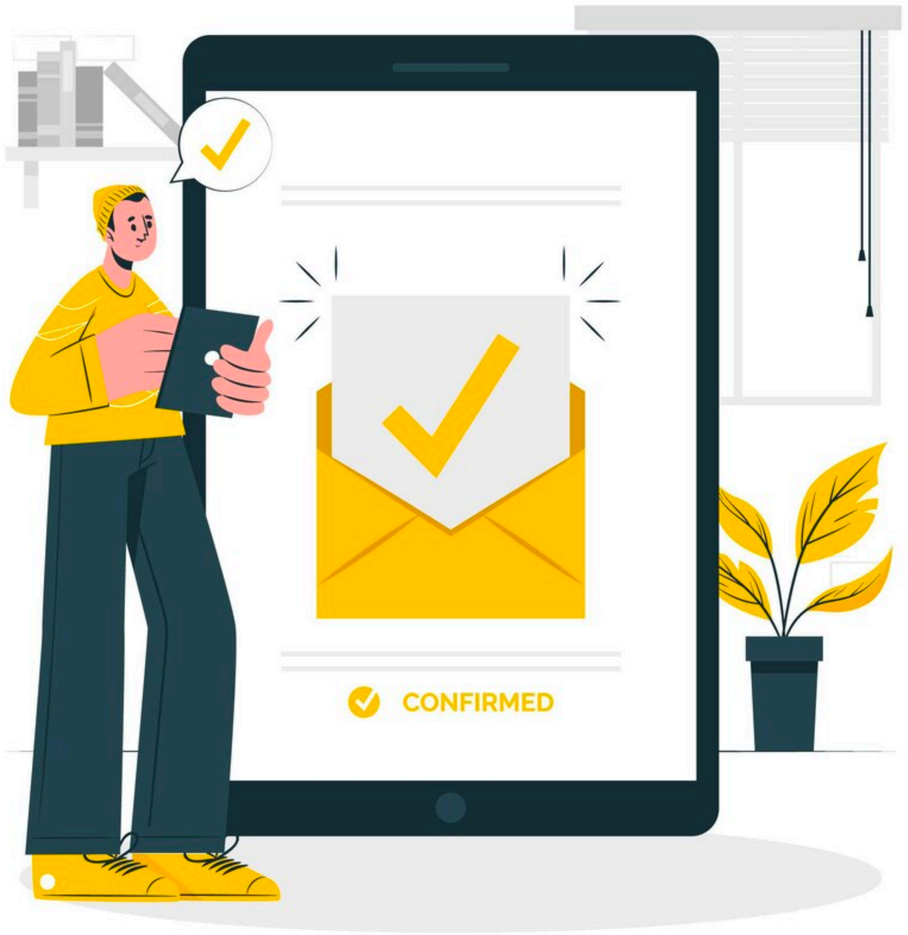
MEMENTO

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a new ***Text Editor*** and you want to provide the ***functionality of the undo and redo mechanism***.

The undo should be ***able to restore the previous state*** and the <u>redo should be able to do the vise-versa</u>.



© Mahmoud Youssef

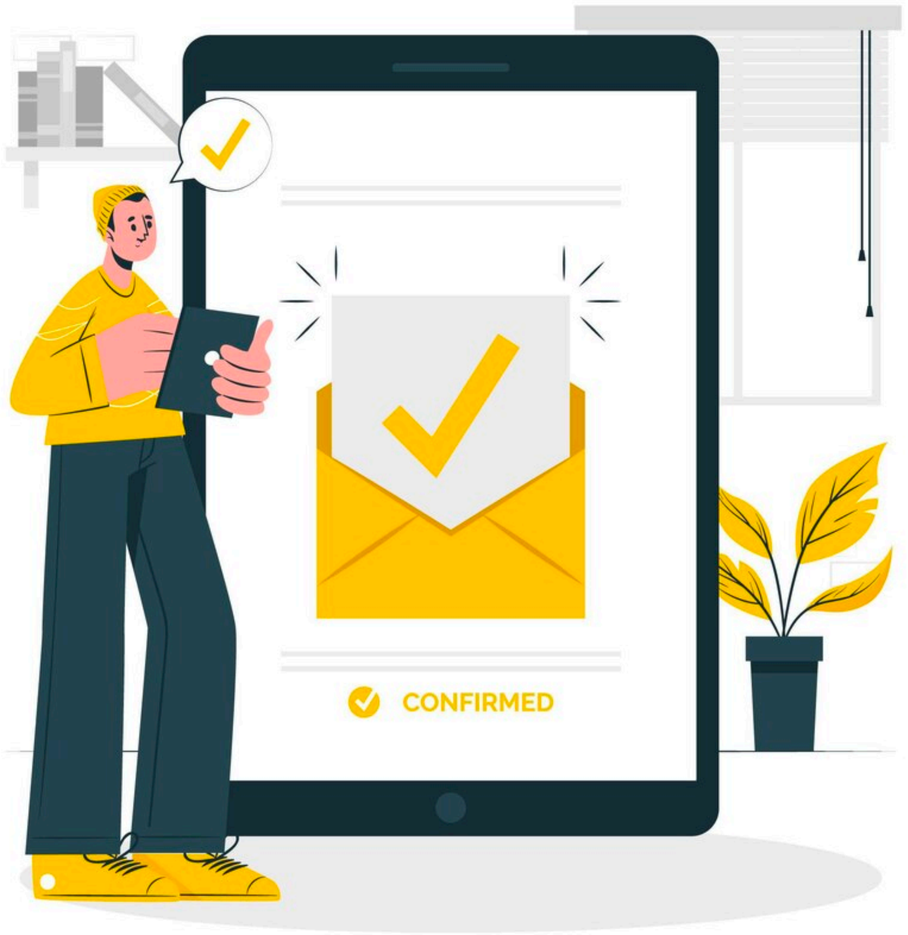
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a new ***Text Editor*** and you want to provide the ***functionality of the undo and redo mechanism***.

The undo should be ***able to restore the previous state*** and the <u>redo should be able to do the vise-versa</u>.



© Mahmoud Youssef

NAIVE SOLUTION

TextEditor

Attributes

- content: String
- prevState: String[]
- nextState: String[]

Operations

+ getContent(): String
+ setContent(String)
+ save()
+ undo(): String
+ redo(): String

Complexity

Flexibility

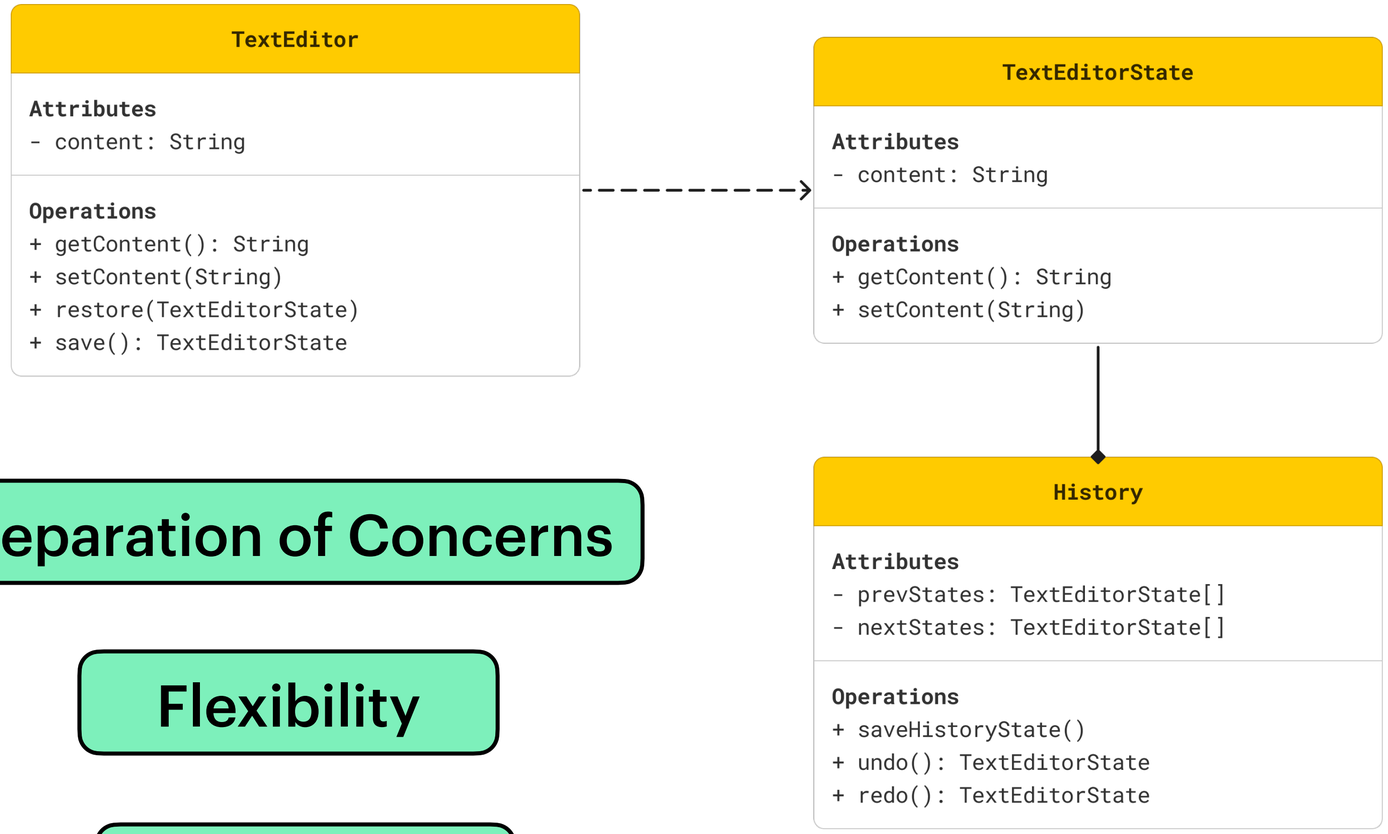
Reusability

Memento

```java
public class Main {
    public static void main(String[] args) {
        TextEditor textEditor = new TextEditor();

        textEditor.setContent("Hello, ");
        textEditor.save();

        textEditor.setContent("World!");
        textEditor.save();

        textEditor.setContent("Welcome to Design Pattern Course!");
        textEditor.undo();
        textEditor.undo();
        textEditor.redo();

        System.out.println(textEditor.getContent());
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Separation of Concerns

Flexibility

Reusability

© Mahmoud Youssef

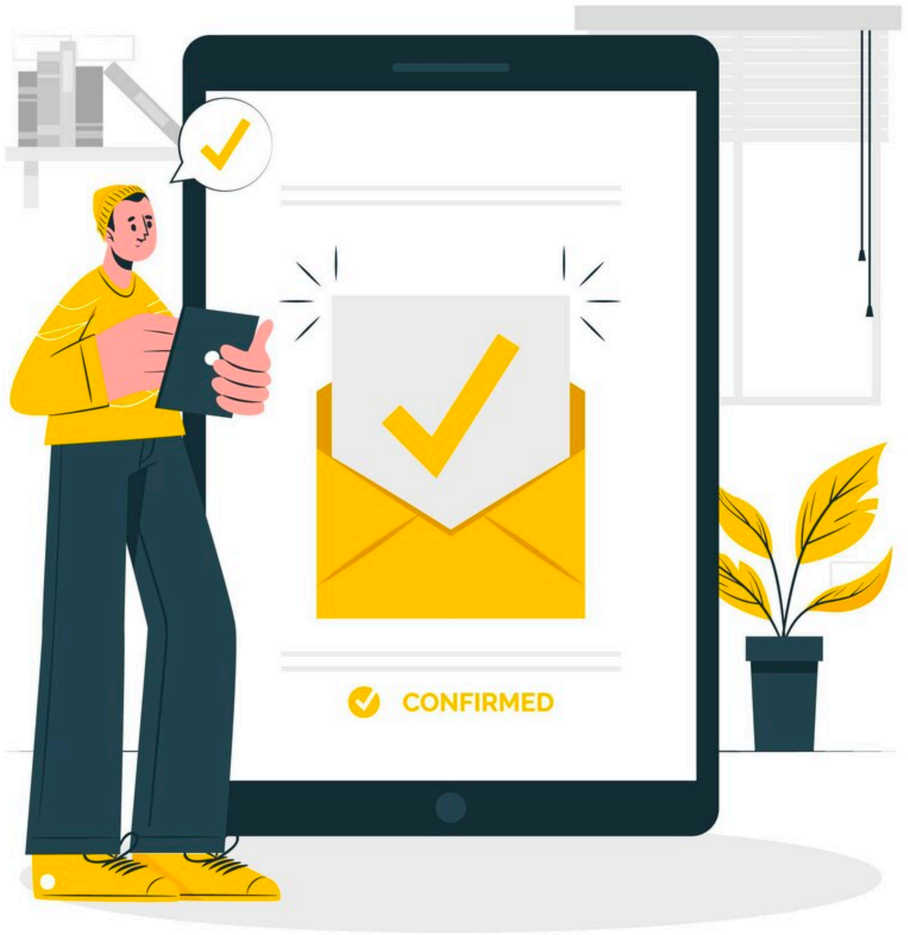
VISITOR

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a software system for ***managing employee schedules across different departments and shifts***. In this system, various operations need to be performed on different types of schedules, such as ***generating reports and calculating overtime.***

You are required to add a new feature that ***manage the leave requests on each schedule type*** <u>without touching the codebase as much as you can as it's in Production</u>. For simplicity let's say we have ***three types of schedules*** e.g (day shift, night shift and remote work).



© Mahmoud Youssef

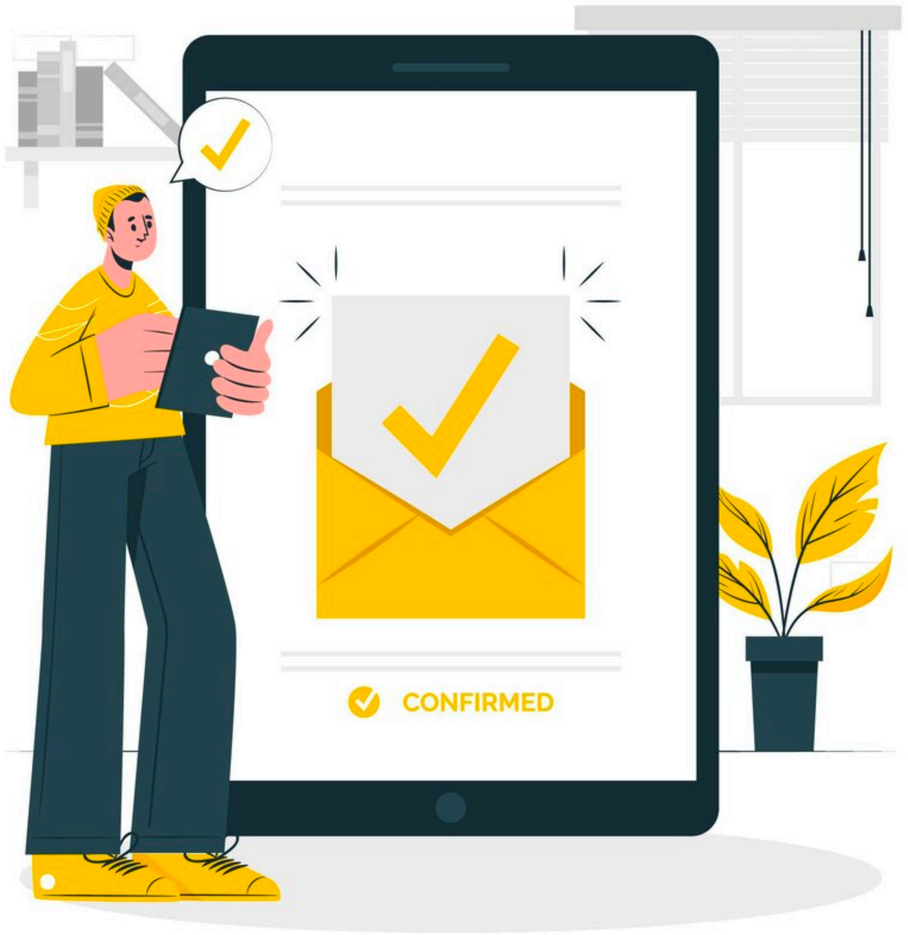
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

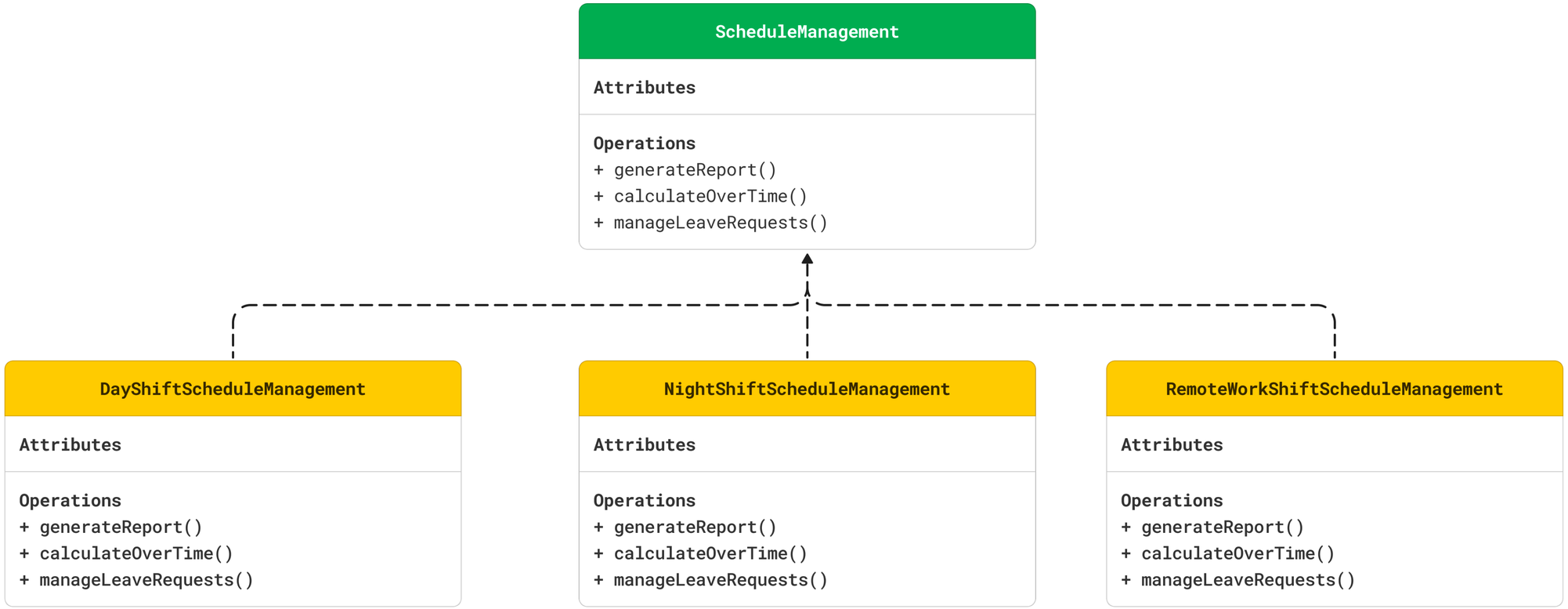
You're developing a software system for ***managing employee schedules across different departments and shifts***. In this system, various operations need to be performed on different types of schedules, such as ***generating reports and calculating overtime.***

You are required to add a new feature that ***manage the leave requests on each schedule type*** <u>without touching the codebase as much as you can as it's in Production</u>. For simplicity let's say we have ***three types of schedules*** e.g (day shift, night shift and remote work).



© Mahmoud Youssef

NAIVE SOLUTION

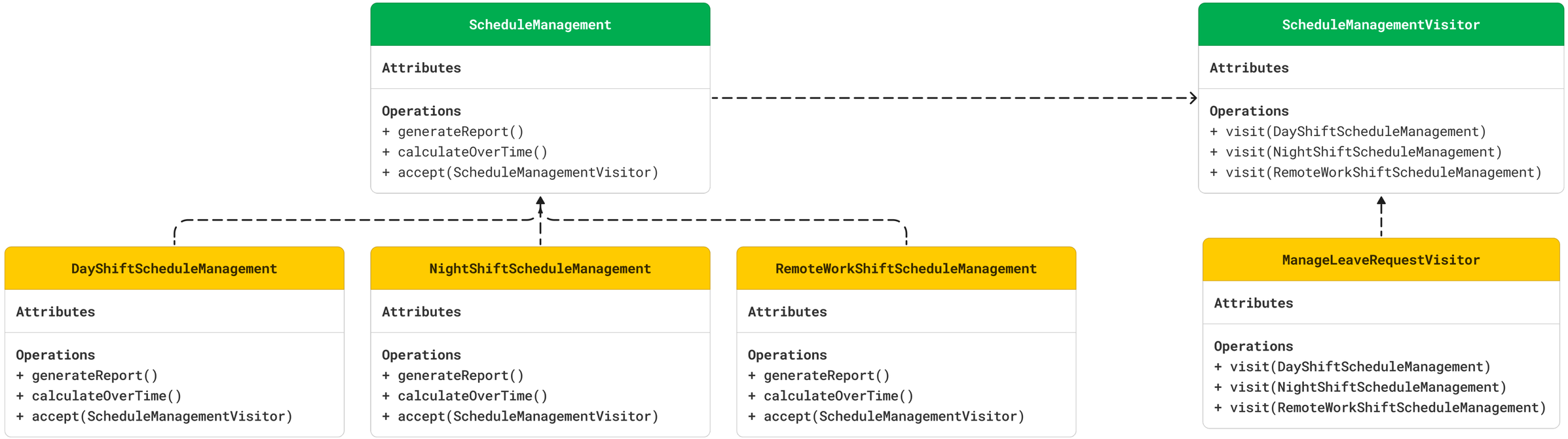


Duplication

Limited Extensibility

© Mahmoud Youssef

ENHANCED SOLUTION



Maintainability **Extensibility**

© Mahmoud Youssef

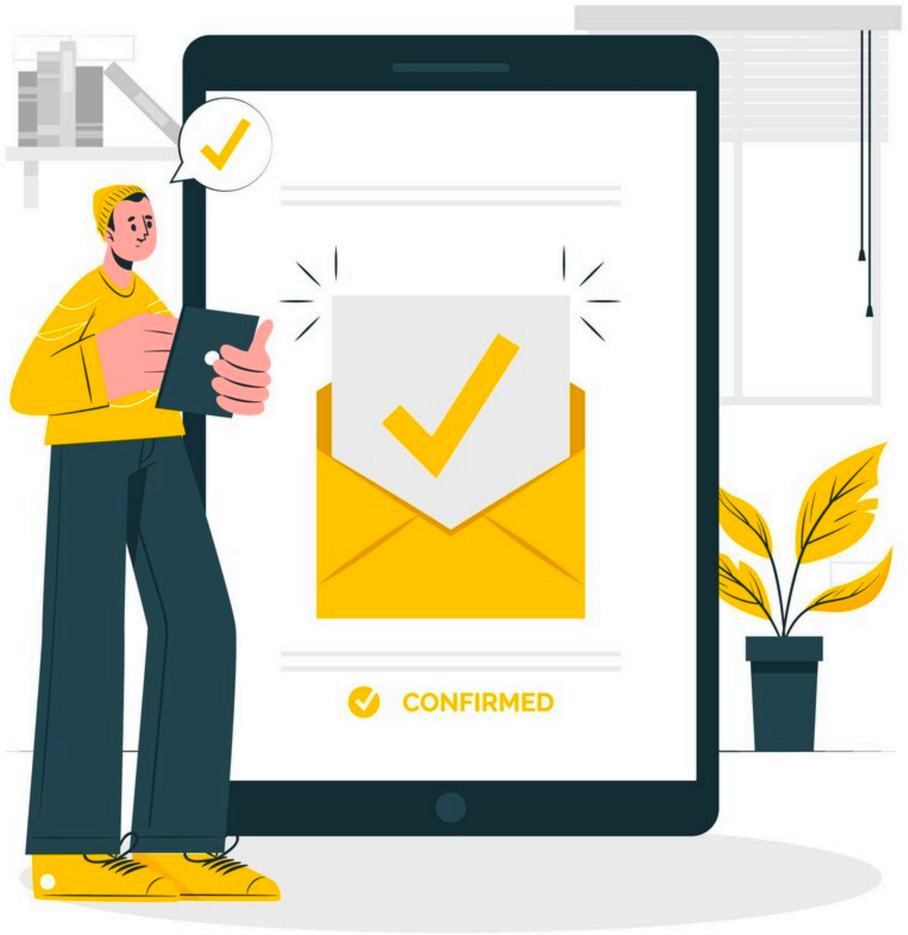
ITERATOR

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a software system that manages all ***employees information inside the company***. This system should be re-used with different IT departments as a library that should be available to be integrated with.

You are required to add the functionality that allow consumer of your library to ***iterate through employees direct reports hierarchy, co-workers and sub-ordinates with the following names e.g getNext and hasNext.***



© Mahmoud Youssef

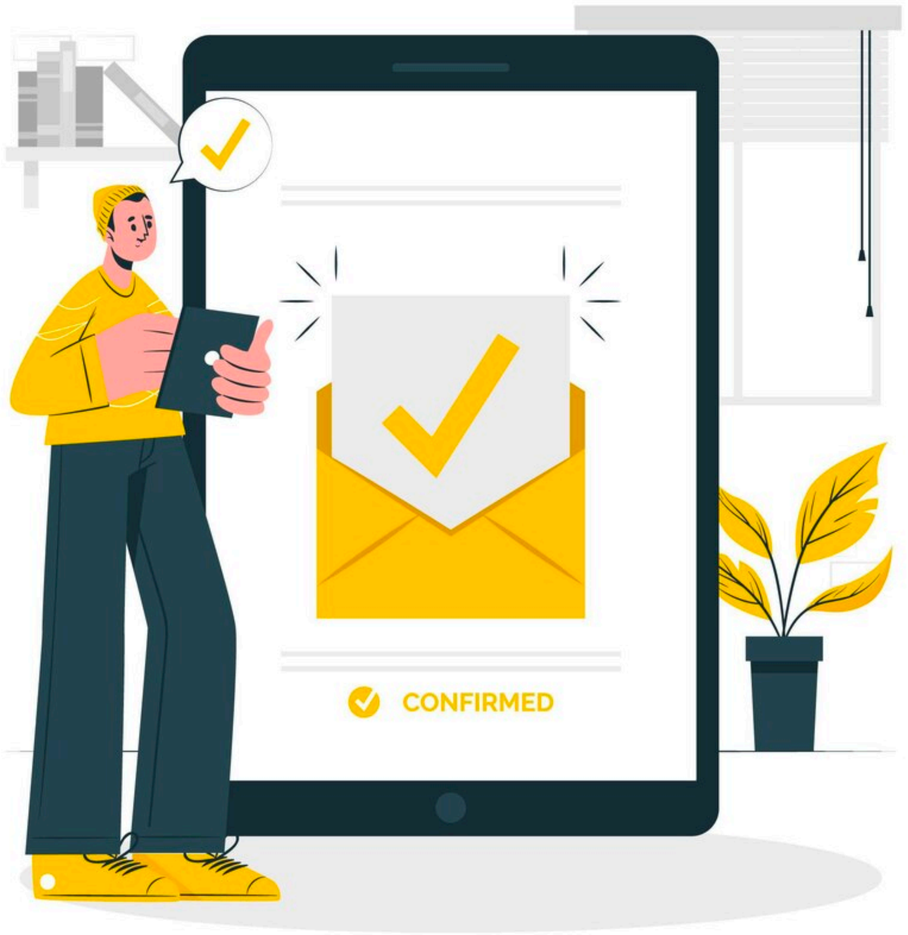
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You're developing a software system that manages all ***employees information inside the company***. This system should be re-used with different IT departments as a library that should be available to be integrated with.

You are required to add the functionality that allow consumer of your library to ***iterate through employees direct reports hierarchy, co-workers and sub-ordinates with the following names e.g getNext and hasNext.***



© Mahmoud Youssef

NAIVE SOLUTION

EmployeeHierarchyCollection

Attributes

- employees: Employee[]
- employeeId: String

Operations

+ EmployeeHierarchyCollection(String)
+ getNextEmployeeDirectReport(): Employee
+ hasNextEmployeeDirectReport(): boolean
+ getNextEmployeeCoWorker(): Employee
+ hasNextEmployeeCoWorker(): boolean
+ getNextEmployeeSubOrdinate(): Employee
+ hasNextEmployeeSubOrdinate(): boolean

Duplication

Limited Flexibility

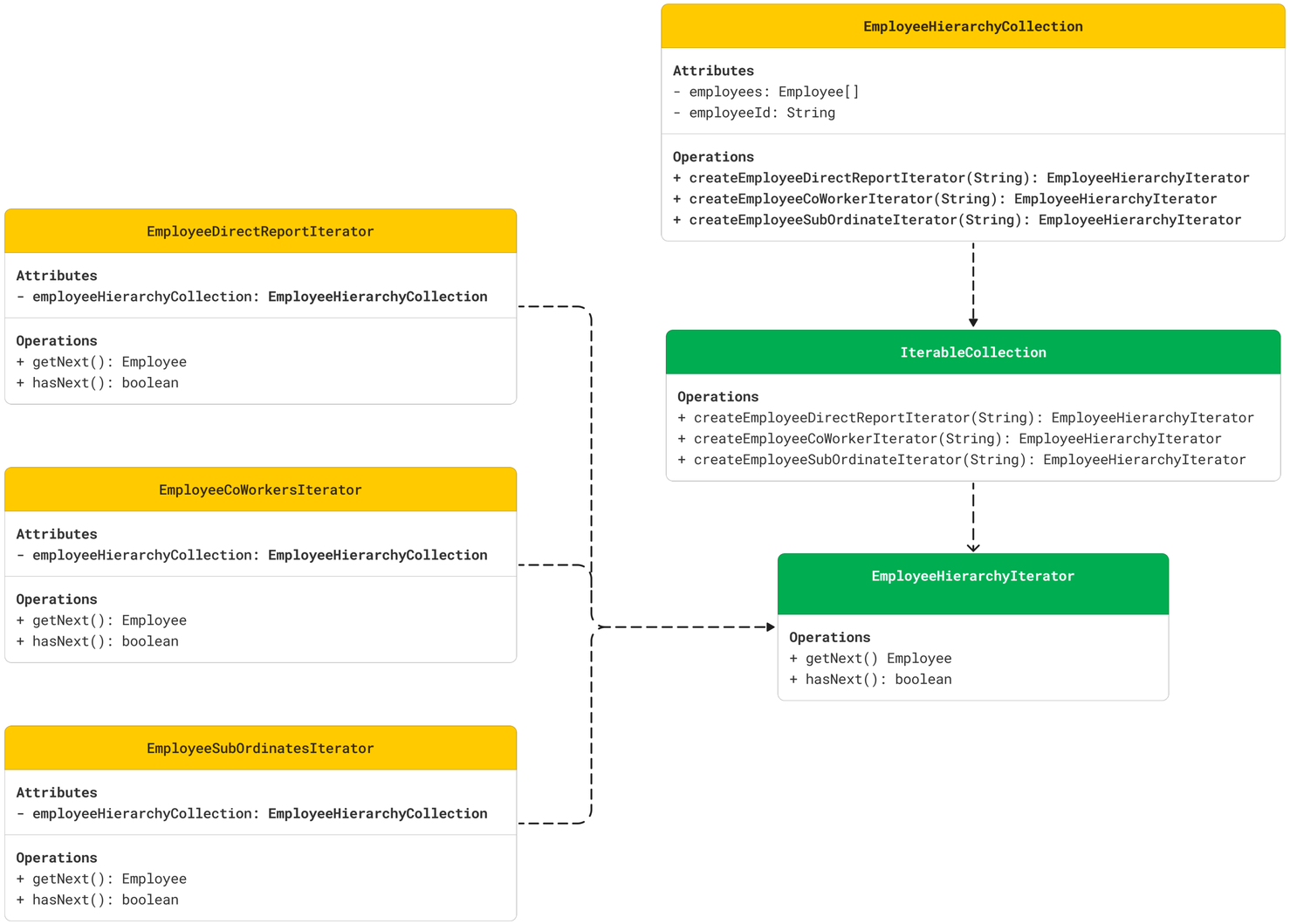
Reusability

Student

```java
public class Main {
    public static void main(String[] args) {
        EmployeeHierarchyCollection employeeHierarchyCollection =
                new EmployeeHierarchyCollection("test");
        while (employeeHierarchyCollection.hasNextEmployeeDirectReport()) {
            System.out.println(employeeHierarchyCollection
                    .getNextEmployeeDirectReport().getName());
        }
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Separation of Concerns

Maintainability

Reusability

© Mahmoud Youssef

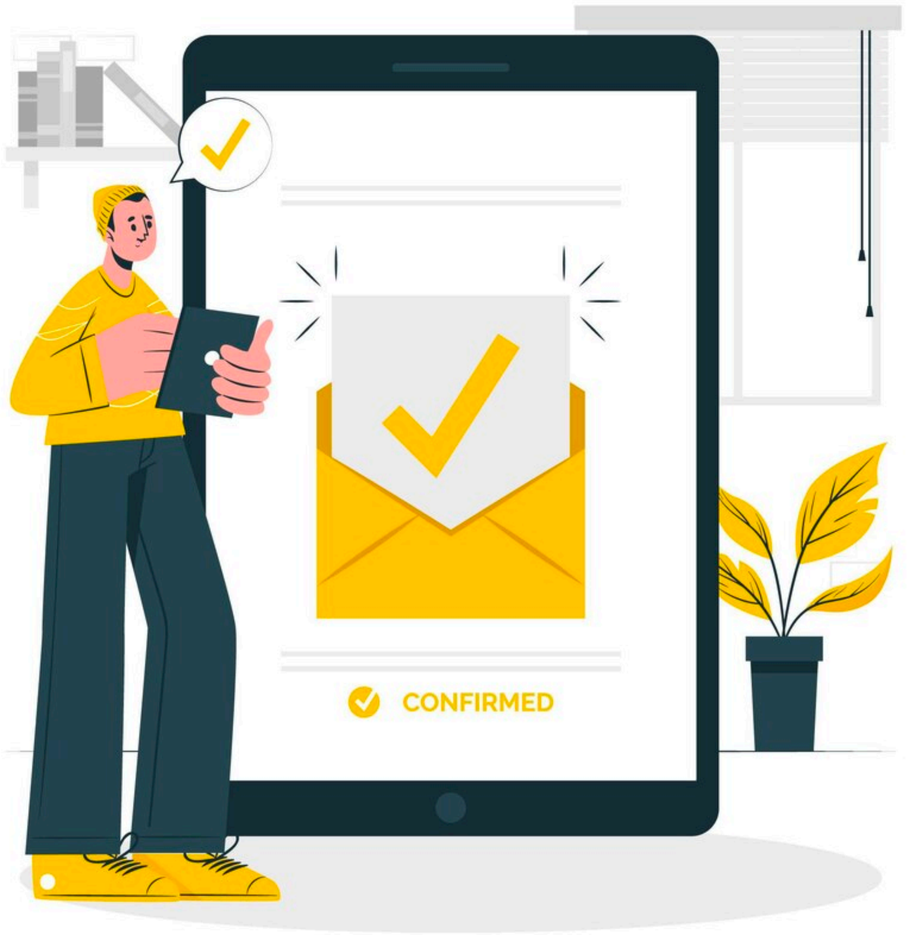
CHAIN OF RESONSIBILITY

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

Consider a web application where ***incoming HTTP requests*** need to go through several layers of middleware for ***authentication, authorization, and security checks*** before reaching the main request handler.

You are required to ***design and implement the request processor*** considering each request need to be ***authenticated, authorized and granted the security checks*** before processing the real request.



© Mahmoud Youssef

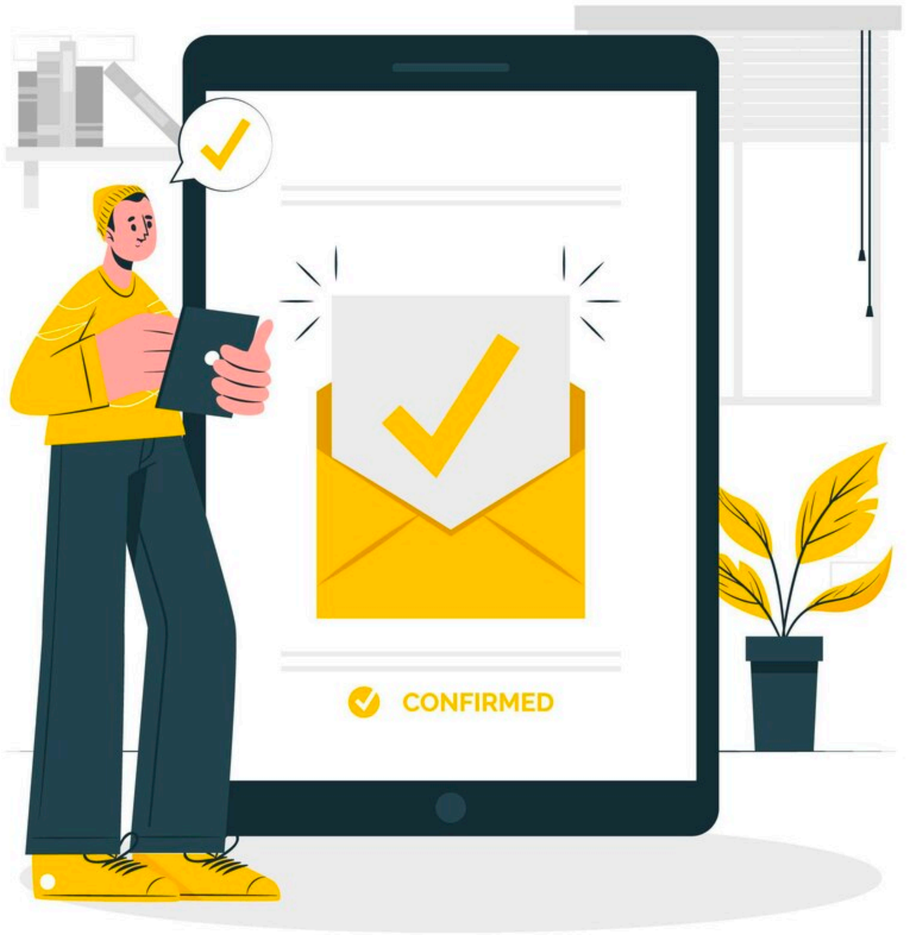
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

Consider a web application where ***incoming HTTP requests*** need to go through several layers of middleware for ***authentication, authorization, and security checks*** before reaching the main request handler.

You are required to ***design and implement the request processor*** considering each request need to be ***authenticated, authorized and granted the security checks*** before processing the real request.



© Mahmoud Youssef

NAIVE SOLUTION

RequestProcessor

Attributes

Operations

```
+ processRequest(Request): Response
- hasPassedSecurityChecks(Request): boolean
- isAuthenticated(Request): boolean
- isAuthorized(Request): boolean
```

Tightly Coupled

Reusability

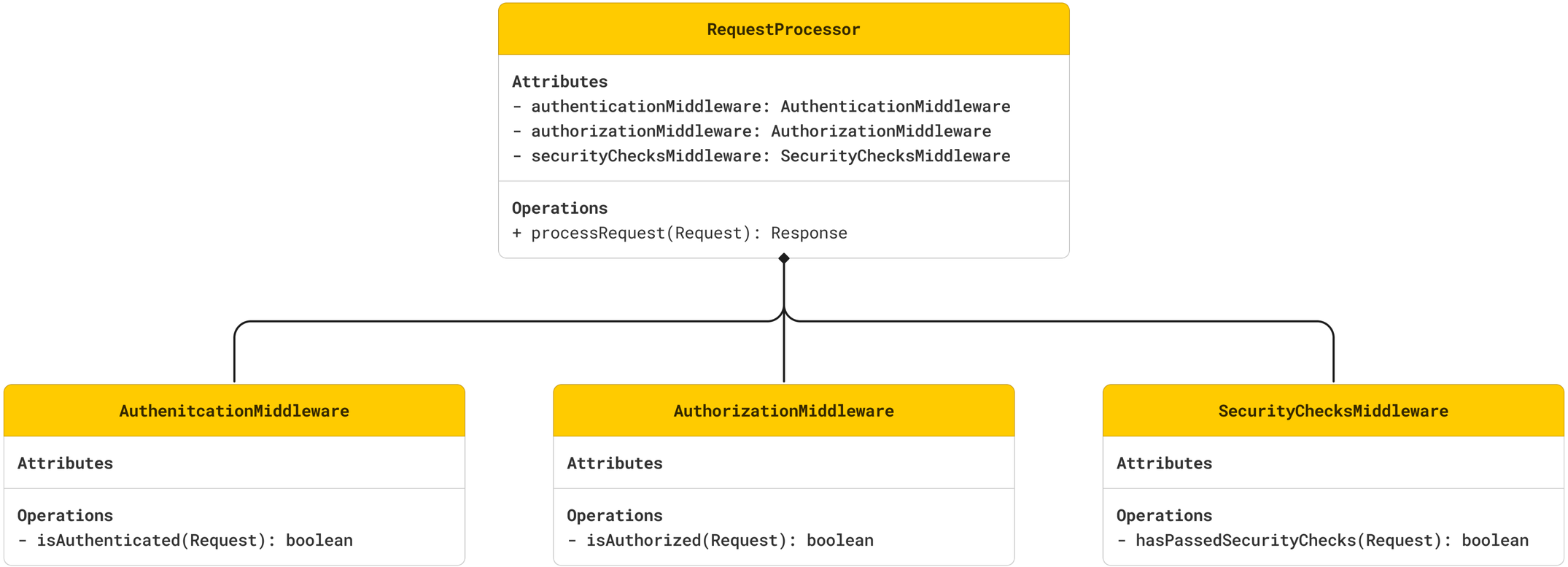
Extensibility

Chain of Responsibility

```java
public class Main {
    public static void main(String[] args) {
        RequestProcessor requestProcessor = new RequestProcessor();
        Request request = new Request("GET", true, true, true);
        var result = requestProcessor.processRequest(request);
        System.out.println("Request is Succeeded: " + result.isSucceeded() + "
because of: " + result.getReason());
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



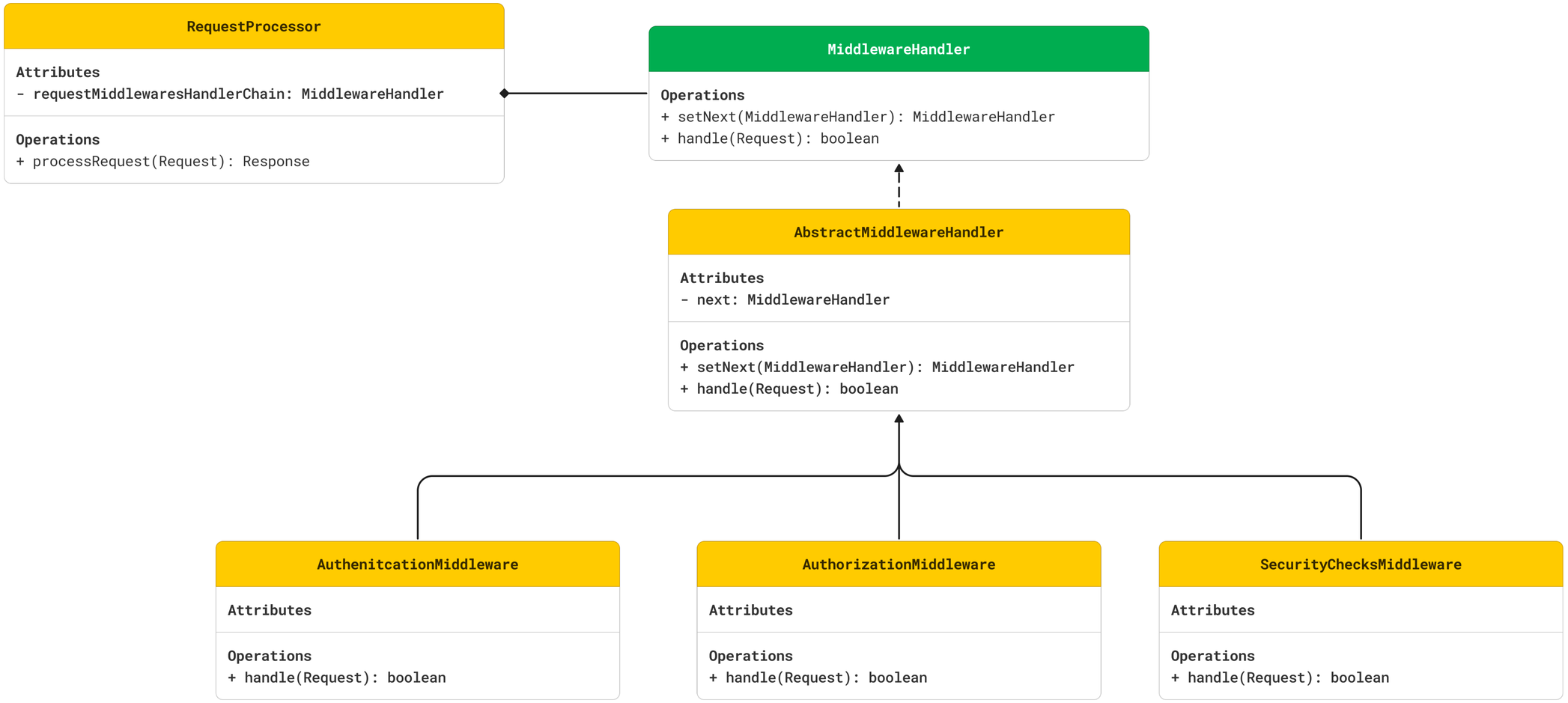
Separation of Concerns

Maintainability

Reusability

© Mahmoud Youssef

ENHANCED SOLUTION



© Mahmoud Youssef

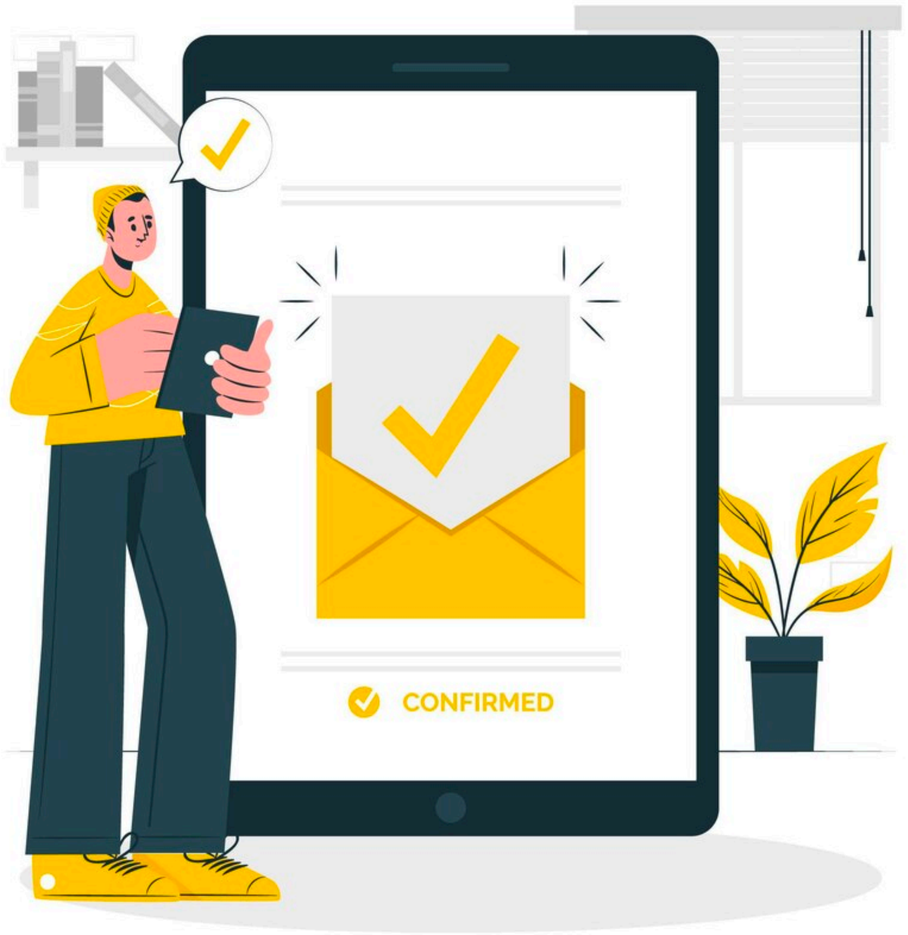
STATE

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing an E-commerce system which has the following requirements: ***orders go through different stages*** such as ***"New", "Processing", "Shipped", and "Delivered"***. The behavior of an order changes based on its current state.

For example, a ***"New" order can be processed and cancelled***, while a "Shipped" order cannot be canceled but can be marked as delivered and ***"Processing" order can be cancelled and shipped***.



© Mahmoud Youssef

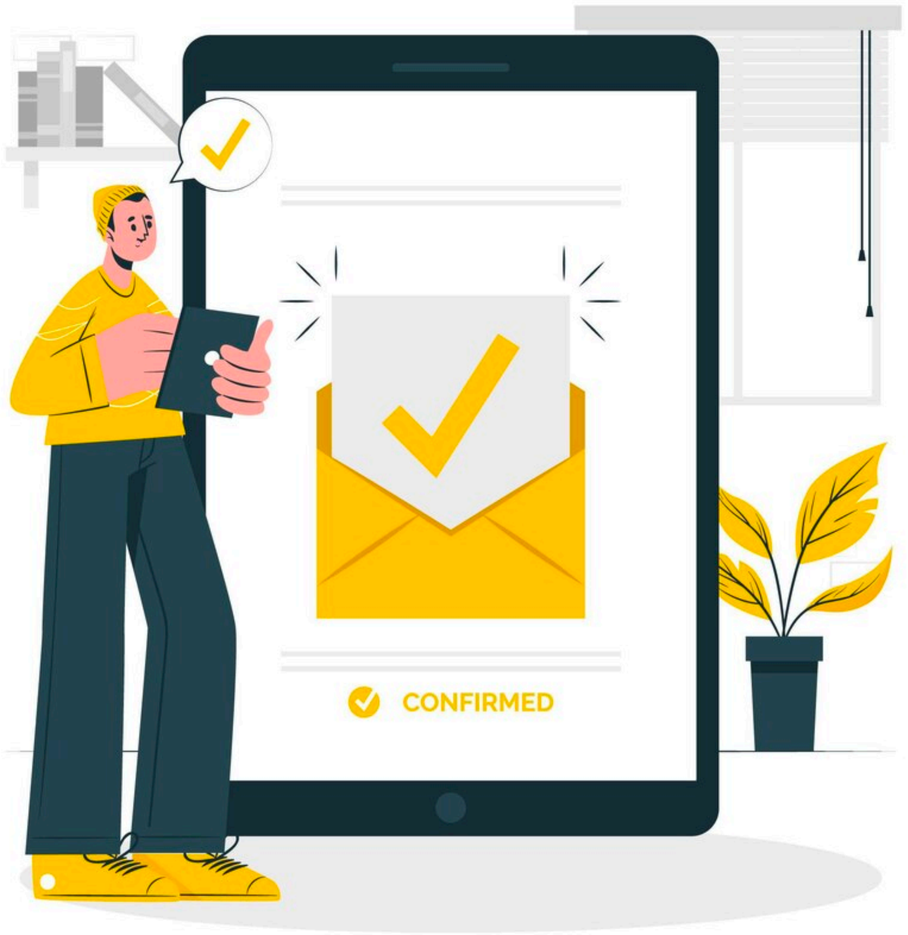
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing an E-commerce system which has the following requirements: ***orders go through different stages*** such as ***"New", "Processing", "Shipped", and "Delivered"***. The behavior of an order changes based on its current state.

For example, a ***"New" order can be processed and cancelled***, while a "Shipped" order cannot be canceled but can be marked as delivered and ***"Processing" order can be cancelled and shipped***.



© Mahmoud Youssef

NAIVE SOLUTION

OrderManagement

Attributes
- order: Order
- orderState: String

Operations
+ processOrder()
+ shipOrder()
+ deliverOrder()
+ cancelOrder()

Complexity

Tightly Coupled

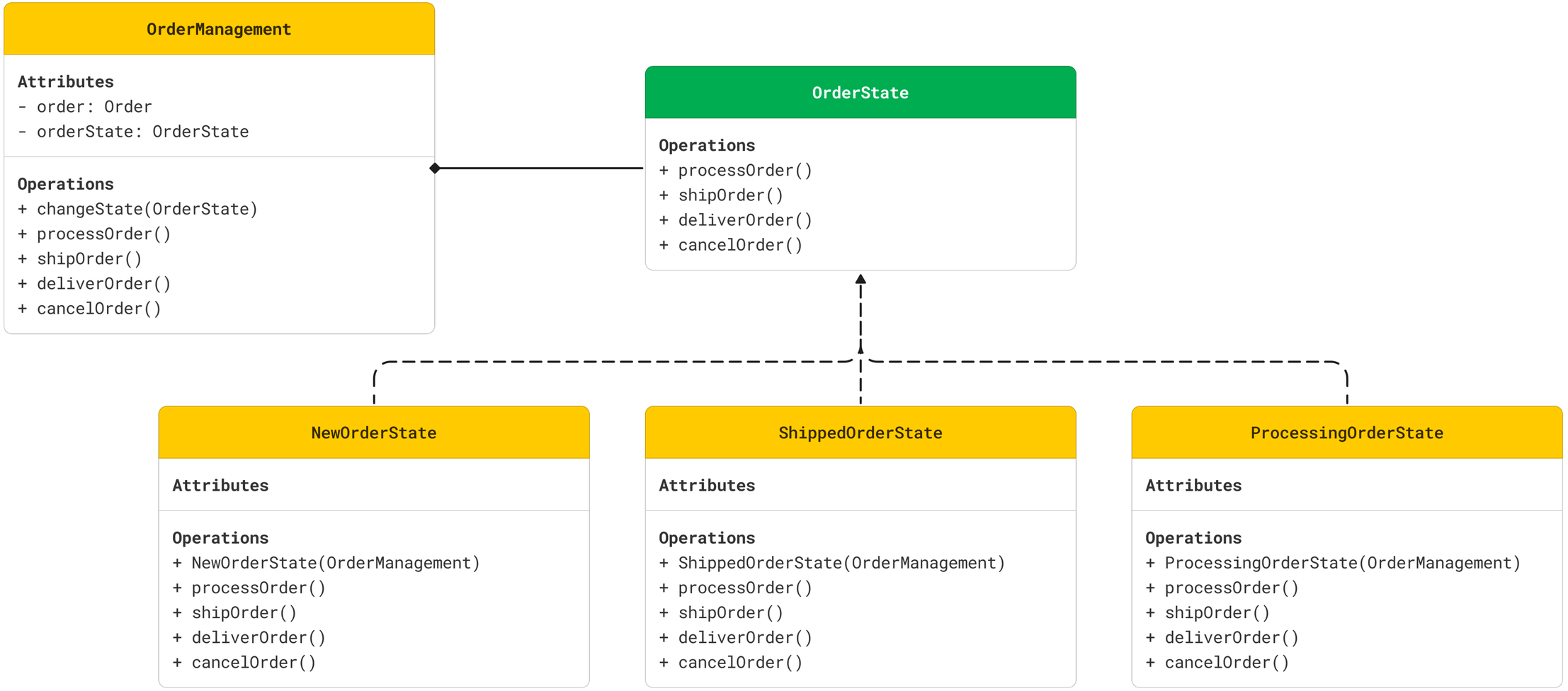
Extensibility

Chain of Responsibility

```java
public class Main {
    public static void main(String[] args) {
        Order order = new Order("Jacket", 100.0);
        OrderManagement orderManagement = new OrderManagement(order);

        orderManagement.processOrder();
        orderManagement.cancelOrder();
        orderManagement.shipOrder();
        orderManagement.deliverOrder();
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Maintainability **Readability** **Extensibility**

© Mahmoud Youssef

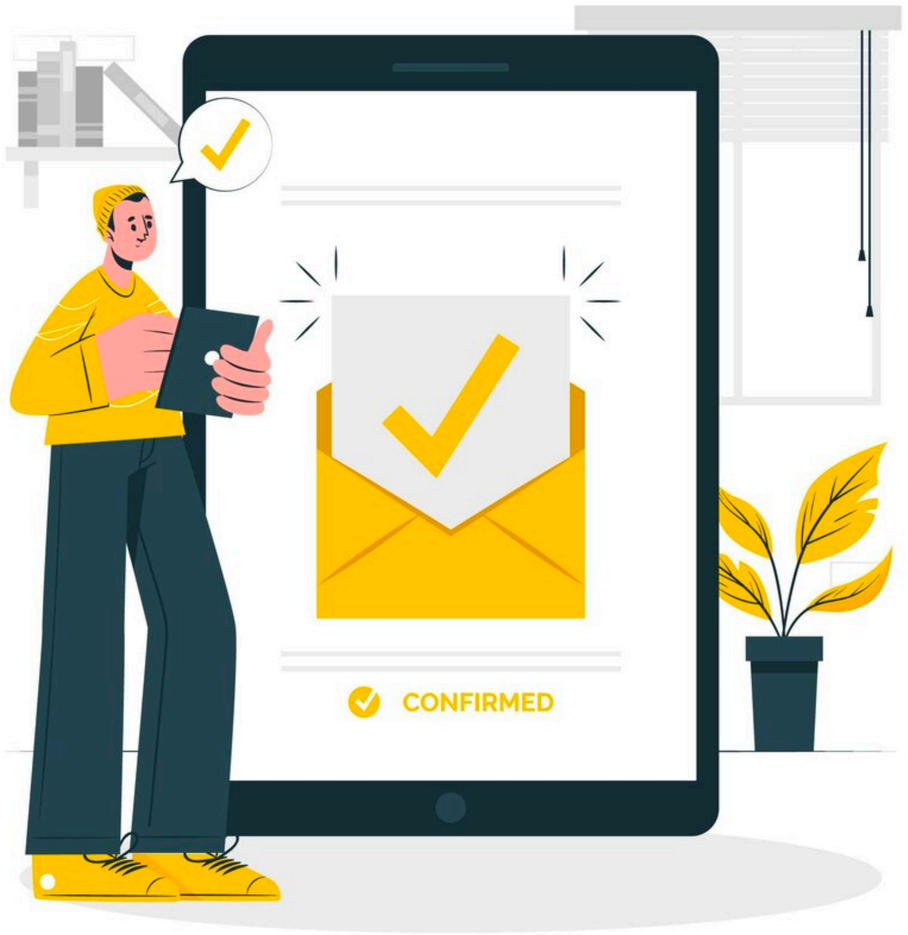
MEDIATOR

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing a ***Chat Application*** where ***multiple users can communicate with each others*** directly or in a group.

Each user can ***send messages and receive messages*** from other users and chat groups.



© Mahmoud Youssef

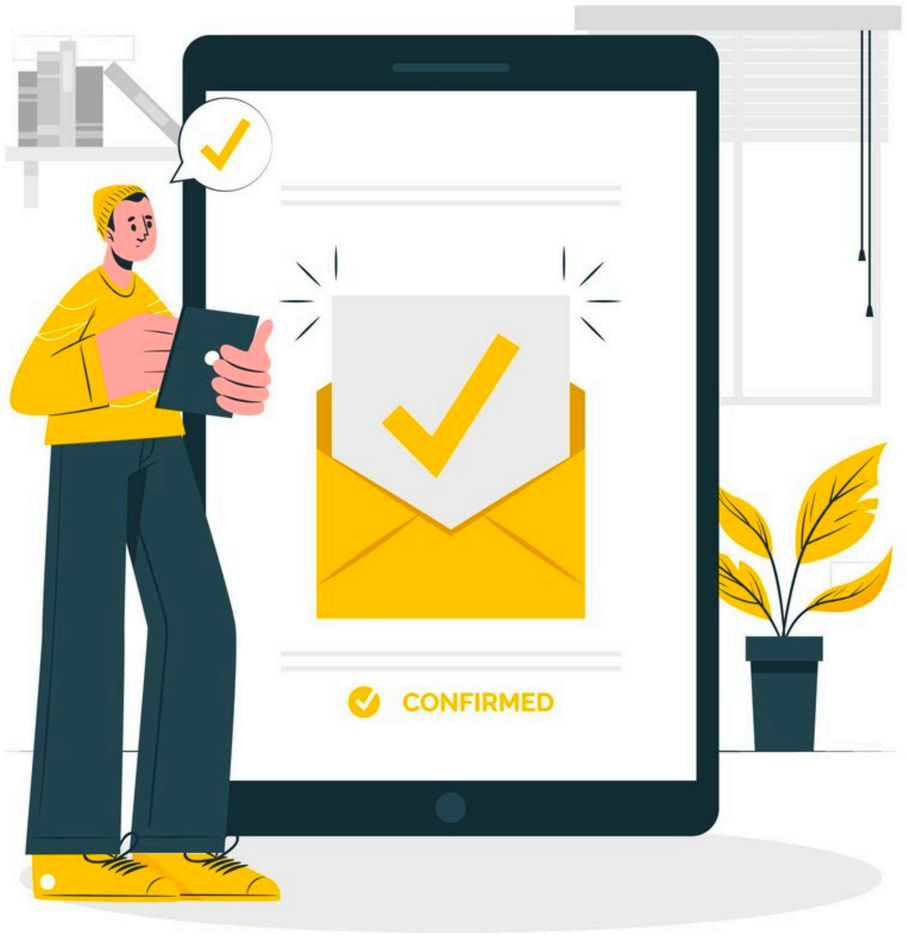
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

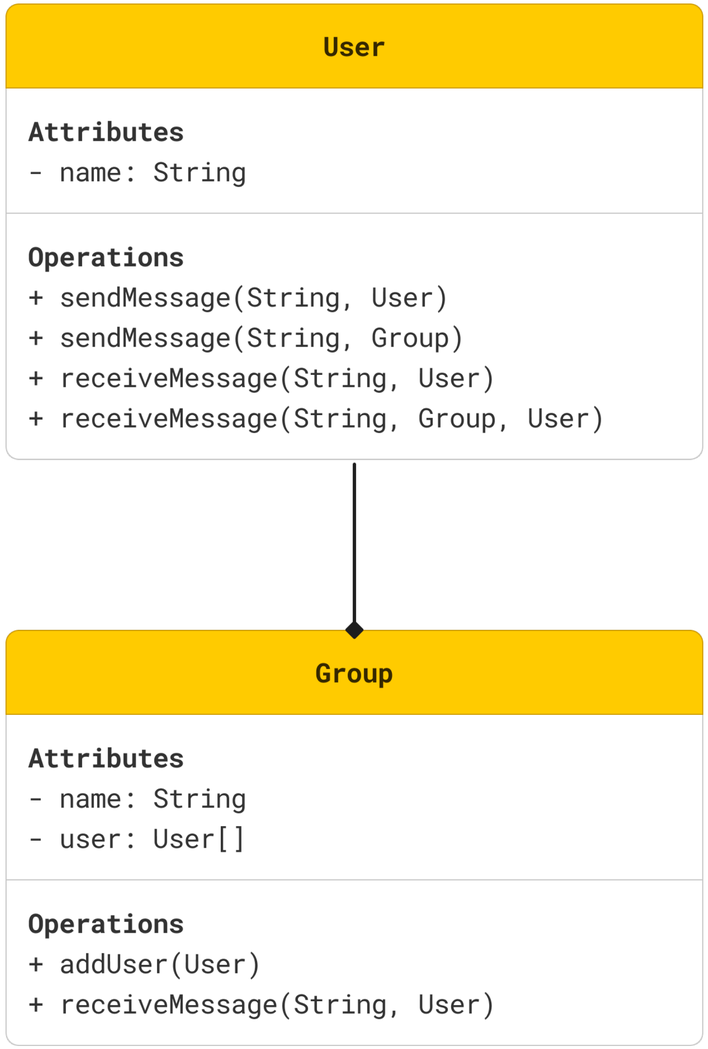
You are developing a ***Chat Application*** where ***multiple users can communicate with each others*** directly or in a group.

Each user can ***send messages and receive messages*** from other users and chat groups.



© Mahmoud Youssef

NAIVE SOLUTION



Tightly Coupled

Complexity

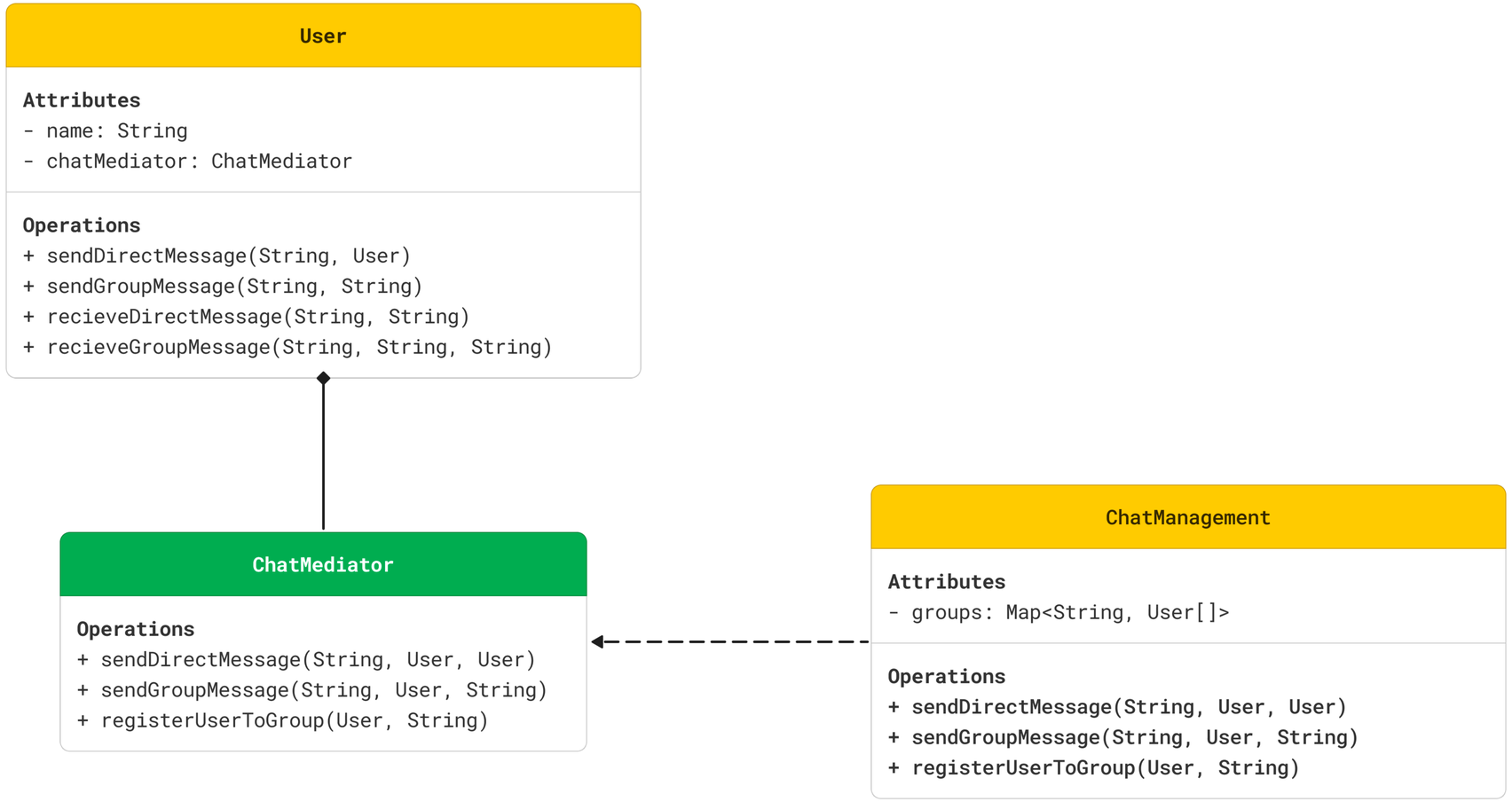
Reusability

Mediator

```java
public class Main {
    public static void main(String[] args) {
        User mahmoud = new User("Mahmoud");
        User ahmed = new User("Ahmed");
        User youssef = new User("Youssef");

        Group group = new Group("Fun-Gaming");
        group.addUser(mahmoud);
        group.addUser(ahmed);
        group.addUser(youssef);

        mahmoud.sendMessage("Hello everyone!", group);
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Unified Communication

Extensibility

Encapsulation

© Mahmoud Youssef

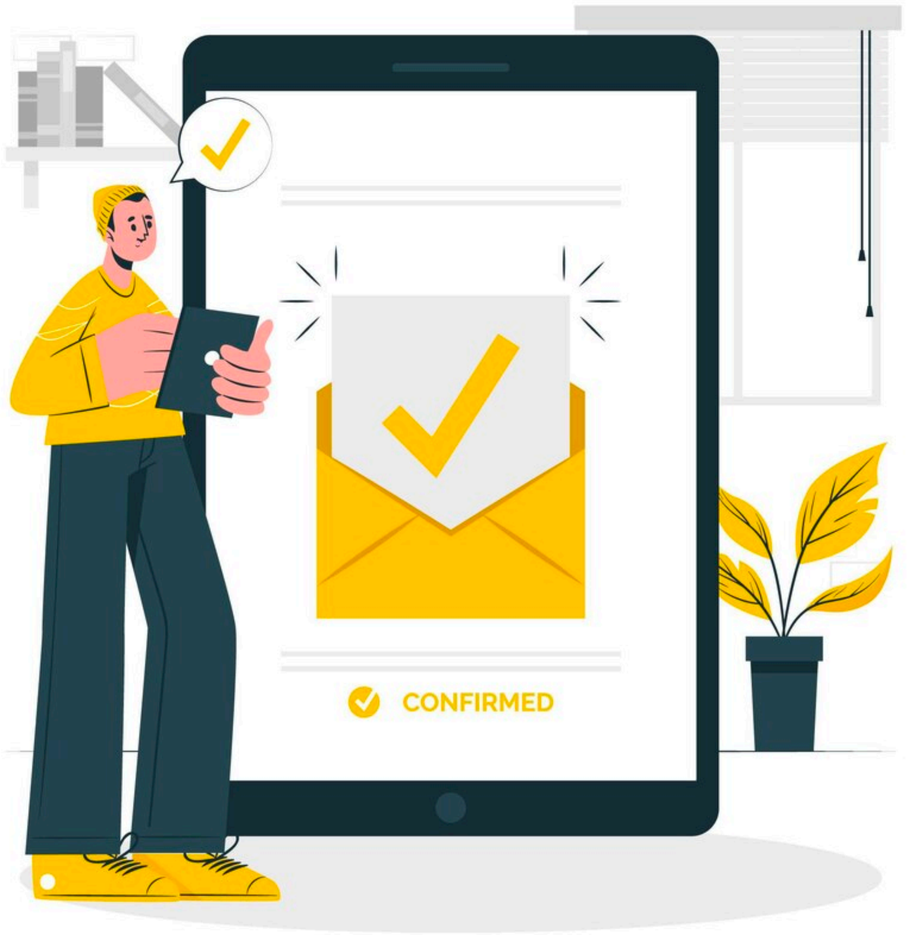
OBSERVER

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

We want to create a Marketplace Notification system where ***users can subscribe to notifications for new product arrivals or new offers*** in an online marketplace. Whenever a new product or new offer is added to the marketplace, subscribed users should receive notifications.

We should allow both ***adding new subscribers and cancelling subscription as well***.



© Mahmoud Youssef

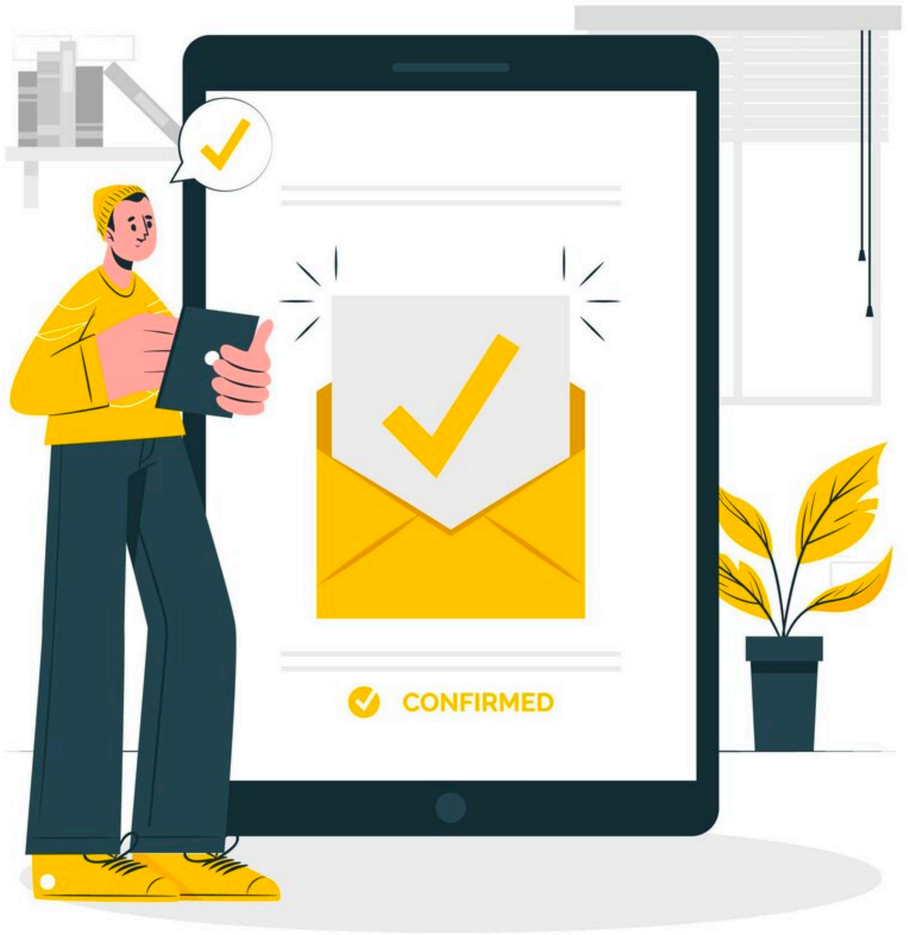
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

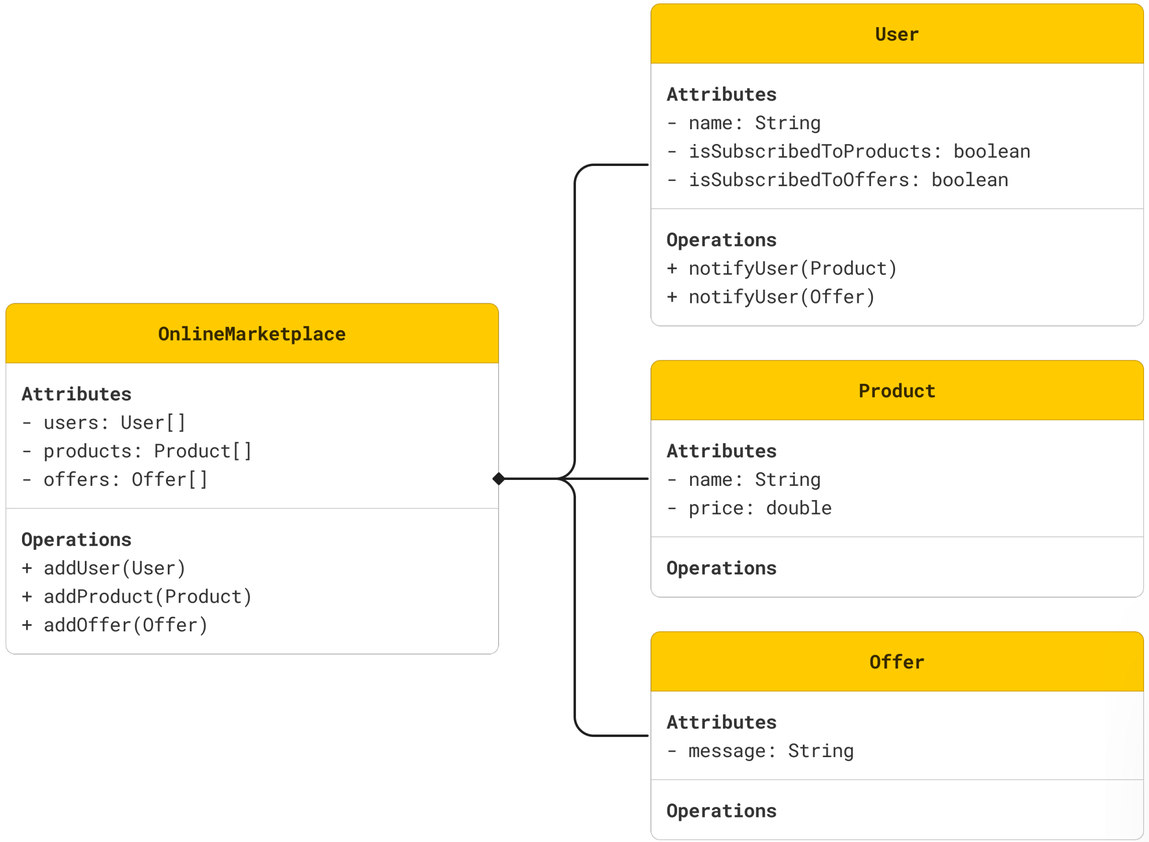
We want to create a Marketplace Notification system where ***users can subscribe to notifications for new product arrivals or new offers*** in an online marketplace. Whenever a new product or new offer is added to the marketplace, subscribed users should receive notifications.

We should allow both ***adding new subscribers and cancelling subscription as well***.



© Mahmoud Youssef

NAIVE SOLUTION



Complexity

Tightly Coupling

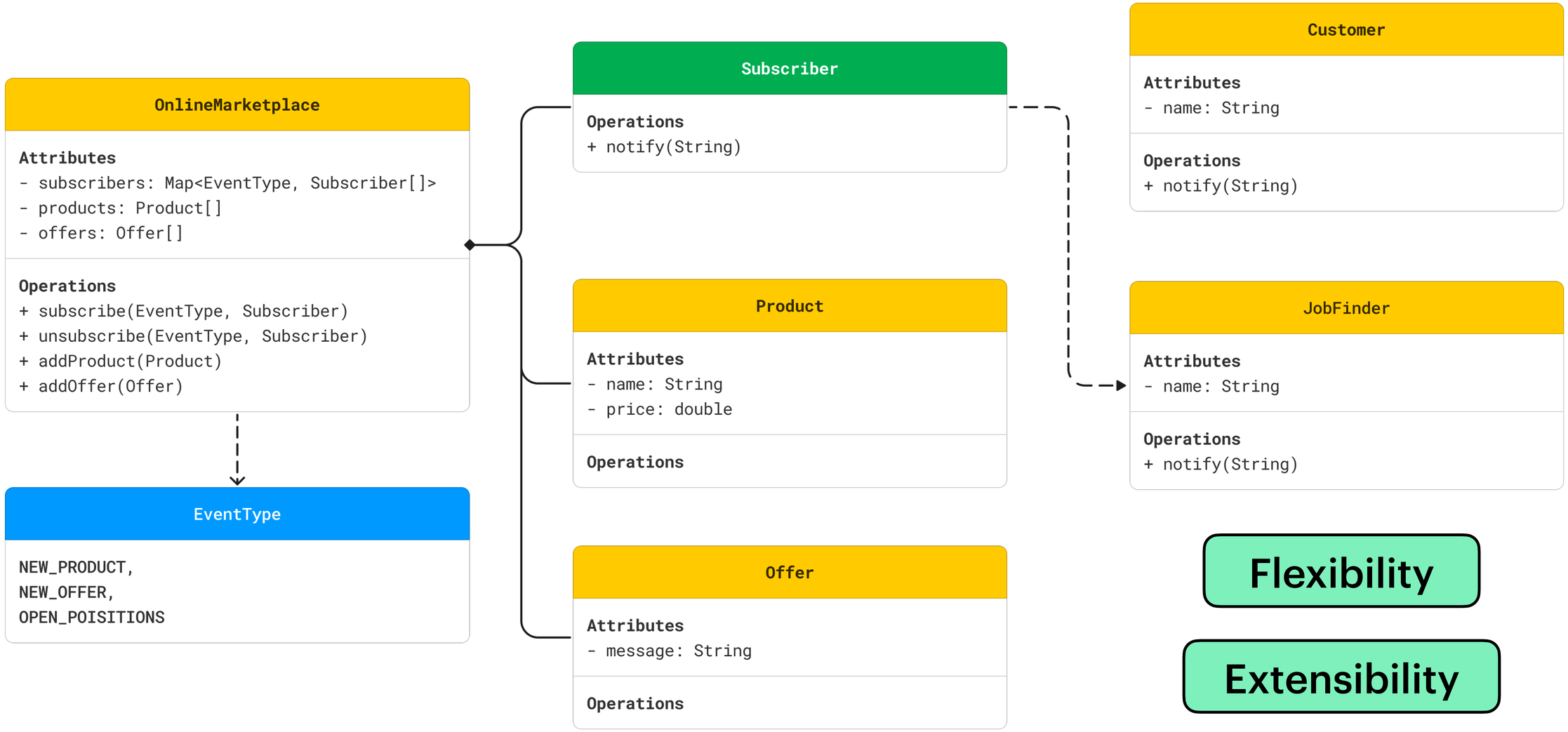
Extensibility

Observer

```java
public class Main {
    public static void main(String[] args) {
        OnlineMarketplace onlineMarketplace = new OnlineMarketplace();
        onlineMarketplace.addUser(new User("Ahmed", true, true));
        onlineMarketplace.addUser(new User("Mahmoud", false, true));
        onlineMarketplace.addUser(new User("Youssef", true, false));
        onlineMarketplace.addUser(new User("Mostafa", false, false));

        onlineMarketplace.addNewProduct(new Product("Mobile", 1000.0));
        onlineMarketplace.addNewOffer(new Offer("New offer with 20% discount
for every item."));
    }
}
```

© Mahmoud Youssef

ENHANCED SOLUTION



Flexibility

Extensibility

© Mahmoud Youssef

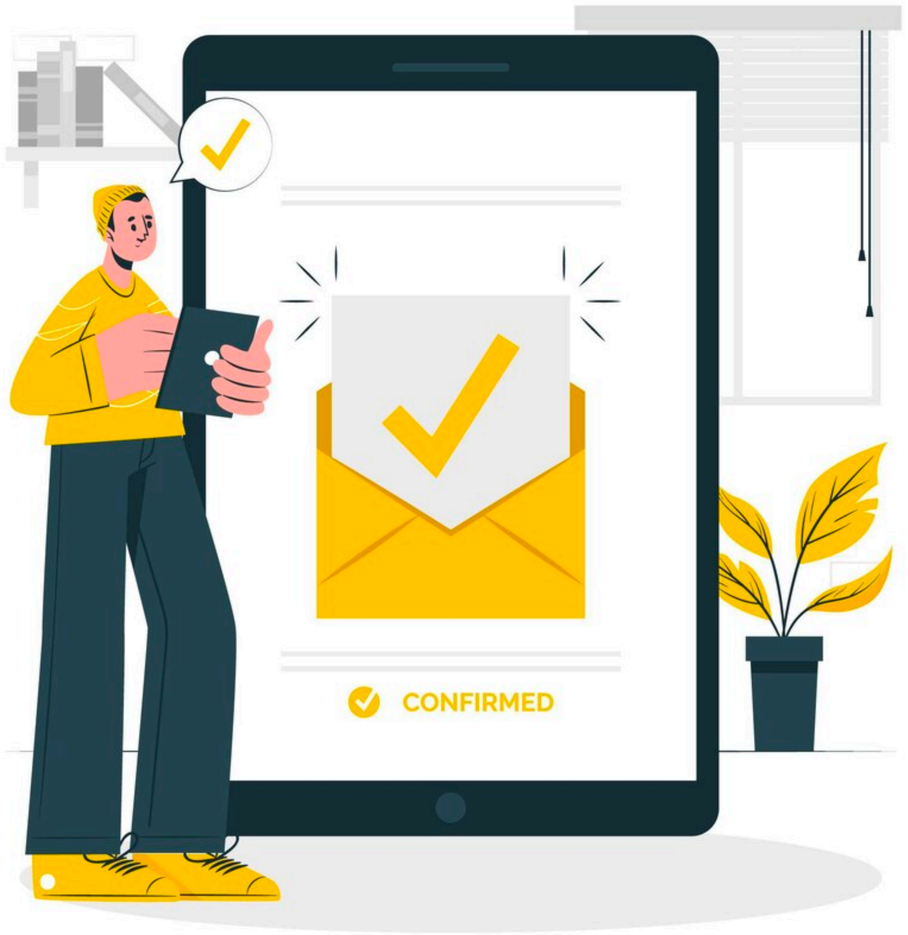
COMMAND

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing a ***smart home mobile application***, which provides the following functionalities: ***turn lights on and off, lock and unlock front door, turn TV on and off***.

These functionalities need to be provided through <u>the mobile application as well as voice assistant and some shortcuts</u>.



© Mahmoud Youssef

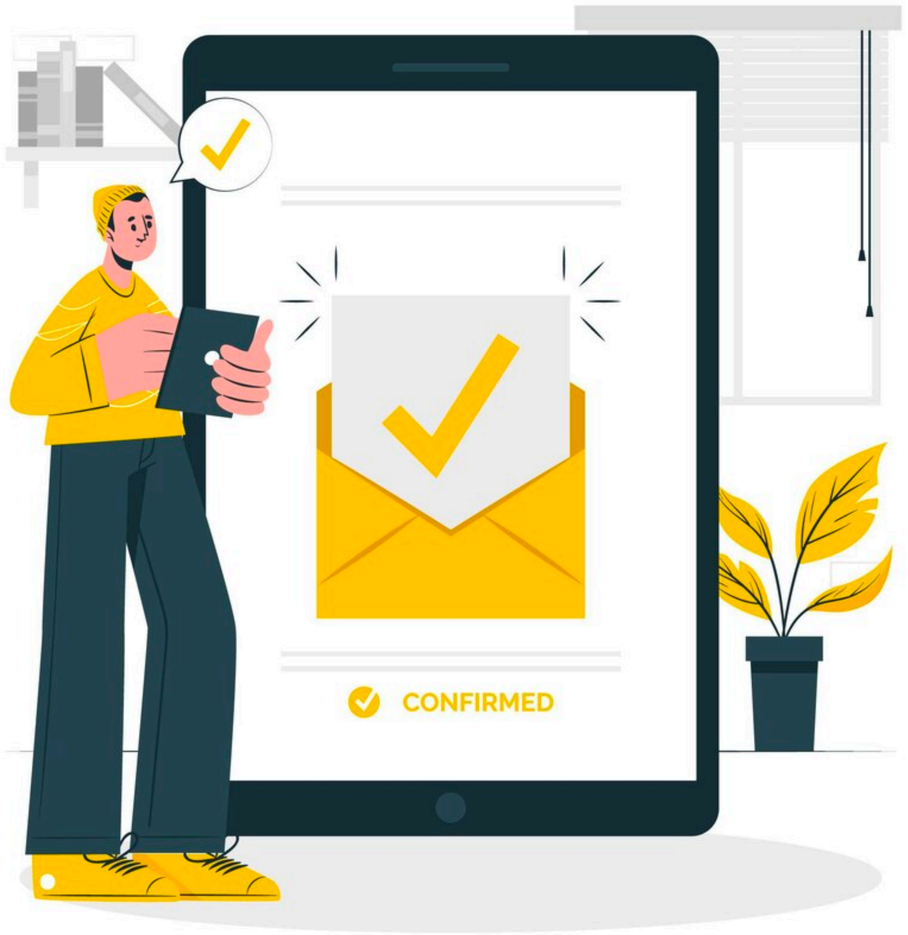
SOLUTION

MAHMOUD YOUSSEF
SOFTWARE ENGINEER

WHAT IS THE PROBLEM?

You are developing a ***smart home mobile application***, which provides the following functionalities: ***turn lights on and off, lock and unlock front door, turn TV on and off***.

These functionalities need to be provided through <u>the mobile application as well as voice assistant and some shortcuts</u>.



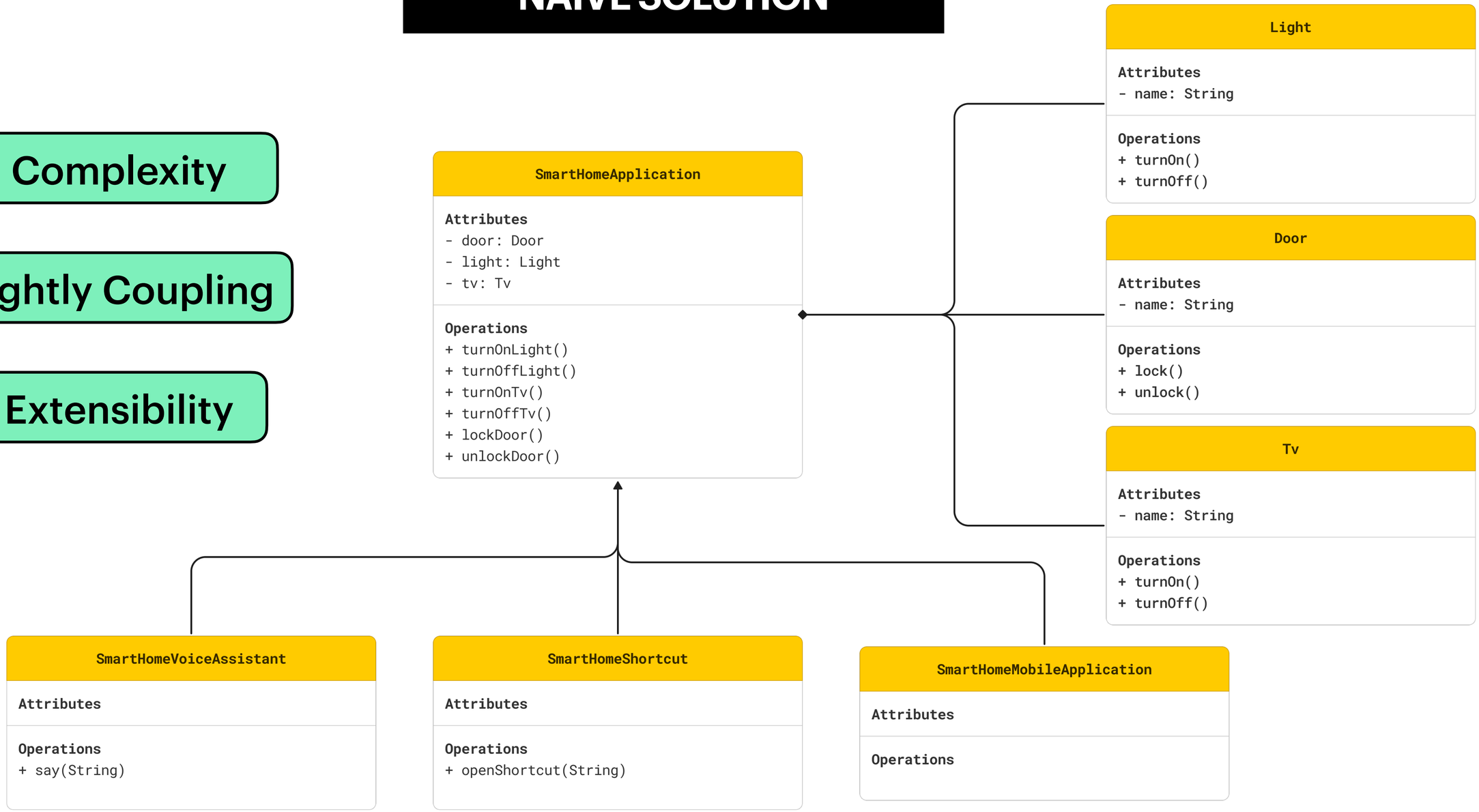
© Mahmoud Youssef

NAIVE SOLUTION

Complexity

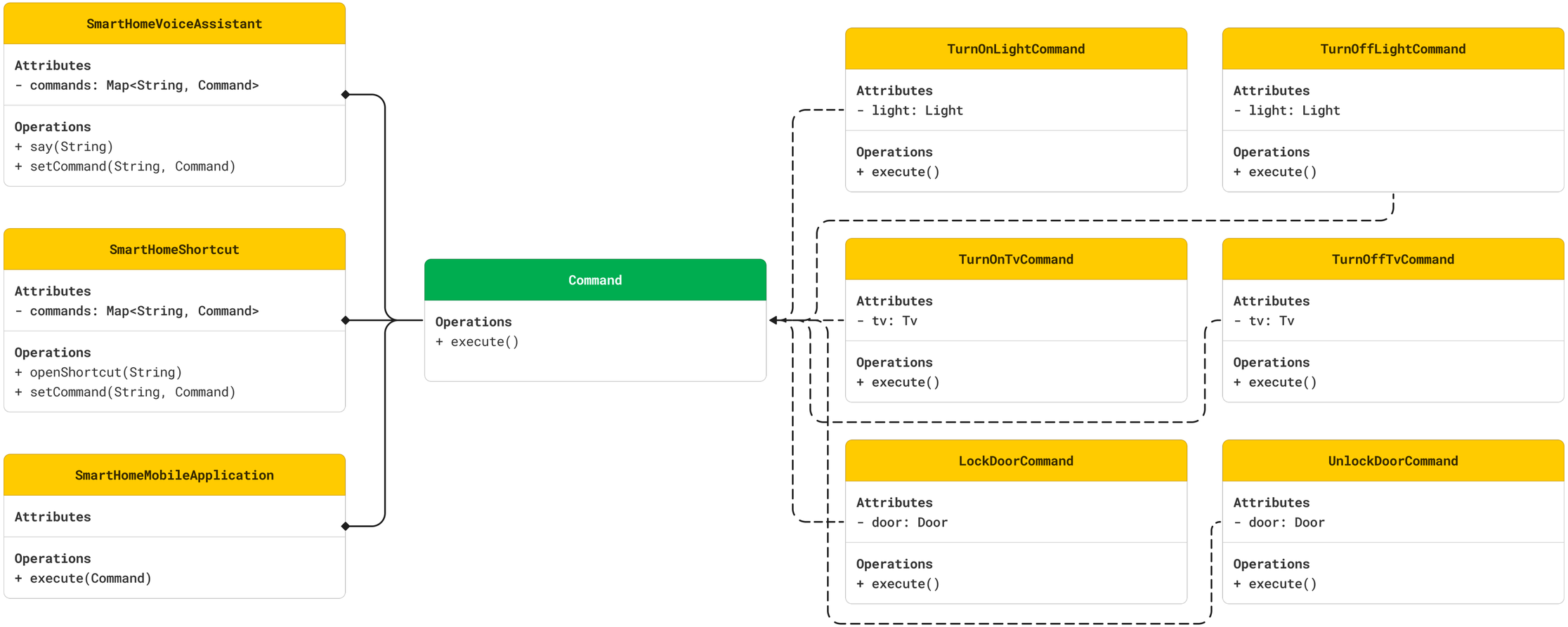
Tightly Coupling

Extensibility



© Mahmoud Youssef

ENHANCED SOLUTION



Maintainability **Flexibility** **Extensibility**

© Mahmoud Youssef