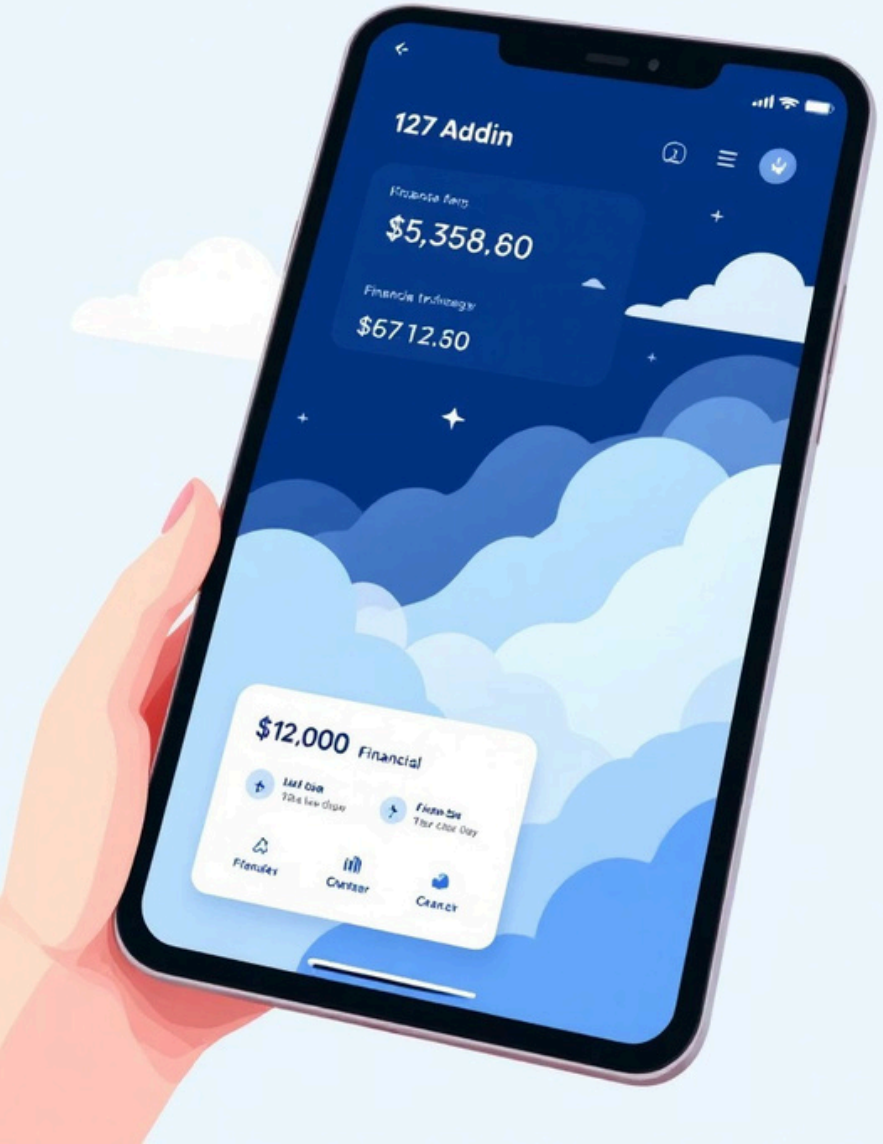


Loan Processing System

Building a robust loan management system using advanced OOP concepts in C++

Presentedby:

Sathyan.B

System Overview

Core Components

Customer Class

Baseclassmanagingcustomer details, identification, and credit information

Loan Class

Derived classhandling loan amount, interest rates, tenure, and EMI processing

Operator Overloading

CustomoperatorsforEMIcalculations and loan comparisons

Exception Handling

Robustvalidationforinputdata and calculation errors

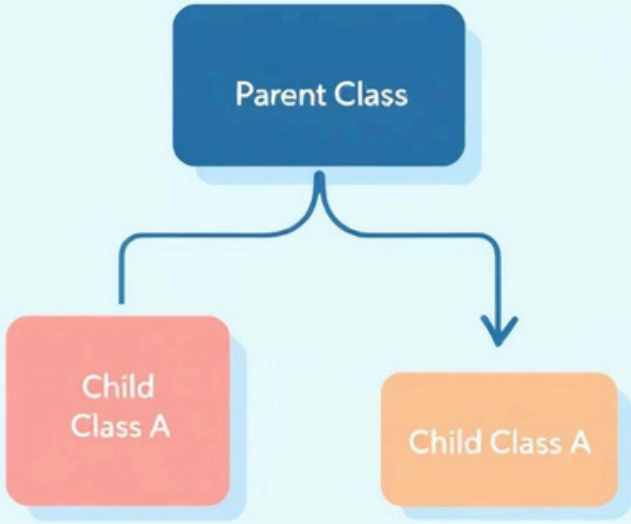
Class Hierarchy Design

Customer Base Class

- Stores customerID, name,and credit score
- Validates customer eligibility
- Provides base data for loan processing

Loan Derived Class

- Inherits customer properties
- Adds loan-specific attributes
- Implements EMI calculation logic



Customer Class Implementation

```
class Customer{
protected:
string customerId;
string name;
int creditScore;

public:
Customer(string id, string n, int score)
 : customerId(id), name(n), creditScore(score) {
 if (score < 300 || score > 900)
throw invalid_argument("Invalid credit score");
}

 bool isEligible() const {
 return creditScore >= 650;
}

 void displayInfo() const {
 cout << "Customer: " << name
 << " (ID: " << customerId << ")\n"
 << "Credit Score: " << creditScore << endl;
} };
```

Loan Class with Inheritance

```
class Loan:publicCustomer{
private:
double principal;
double interestRate;
int tenure; // in months

public:
Loan(string id, string n, int score,
 double p, double rate, int t)
 : Customer(id, n, score),
 principal(p), interestRate(rate), tenure(t) {
 if (p <= 0 || rate < 0 || t <= 0)
throw invalid_argument("Invalid loan parameters");
if (!isEligible())
throw runtime_error("Customer not eligible");
}

 // EMI calculation method
 double calculateEMI() const {
double monthlyRate = interestRate / (12 * 100);
return (principal * monthlyRate *
 pow(1 + monthlyRate, tenure)) /
 (pow(1 + monthlyRate, tenure) - 1);
}
};
```

Operator Overloading Magic

1

Addition Operator (+)

Combinestwo loansto calculate total EMI burden for a customer

2

Comparison Operator (>)

Compares loanamounts to determine which is larger or more expensive

3

Output Operator (<<)

Formatsloandetailsforclean console output and reporting

Operator Overloading Code

```
class Loan:public Customer{
 // ... previous code ...

 // Overload + to add EMIs
 double operator+(const Loan& other) const {
return this->calculateEMI() + other.calculateEMI();
}

 // Overload > to compare loan amounts
 bool operator>(const Loan& other) const {
return this->principal > other.principal;
}

 // Overload << for output
 friend ostream& operator<<(ostream& os,
 const Loan& loan) {
 loan.displayInfo();
os << "Loan Amount: ¹" << loan.principal << "\n"
<< "Interest Rate: " << loan.interestRate << "%\n"
<< "Tenure: " << loan.tenure << " months\n"
<< "Monthly EMI: ¹" << fixed
 << setprecision(2) << loan.calculateEMI();
 return os;
}
};
```

Exception Handling Framework

Input Validation

Checksfornegativeamounts,invalid credit scores, or zero tenure values at object creation

Eligibility Verification

Ensurescustomermeetsminimum credit score requirements before loan approval

Runtime Error Handling

Catchescalculationerrors,division by zero, and mathematical exceptions during processing

Graceful Recovery

Providesmeaningfulerrormessages and prevents system crashes from invalid operations

Complete Program & Output

```
int main() {
    try {
        Loan loan1("C001", "Rajesh Kumar",
                   720, 500000, 8.5, 60);
        Loan loan2("C001", "Rajesh Kumar",
                   720, 200000, 9.0, 36);

        cout << loan1 << "\n\n";
        cout << loan2 << "\n\n";
        if (loan1 > loan2)

            cout << "Loan 1 is larger\n\n";

        double totalEMI = loan1 + loan2;
        cout << "Total EMI: ₹"
             << fixed << setprecision(2)
             << totalEMI << endl;

    } catch (const exception& e) {
        cerr << "Error: "
             << e.what() << endl;
}   }
    return 0;
```

Sample Output

Customer: Rajesh Kumar (ID: C001)
Credit Score: 720
Loan Amount: ₹500000
Interest Rate: 8.5%
Tenure: 60 months
Monthly EMI: ₹10253.63

Customer: Rajesh Kumar (ID: C001)
Credit Score: 720
Loan Amount: ₹200000
Interest Rate: 9.0%
Tenure: 36 months
Monthly EMI: ₹6362.82
Loan 1 is larger

Total EMI: ₹16616.45

Key Takeaways

Inheritancesimplifies code

CustomerpropertiesarereusedinLoan class, reducing redundancy and improving maintainability

Operator overloading adds elegance

Mathematical operations on loan objects become intuitive and readable, enhancing code quality

Exception handling ensures robustness

System gracefully manages errors, providing clear feedback and preventing unexpected crashes