**Program-1**

#include<iostream>

using namespace std;

class shape

{

private:

public:

virtual inline double area(){

return 0;

}

};

class circle:public shape{

private:

int r;

public:

circle(){

cout<<"Enter Radius of the Circle"<<endl;

cin>>r;

}

inline double area(){

return 3.14\*r\*r;

}

};

class square:public shape{

private:

int b;

public:

square(){

cout<<"Enter Breadth of the Square"<<endl;

cin>>b;

}

inline double area(){

return b\*b;

}

};

class triangle: public shape{

private:

int h,b;

public:

triangle(){

cout<<"Enter Height and Breadth of the Triangle"<<endl;

cin>>h>>b;

}

inline double area(){

return 0.5\*h\*b;

}

};

int main()

{

square s;

cout<<"Area of the Square "<<s.area()<<endl;

triangle t;

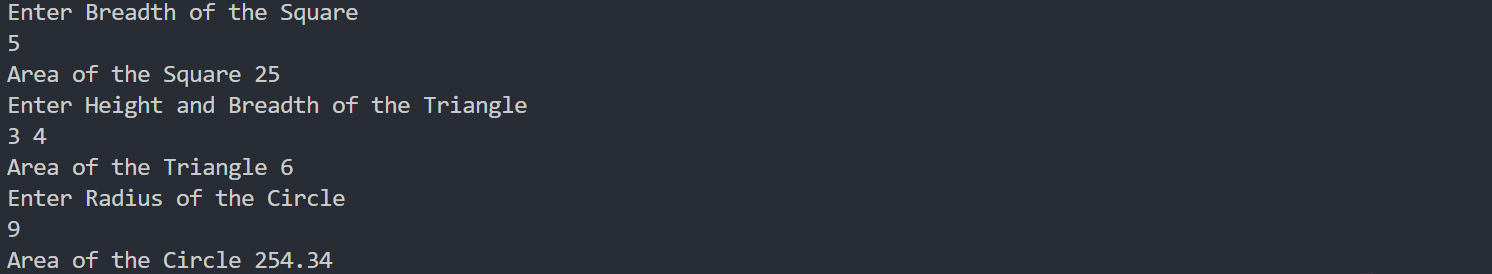
cout<<"Area of the Triangle "<<t.area()<<endl;

circle c;

cout<<"Area of the Circle "<<c.area()<<endl;

return 0;

}



**Program-2**

#include <iostream>

using namespace std;

class Account

{

public:

long accNo;

string cname;

double balance;

void getData()

{

cout << "Enter Account Number " << endl;

cin >> accNo;

cout << "Enter Customer Name " << endl;

getline(cin>>ws,cname);

cout << "Enter Available Balance " << endl;

cin >> balance;

balance = balance + 1000;

}

virtual void displayBalance(){

cout << "Available Balance is " << Account::balance << endl;

}

};

class Savings : public Account

{

public:

double minimum\_acc = 1000;

void deposit(int money\_amt)

{

balance = balance + money\_amt;

cout << "Balance After Deposit is " << balance << endl;

}

void withdraw(int money\_amt)

{

if (balance - money\_amt > minimum\_acc)

{

balance = balance - money\_amt;

}

else

{

cout << "Insufficient Balance \n";

}

}

void displayBalance()

{

cout << "Available Balance is " << Account::balance << endl;

}

};

class Current : public Account

{

public:

int overDueAmt;

void overDueInp(int leftMoney)

{

// Amount of money to be paid after Intrest

overDueAmt = leftMoney \* intrest \* days;

}

double intrest = 5.5;

int days = 2;

void withDraw(int money\_amt)

{

if (balance - money\_amt > overDueAmt)

{

balance = balance - money\_amt;

}

else

{

cout << "Insufficient Balance \n";

}

}

void displayBalance()

{

cout << "Available Balance is " << Account::balance << endl;

}

};

int main()

{

Savings s1;

s1.getData();

s1.deposit(500);

s1.withdraw(1500);

s1.displayBalance();

Current c1;

c1.getData();

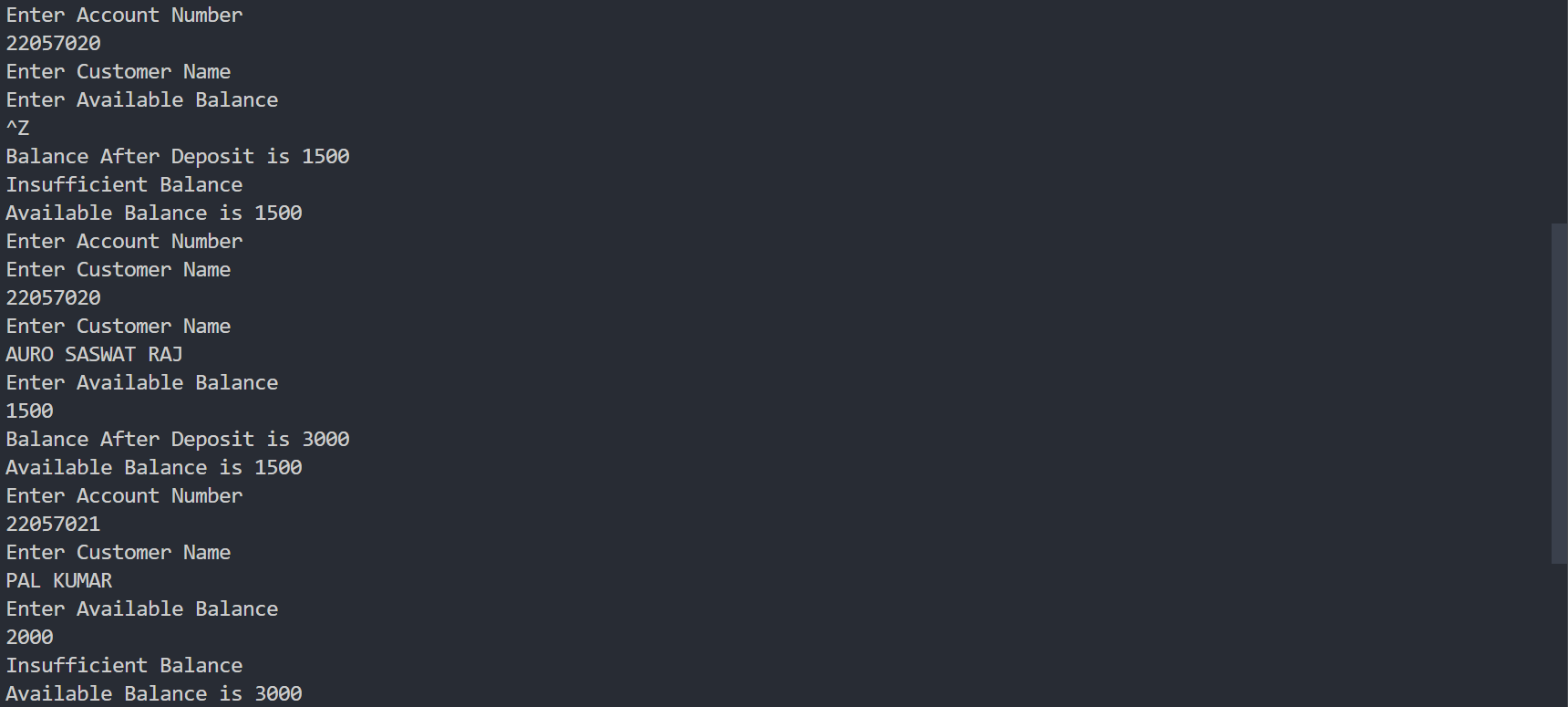
c1.overDueInp(5600);

c1.withDraw(500);

c1.displayBalance();

return 0;

}

****

**Program-3**

#include <iostream>

using namespace std;

class Base{ // abstract base class

public:

void func1(){ // normal member function

cout << "\nHello Base function 1";

}

virtual void func2(){ // virtual member function

cout << "\nHi Base function 2";

}

virtual void func3()=0; // pure virtual function

};

// Since base class has a pure virtal function

// it is also known as abstract class

class Derived : public Base{

public:

void func1(){ // gets overloaded by func1() of base class

cout << "\nHello Derived function 1";

}

void func2(){ // derived func2() overrides base class func2()

cout << "\nHi Derived function 2";

}

void func3(){

// func3() inside child class must be defined, otherwise

// compiler gives errors

cout << "\nHola Derived function 3";

}

};

int main(){

Base \*b;

Derived d;

b = &d;

b->func1();

b->func2();

b->func3();

return 0;

}

Output

