

# Case Study 1

```
1 #include <iostream>
2 using namespace std;
3
4 #define RS "Rs." // Macro for Rupee symbol
5
6 // ----- Base Class -----
7 class Account {
8 protected:
9     int accountNumber;
10    string name;
11    double balance;
12 public:
13     Account(int acctNo, string acctName, double bal)
14         : accountNumber(acctNo), name(acctName), balance(bal) {}
15
16     virtual void deposit(double amount) {
17         cout << RS << amount << " deposited. New balance: " << RS << balance << "\n";
18         balance += amount;
19     }
20
21     virtual void withdraw(double amount) {
22         if (amount > balance) {
23             cout << "Insufficient balance!\n";
24         } else {
25             balance -= amount;
26             cout << RS << amount << " withdrawn. New balance: " << RS << balance << "\n";
27         }
28     }
29
30     virtual void applyInterestPenalty() = 0; // Pure virtual
31
32     // Operator overloading for fund transfer
33     void operator=(Account &receiver) {
34         double amount;
35         cout << "Enter amount to transfer from: " << name;
36         cin >> amount;
37         cout << "Receiver name: " << receiver.name << " : " << RS;
38         cin >> amount;
39
40         if (amount > balance) {
41             cout << "Transfer failed. Insufficient funds.\n";
42         } else {
43             balance -= amount;
44             receiver.balance += amount;
45             cout << "Transferred " << RS << amount << " successfully.\n";
46         }
47     }
48
49     virtual void display() const {
50         cout << "Account No: " << accountNumber;
51         cout << " Name: " << name;
52         cout << " Balance: " << RS << balance << "\n";
53     }
54
55     int getAcctNo() const { return accountNumber; }
56     double getBalance() const { return balance; }
57     string getAcctName() const { return name; }
58 };
59
60 // ----- Derived Classes -----
61 class SavingsAccount : public Account {
62 public:
63     SavingsAccount(int acctNo, string acctName, double bal)
64         : Account(acctNo, acctName, bal) {}
65
66     void applyInterestPenalty() override {
67         double interest = balance * 0.05; // 5% annual
68         balance += interest;
69         cout << "5% interest added. New balance: " << RS << balance << "\n";
70     }
71 };
72
73 class CurrentAccount : public Account {
74 public:
75     CurrentAccount(int acctNo, string acctName, double bal)
76         : Account(acctNo, acctName, bal) {}
77
78     void applyInterestPenalty() override {
79         if (balance < 1000) {
80             balance -= 1000;
81             cout << "Balance " << RS << "1000. Penalty of " << RS << "500 applied. Balance: " << RS << balance << "\n";
82         } else {
83             cout << "No penalty applied.\n";
84         }
85     }
86 };
87
88 // ----- Bank Class -----
89 class Bank {
90     vector<Account*> accounts;
91     int nextAcctNo = 1001;
92 public:
93     void createAccount() {
94         string name;
95         int type;
96         double initialDeposit;
97
98         cout << "Enter name: ";
99         cin >> name;
100        cout << "Account type (1 - Savings, 2 - Current): ";
101        cin >> type;
102        cout << "Enter initial deposit: " << RS;
103        cin >> initialDeposit;
104
105        if (type == 1)
106            accounts.push_back(new SavingsAccount(nextAcctNo++, name, initialDeposit));
107        else
108            accounts.push_back(new CurrentAccount(nextAcctNo++, name, initialDeposit));
109
110        cout << "Account created successfully!\n";
111    }
112
113     Account* findAccount(int acctNo) {
114         for (auto acc : accounts)
115             if (acc->getAcctNo() == acctNo) return acc;
116         return nullptr;
117     }
118
119     void depositMoney() {
120         int acctNo; double amount;
121         cout << "Enter account number: "; cin >> acctNo;
122         cout << "Enter deposit amount: " << RS; cin >> amount;
123
124         Account* acc = findAccount(acctNo);
125         if (acc) acc->deposit(amount);
126         else cout << "Account not found!\n";
127     }
128
129     void withdrawMoney() {
130         int acctNo; double amount;
131         cout << "Enter account number: "; cin >> acctNo;
132         cout << "Enter withdrawal amount: " << RS; cin >> amount;
133
134         Account* acc = findAccount(acctNo);
135         if (acc) acc->withdraw(amount);
136         else cout << "Account not found!\n";
137     }
138
139     void transferFunds() {
140         int fromAcct, toAcct;
141         cout << "Enter sender account no: "; cin >> fromAcct;
142         cout << "Enter receiver account no: "; cin >> toAcct;
143
144         Account* sender = findAccount(fromAcct);
145         Account* receiver = findAccount(toAcct);
146
147         if (sender && receiver) {
148             sender->transfer(receiver); // operator overloading
149         } else {
150             cout << "Invalid account number(s).\n";
151         }
152     }
153
154     void applyYearEnd() {
155         for (auto acc : accounts) {
156             acc->applyInterestPenalty();
157         }
158     }
159
160     void showAllAccounts() {
161         for (auto acc : accounts) acc->display();
162     }
163
164     // Show balance of specific account
165     void showBalance() {
166         int acctNo;
167         cout << "Enter account number: ";
168         cin >> acctNo;
169
170         Account* acc = findAccount(acctNo);
171         if (acc) {
172             cout << "Account No: " << acc->getAcctNo()
173                 << " Name: " << acc->getAcctName()
174                 << " Balance: " << RS << acc->getBalance() << "\n";
175         } else {
176             cout << "Account not found!\n";
177         }
178     }
179
180     // Delete an account
181     void deleteAccount() {
182         int acctNo;
183         cout << "Enter account number to delete: ";
184         cin >> acctNo;
185
186         for (auto it = accounts.begin(); it != accounts.end(); ++it) {
187             if ((*it)->getAcctNo() == acctNo) {
188                 delete *it; // Free memory
189                 accounts.erase(it);
190                 cout << "Account deleted successfully!\n";
191                 return;
192             }
193         }
194         cout << "Account not found!\n";
195     }
196
197     ~Bank() {
198         for (auto acc : accounts) delete acc;
199     }
200 };
201
202 // ----- Main -----
203 int main() {
204     Bank bank;
205     int choice;
206
207     do {
208         cout << "\n----- Bank Menu -----!\n";
209         cout << "1. Create Account!\n";
210         cout << "2. Deposit Money!\n";
211         cout << "3. Withdraw Money!\n";
212         cout << "4. Transfer Funds!\n";
213         cout << "5. Apply Interest/Penalty!\n";
214         cout << "6. Show All Accounts!\n";
215         cout << "7. Show Balance!\n"; // New Option
216         cout << "8. Delete Account!\n"; // New Option
217         cout << "9. Exit!\n";
218         cout << "Enter choice: ";
219         cin >> choice;
220
221         switch(choice) {
222             case 1: bank.createAccount(); break;
223             case 2: bank.depositMoney(); break;
224             case 3: bank.withdrawMoney(); break;
225             case 4: bank.transferFunds(); break;
226             case 5: bank.applyInterestPenalty(); break;
227             case 6: bank.showAllAccounts(); break;
228             case 7: bank.showBalance(); break;
229             case 8: bank.deleteAccount(); break;
230             case 9: cout << "Exiting...\n"; break;
231             default: cout << "Invalid choice!\n";
232         }
233     } while (choice != 9);
234
235     return 0;
236 }
```