OBJECT ORIENTED PROGRAMMING

Assignment #4

Name: Muhammad Abbas

Reg: L1F24BSDS0105

Section: DC-2

Submitted To: Sir Atir Raza Sh

Faculty of Information and Technology

University of Central Punjab, Lahore

Git Hub Link:

https://github.com/Muhammad-Abbas-macc/OOP-Assignment-4.git

Code:

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

using namespace std;

class BankAccount {
private:
    string accountHolder;
    double balance;

public:
    // Constructor - throws exception if balance is negative
    BankAccount(string name, double initialBalance) {
        cout << "Constructor Creating account for: " << name <<
endl;


        if (initialBalance < 0) {
            cout << "Constructor Error: Negative balance!" <<
endl;
            throw string("Cannot create account with negative
balance!");
        }

        accountHolder = name;
        balance = initialBalance;
        cout << "Constructor Account created! Balance: " <<
balance << endl;
    }

    ~BankAccount() {
        cout << "Destructor Deleting account: " << accountHolder
<< endl;
    }

    void withdraw(double amount) {
        cout << "Trying to withdraw : " << amount << endl;

        if (amount > balance) {
            cout << "Error: Not enough money!" << endl;
            throw string("Insufficient funds!");
        }

        balance = balance - amount;
        cout << "Successful! Withdraw $" << amount << endl;
```

```
        cout << "New balance: $" << balance << endl;
    }

    void display() {
        cout << "--- Account Info ---" << endl;
        cout << "Name: " << accountHolder << endl;
        cout << "Balance: $" << balance << endl;
        cout << "----------------------" << endl;
    }
};

int main() {
    cout << "Valid Account"<<endl;
    cout << "------------------------" << endl;

    BankAccount* acc1 = nullptr;

    try {
        acc1 = new BankAccount("Ali", 1500.6);
        acc1->display();
        acc1->withdraw(200.0);
        acc1->display();

        delete acc1;
        acc1 = nullptr;
    }
    catch (string error) {
        cout << "Caught error: " << error << endl;
        if (acc1 != nullptr) {
            delete acc1;
        }
    }

    cout << endl;
    cout << "Negative Balance (ERROR)" << endl;
    cout << "-----------------------------------" << endl;

    BankAccount* acc2 = nullptr;

    try {
        acc2 = new BankAccount("Samir", -500.0);
        acc2->display();
    }
    catch (string error) {
        cout << " Caught error: " << error << endl;
        cout << " Account was not created.\n" << endl;
    }

    cout << endl;
    cout << "Withdraw Too Much (ERROR)\n";
    cout << "-----------------------------------" << endl;
```

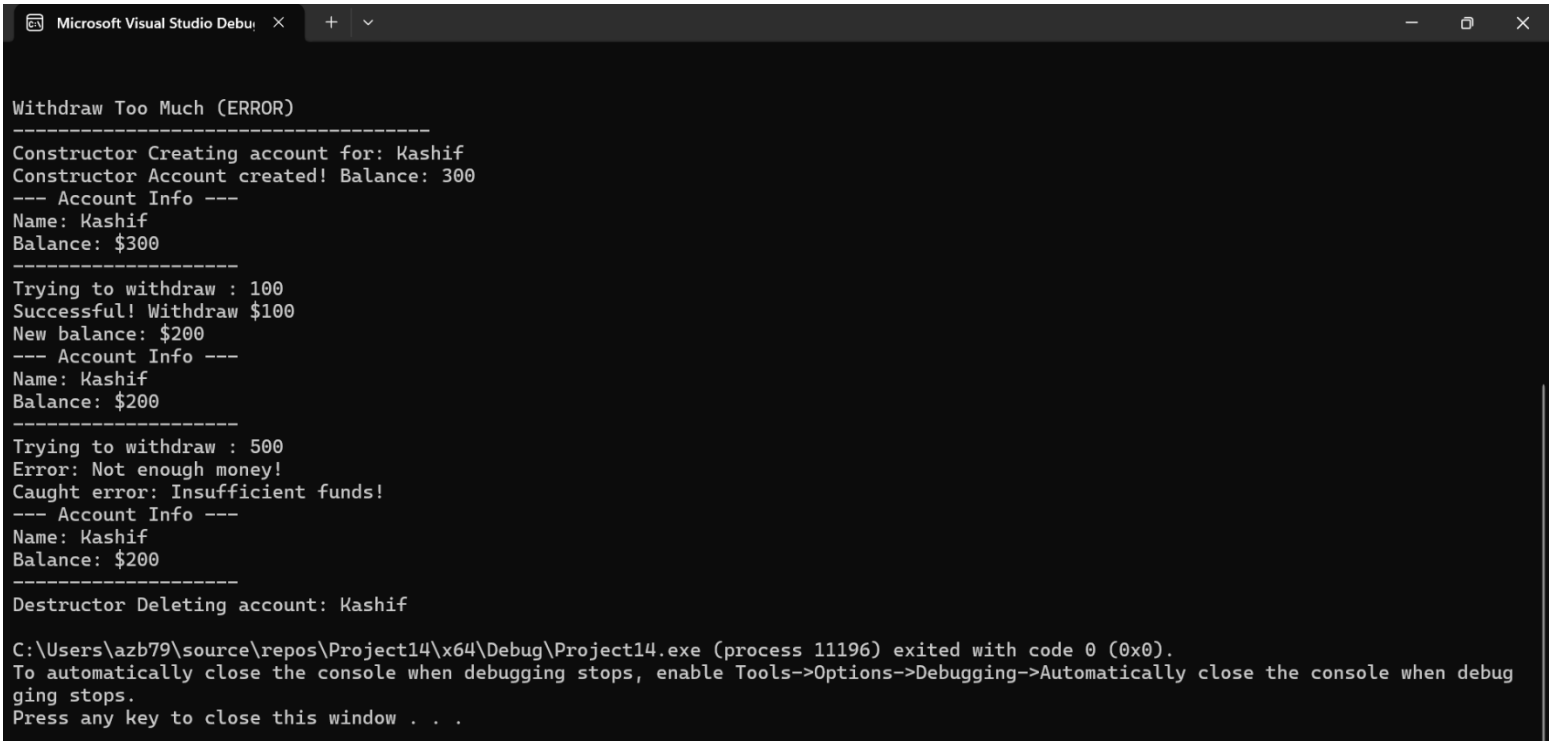
```
    BankAccount* acc3 = nullptr;

    try {
        acc3 = new BankAccount("Kashif", 300.0);
        acc3->display();

        acc3->withdraw(100.0);
        acc3->display();

        acc3->withdraw(500.0);  // This will throw exception!

        acc3->display();
    }
    catch (string error) {
        cout << "Caught error: " << error << endl;
        if (acc3 != nullptr) {
            acc3->display();
            delete acc3;
        }
    }
    return 0;
}
```


Microsoft Visual Studio Debu
Withdraw Too Much (ERROR)
Constructor Creating account for: Kashif
Constructor Account created! Balance: 300
--- Account Info ---
Name: Kashif
Balance: $300
Trying to withdraw : 100
Successful! Withdraw $100
New balance: $200
--- Account Info ---
Name: Kashif
Balance: $200
Trying to withdraw : 500
Error: Not enough money!
Caught error: Insufficient funds!
--- Account Info ---
Name: Kashif
Balance: $200
Destructor Deleting account: Kashif
C:\Users\azb79\source\repos\Project14\x64\Debug\Project14.exe (process 11196) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debug
ging stops.
Press any key to close this window . . .

Assignment #4:-

Code:-

```
#include <iostream>
#include <string>
using namespace std;

class BankAccount {
 private:
   string accountHolder;
   double balance;

 public:
   // Constructor throws exception if balance is negative
   BankAccount(string name, double initialBalance) {
      cout << "Constructor creating account for: " << name << endl;

   if(initialBalance < 0) {
        cout << "Constructor Error: Negative Balance! " << endl;
        throw string("Cannot create account with negative balance!");
      }
     accountHolder = name;
     balance = initialBalance;
     cout << "Constructor Account created! Balance: " << balance <<
     endl;
   }
  ~BankAccount() {
     cout << "Destructor Deleting account: " << accountHolder << endl;
  }
```

```
void withdraw(double amount){
  cout << "Trying to withdraw : " << amount << endl;
  if(amount > balance) {
    cout <<" Error: Not enough money! " << endl;
    throw string ("Insufficient funds!");
  }
  balance = balance -amount;
  cout << "Successful! Withdraw $" << amount << endl;
  cout << " New balance: $ " << balance << endl;
}
  void display() {
    cout <<"--- Account Info ---" << endl;
    cout << " Name: " << accountHolder << endl;
    cout << " Balance: $" << balance << endl;
    cout <<"---------------- " << endl;
  }
};
int main() {
  cout <<" Valid Account " << endl;
  cout <<"---------------- "<< endl;
  BankAccount *acc1 = nullptr;
```

```
try {
 acc1 = new BankAccount ("Ali ", 1500.6);
 acc1 -> display();
 acc1 -> withdraw(200.0);
 acc1 -> display();

 delete acc1;
 acc1 = nullptr;
}

catch (string error) {
    cout <<"Caught error " << error << endl;
    if (acc1 != nullptr ){
      delete acc1;
    }
  }
 cout << endl;
 cout << "Negative Balance (error) "<< endl;
 cout << " ---------- "<<endl;

 try {
   acc2 = new BankAccount ("Samir ", -500.0);
   acc2 -> display(); // This will not run.
 }
 catch (string error){
   cout << "Caught error : " << error <<endl;
   cout <<"Account was not created.\n" <<endl;
}
```

```
cout << endl;
cout << "Withdraw too Much (ERROR) " \n";
cout <<" -- -- -- ---- -- -- "<<endl;

BankAccount *acc3 = nullptr;
try {
  acc3 = new BankAccount ("Kashif", 300.0);
  acc3 -> display();
  acc3 -> withdraw(100.0);
  acc3 -> display();
  acc3 -> withdraw(300.0); // This will throw exception!
  acc3 -> display();
}
catch (string error) {
   cout << "Caught error: " << error <<endl;
   if (acc3 != nullptr) {
        acc3 -> display();
        delete acc3;
   }
  }
  return 0;
}
```