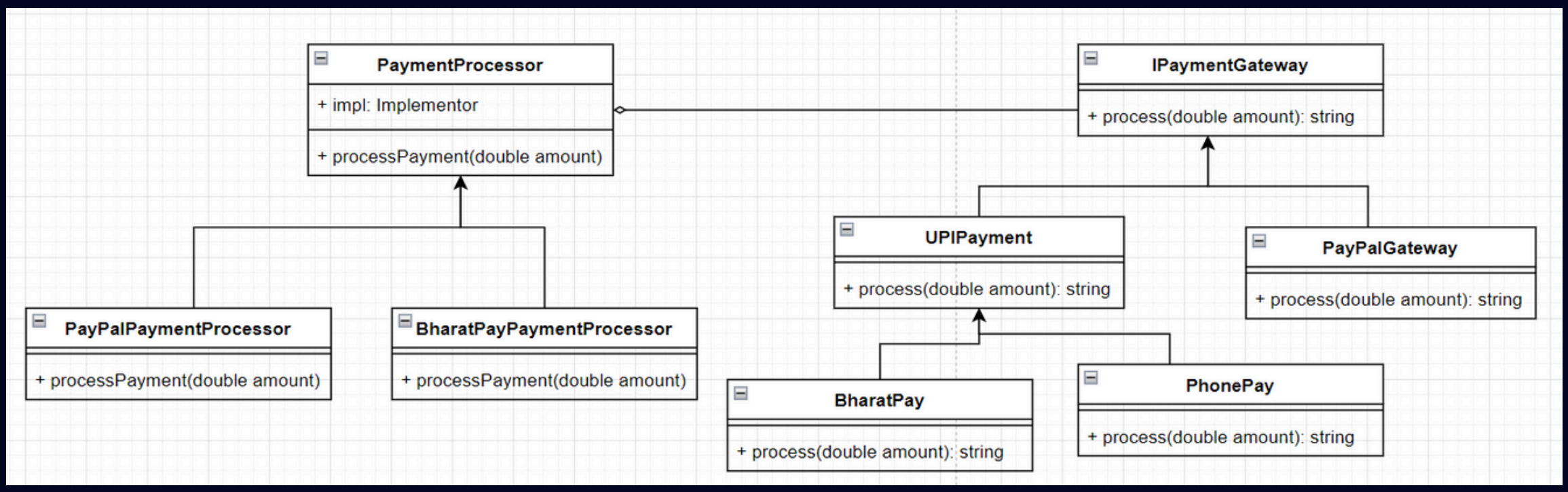
Bridge Design Pattern

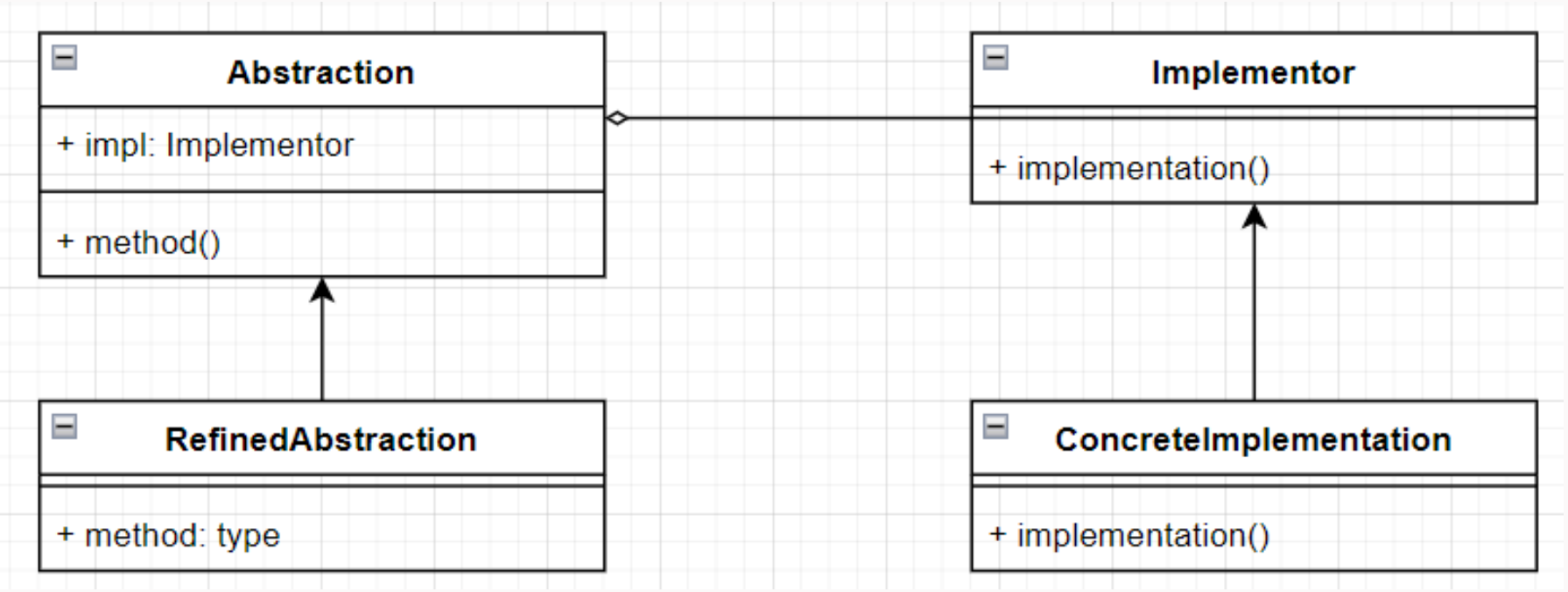


Smitesh Tamboli

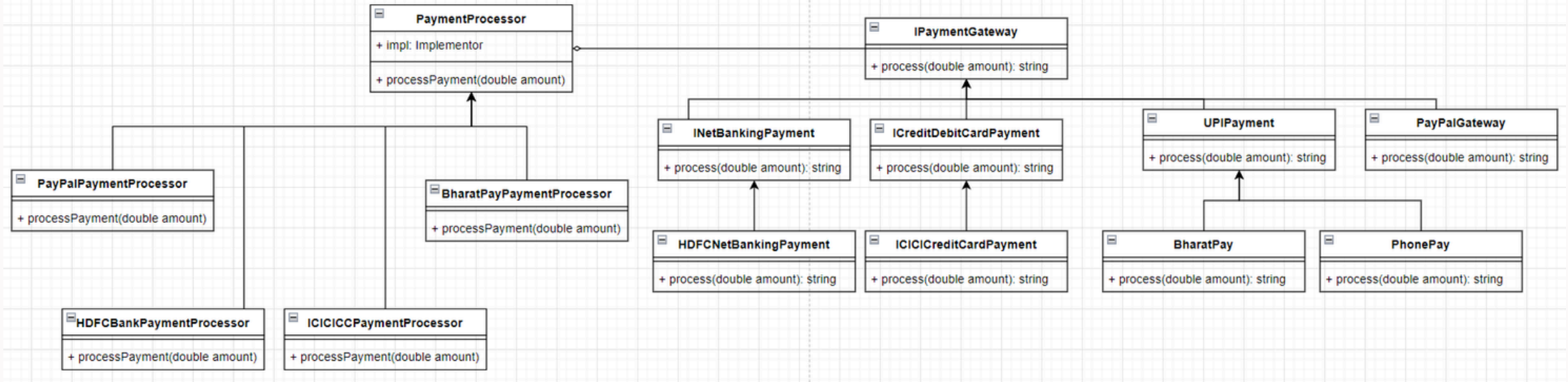
NEXT ➡

Bridge Design Pattern

- Bridge Pattern is a structural design pattern
- The intent of the pattern is to separates an abstraction from its implementation so that the two can vary independently.
- The pattern in particularly useful when we need to extend both sides of the abstraction-implementation relationship independently.
- The pattern helps to avoid the creation of a large number of subclasses and improves flexibility and extensibility



Example



Example

Let's consider, we are building an **eCommerce Portal** and need to **process the payment in different modes** like **net banking, credit/debit card, PayPal, and UPI methods**. Without using a Bridge Pattern, implementation of the requirement may end up making a large class hierarchy like for each bank we need to build a class for net banking, credit, and debit cards, payment through PayPal, Stripe, and all the UPI methods (PhonePe, BharatPe, GPay, etc.). This class hierarchy can be difficult to manage.

Bridge Pattern helps this long class hierarchy into separate abstraction and implementation. We can separate **PaymentGateway** to **PaymentProcessor**.

Implementation Interface (IPaymentGateway)

- The IPaymentGateway interface defines the process() method that processes the payment. Different implementations of this interface provide concrete details of how the payment is processed.

Concrete Implementations

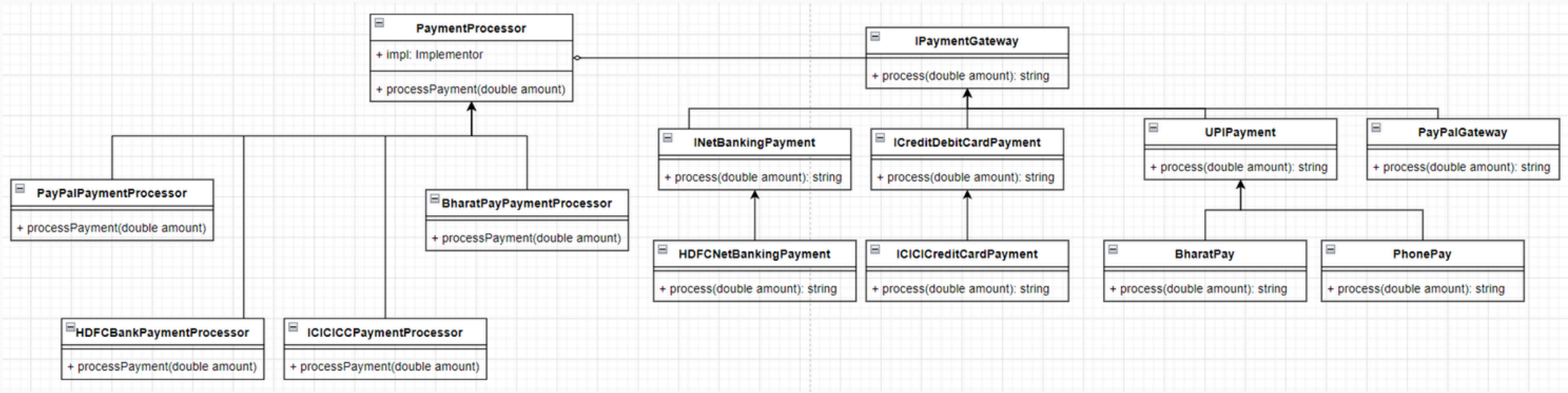
- HDFCBakPayment, ICICIBankCreditCardPayment, PayPalGateway, BharatPayUPI

Abstraction (PaymentProcessor)

- The PaymentProcessor class serves as the abstraction and maintains the reference of the implementor (IPaymentGateway)

Concrete Implementation

- HDFCBakPaymentProcessor
- ICICIBankCreditCardPaymentProcessor
- PayPalPaymentProcessor
- BharatPayPaymentProcessor



Code : Click Here

```cpp
struct IPaymentGateway
{
    virtual string process(double amount) = 0;
    virtual ~IPaymentGateway() {}
};

struct INetBankingPayment : public IPaymentGateway {
    virtual ~INetBankingPayment() {}
};

class HDFCBakPayment : public INetBankingPayment {
public:
    string process(double amount) override {
        string transactionId = "HDFC00001";
        cout<< "HDFC Net Banking Payment : " << std::to_string(amount) << endl;
        return transactionId;
    }

    ~HDFCBakPayment(){}
};


struct ICreditDebitCardPayment : public IPaymentGateway {
    virtual ~ICreditDebitCardPayment() {}
};
```

```cpp
class ICICIBankCreditCardPayment : public ICreditDebitCardPayment {
private:
    // string credit_card_number
public:
    string process(double amount) override {
        string transactionId = "ICICICC00001";
        cout << "ICICI Bank Credit Card Payment :" << std::to_string(amount) << endl;
        return transactionId;
    }

    ~ICICIBankCreditCardPayment() {}
};

class PayPalGateway : public IPaymentGateway {
public:
    string process(double amount) override {
        string transactionId = "PAYPAL00001";
        cout << "PayPal Payment : " << std::to_string(amount) << endl;
        return transactionId;
    }

    ~PayPalGateway() {}
};

struct UPIPayment : public IPaymentGateway {
    virtual ~UPIPayment() {}
};

class BharatPayUPI : public UPIPayment {
public:
    string process(double amount) override {
        string transactionId = "BHARAT00001";
        cout << "Bharat UPI Payment : " << std::to_string(amount) << endl;
        return transactionId;
    }

    ~BharatPayUPI() {}
};

class PhonePePayUPI : public UPIPayment {
public:
    string process(double amount) override {
        string transactionId = "PHONEPAY00001";
        cout << "PhonePe UPI Payment : " << std::to_string(amount)<<endl;
        return transactionId;
    }

    ~PhonePePayUPI() {}
};
```

```cpp
class PaymentProcessor {
protected:
    shared_ptr<IPaymentGateway> _paymentGateway;

public:
    PaymentProcessor(shared_ptr<IPaymentGateway> paymentGateway)
        :_paymentGateway(paymentGateway)
    {}

    virtual void processPayment(double amount) {
        string transactionID = _paymentGateway->process(amount);
        cout << "Transaction ID: " << transactionID << endl;
    }

    virtual ~PaymentProcessor(){}
};

class HDFCBakPaymentProcessor : public PaymentProcessor {
public:
    HDFCBakPaymentProcessor(shared_ptr<IPaymentGateway> paymentGateway)
        :PaymentProcessor(paymentGateway)
    {}

    void processPayment(double amount) override {
        PaymentProcessor::processPayment(amount);
    }
};

class ICICIBankCreditCardPaymentProcessor : public PaymentProcessor {
public:
    ICICIBankCreditCardPaymentProcessor(shared_ptr<IPaymentGateway> paymentGateway)
        :PaymentProcessor(paymentGateway)
    {}

    void processPayment(double amount) override {
        PaymentProcessor::processPayment(amount);
    }
};

class PayPalPaymentProcessor : public PaymentProcessor {
public:
    PayPalPaymentProcessor(shared_ptr<IPaymentGateway> paymentGateway)
        :PaymentProcessor(paymentGateway)
    {}

    void processPayment(double amount) override {
        PaymentProcessor::processPayment(amount);
    }
};

class BharatPayPaymentProcessor : public PaymentProcessor {
public:
    BharatPayPaymentProcessor(shared_ptr<IPaymentGateway> paymentGateway)
        :PaymentProcessor(paymentGateway)
    {}

    void processPayment(double amount) override {
        PaymentProcessor::processPayment(amount);
    }
};
```

```cpp
int main()
{
    shared_ptr<IPaymentGateway> hdfcNetBankingPayGway = make_shared<HDFCBakPayment>();
    shared_ptr<IPaymentGateway> icicicCCPayGway = make_shared<ICICIBankCreditCardPayment>();
    shared_ptr<IPaymentGateway> paypalPayGway = make_shared<PayPalGateway>();
    shared_ptr<IPaymentGateway> bharatUPIPayGway = make_shared<BharatPayUPI>();

    shared_ptr<PaymentProcessor> paymentProcessor = make_shared<HDFCBakPaymentProcessor>
(hdfcNetBankingPayGway);
    paymentProcessor->processPayment(200);

    paymentProcessor = make_shared<ICICIBankCreditCardPaymentProcessor>(icicicCCPayGway);
    paymentProcessor->processPayment(150);

    paymentProcessor = make_shared<PayPalPaymentProcessor>(paypalPayGway);
    paymentProcessor->processPayment(499);

    paymentProcessor = make_shared<BharatPayPaymentProcessor>(bharatUPIPayGway);
    paymentProcessor->processPayment(1100);

}

OUTPUT
=========================
HDFC Net Banking Payment : 200.000000
Transaction ID: HDFC00001
ICICI Bank Credit Card Payment :150.000000
Transaction ID: ICICICC00001
PayPal Payment : 499.000000
Transaction ID: PAYPAL00001
Bharat UPI Payment : 1100.000000
Transaction ID: BHARAT00001
```

Thank You