Experiment No. 7 ->

```
#include <iostream>
#include <string>
#include <exception>

class InsufficientFundsException : public std::exception {
public:
    const char* what() const noexcept override {
        return "Error: Insufficient funds!";
    }
};
class InvalidTransactionException : public std::exception {
public:
    const char* what() const noexcept override {
        return "Error: Invalid transaction amount!";
    }
};
class DivideByZeroException : public std::exception {
public:
    const char* what() const noexcept override {
        return "Error: Cannot divide by zero!";
    }
};
class BankAccount {
private:
    std::string ownerName;
    double balance;
```

```
public:
    BankAccount(const std::string& name, double initialDeposit) : ownerName(name),
balance(0.0) {
        if (initialDeposit < 0) {
            throw InvalidTransactionException();
        }
        balance = initialDeposit;
        std::cout << "Creating Account for " << ownerName << " with Initial Deposit: ₹" <<
balance << "\n";
    }
    void deposit(double amount) {
        if (amount < 0) {
            throw InvalidTransactionException();
        }
        balance += amount;
        std::cout << "Depositing ₹" << amount << " into " << ownerName << "'s Account\n";
    }
    void withdraw(double amount) {
        if (amount < 0) {
            throw InvalidTransactionException();
        }
        if (amount > balance) {
            throw InsufficientFundsException();
        }
        balance -= amount;
        std::cout << "Withdrawing ₹" << amount << " from " << ownerName << "'s Account\n";
    }
    void transfer(BankAccount& to, double amount) {
        if (amount < 0) {
            throw InvalidTransactionException();
        }
        if (amount > balance) {
            throw InsufficientFundsException();
        }
        balance -= amount;
        to.balance += amount;
        std::cout << "Transferring ₹" << amount << " from " << ownerName
                  << " to " << to.ownerName << "\n";
    }
    void divideBalance(double divisor) {
        if (divisor == 0) {
            throw DivideByZeroException();
        }
        balance /= divisor;
        std::cout << "Dividing Balance by " << divisor << "\n";
    }
    void display() const {
        std::cout << "Account Holder: " << ownerName << "\n"
                  << "Current Balance: ₹" << balance << "\n\n";
    }
};

int main() {
    try {
        // Creating two accounts
        BankAccount rahul("Rahul Sharma", 5000);
        BankAccount priya("Priya Verma", 0);
        rahul.deposit(1000);
        try {
            rahul.withdraw(7000);
        } catch (const std::exception& e) {
            std::cerr << e.what() << "\n";
        }
        rahul.transfer(priya, 3000);
        rahul.display();
        priya.display();
        try {
            rahul.divideBalance(0);
        } catch (const std::exception& e) {
            std::cerr << e.what() << "\n";
        }
        try {
            priya.deposit(-100);
        } catch (const std::exception& e) {
            std::cerr << e.what() << "\n";
        }

    } catch (const std::exception& e) {
        std::cerr << "Unhandled Exception: " << e.what() << "\n";
    }

    return 0;
}
```