**Name:** Waleed Amjad

**Roll No:** 2023-BS-Ai-054

**Department:** AI

**Sec:** A

**Semester:** 2nd

**Subject:** OOP

***Home Assignment***



**Problem 1:**

**Code:**

// File: Problem 1 (Data of Publishing Company).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\* Imagine a publishing company that markets both book and audiocassete versions of its works. Create a

class publication that stores the title (a string) and price (type float) of a publication. From this class

derive two classes: book, which adds a page count (type int), and tape, which adds a playing me in

minutes (type float). Each of these three classes should have a getdata() func on to get its data from the

user at the keyboard, and a putdata() func on to display its data. Write a main() program to test the

book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and

then displaying the data with putdata(). \*/

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

void getData() {

cout << "Enter title: ";

cin.ignore(); // To ignore any leftover newline characters in the input buffer

getline(cin, title);

cout << "Enter price: ";

cin >> price;

}

void putData() const {

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Book : public Publication {

private:

int pageCount;

public:

void getData() {

Publication::getData();

cout << "Enter page count: ";

cin >> pageCount;

}

void putData() const {

Publication::putData();

cout << "Page count: " << pageCount << endl;

}

};

class Tape : public Publication {

private:

float playingTime;

public:

void getData() {

Publication::getData();

cout << "Enter playing time in minutes: ";

cin >> playingTime;

}

void putData() const {

Publication::putData();

cout << "Playing time: " << playingTime << " minutes" << endl;

}

};

int main() {

Book myBook;

Tape myTape;

cout << "Enter details for book:" << endl;

myBook.getData();

cout << "\nEnter details for tape:" << endl;

myTape.getData();

system("CLS");

cout << "\nBook details:" << endl;

myBook.putData();

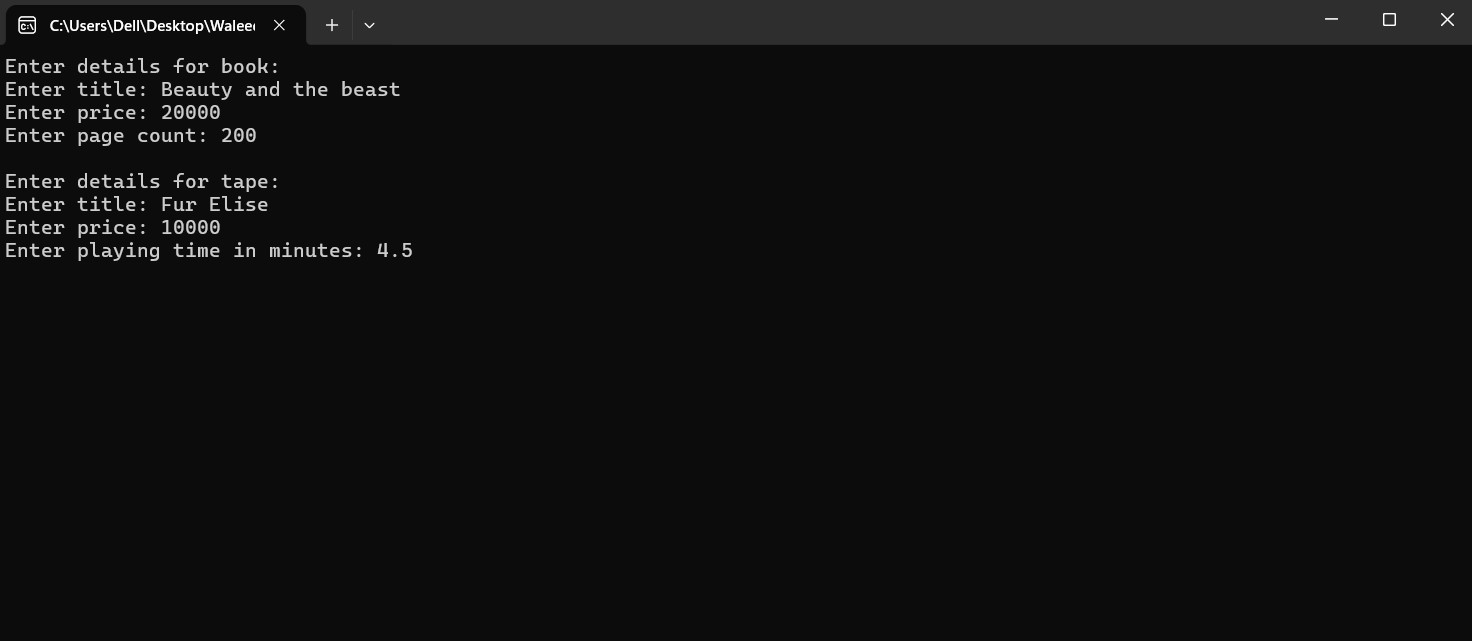
cout << "\nTape details:" << endl;

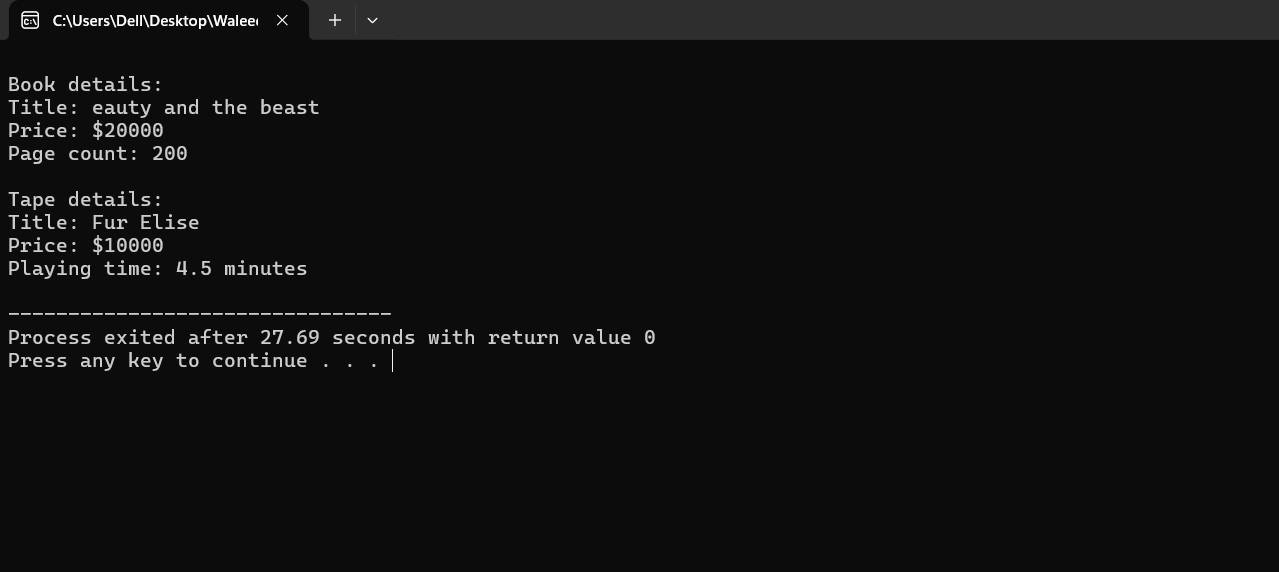
myTape.putData();

return 0;

}

**Output:**





**Problem 2:**

**Code:**

// File: Problem 2 (Data of Publishing Company continued).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\* Start with the publication, book, and tape classes of Quesion 1. Add a base class sales that holds an

array of three floats so that it can record the dollar sales of a particular publication for the last three

months. Include a getdata() function to get three sales amounts from the user, and a putdata() func on

to display the sales figures. Alter the book and tape classes so they are derived from both publication

and sales. An object of class book or tape should input and output sales data along with its other data.

Write a main() func on to create a book object and a tape object and exercise their input/output

capabilities. \*/

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

void getData() {

cout << "Enter title: ";

cin.ignore(); // To ignore any leftover newline characters in the input buffer

getline(cin, title);

cout << "Enter price: ";

cin >> price;

}

void putData() const {

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Sales {

protected:

float sale[3];

public:

void getData(){

cout<<"\nEnter the sales for the last three months:\n";

for(int i=0;i<3;i++){

cout<<"Month"<<i+1<<": ";

cin>>sale[i];

}

}

void putData() const{

cout<<"Sales for the last three months:\n";

for(int i=0;i<3;i++){

cout<<"Month"<<i+1<<":$ "<<sale[i]<<endl;

}

}

};

class Book : public Publication , public Sales {

private:

int pageCount;

public:

void getData() {

Publication::getData();

cout << "Enter page count: ";

cin >> pageCount;

Sales::getData();

}

void putData() const {

Publication::putData();

cout << "Page count: " << pageCount << endl;

Sales::putData();

}

};

class Tape : public Publication , public Sales {

private:

float playingTime;

public:

void getData() {

Publication::getData();

cout << "Enter playing time in minutes: ";

cin >> playingTime;

Sales::getData();

}

void putData() const {

Publication::putData();

cout << "Playing time: " << playingTime << " minutes" << endl;

Sales::putData();

}

};

int main() {

Book myBook;

Tape myTape;

cout << "Enter details for book:" << endl;

myBook.getData();

cout << "\nEnter details for tape:" << endl;

myTape.getData();

system("CLS");

cout << "\nBook details:" << endl;

myBook.putData();

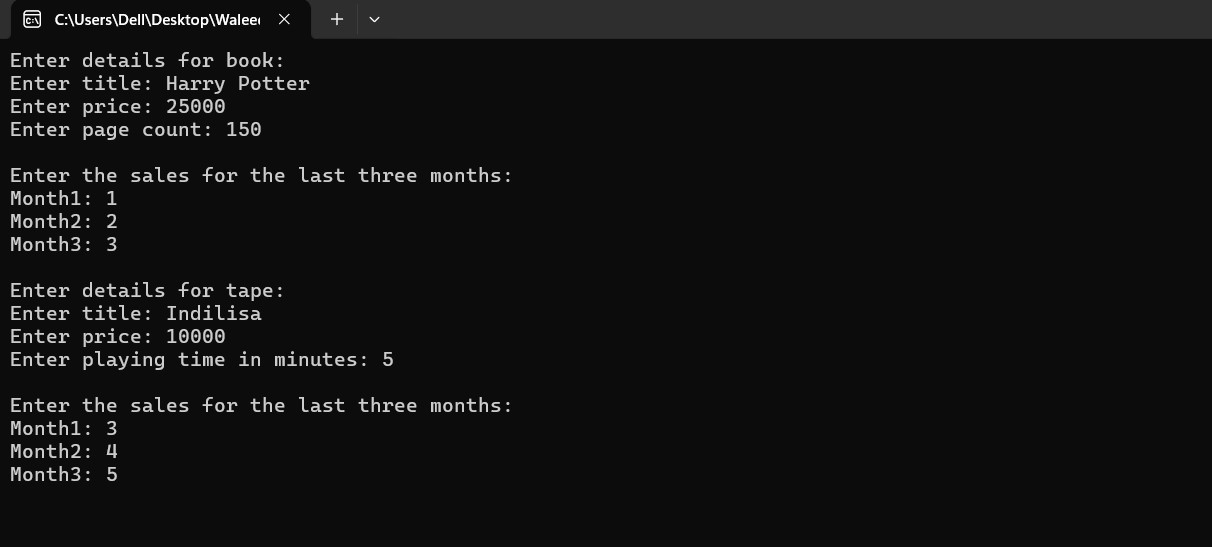
cout << "\nTape details:" << endl;

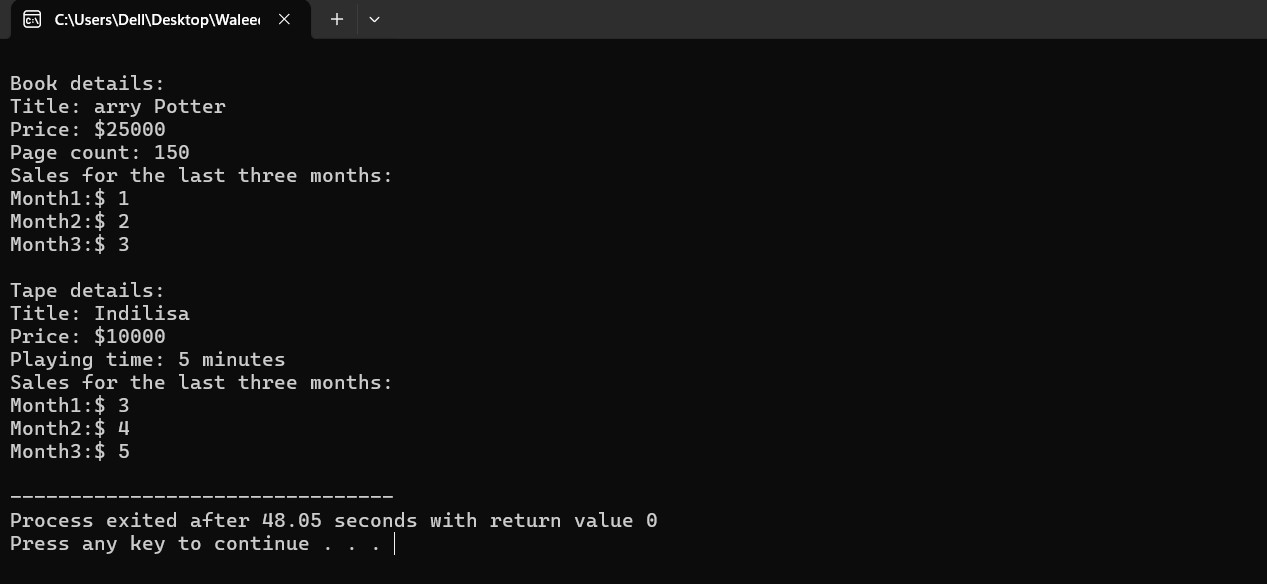
myTape.putData();

return 0;

}

**Output:**





**Problem 3:**

**Code:**

// File: Problem 3 (Data of Publishing Company continued......).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\*Assume that the publisher in Question 1 and 2 decides to add a third way to distribute books: on computer

disk, for those who like to do their reading on their laptop. Add a disk class that, like book and tape, is

derived from publication. The disk class should incorporate the same member functions as the other

classes. The data item unique to this class is the disk type: either CD or DVD. You can use an enum type to

store this item. The user could select the appropriate type by typing c or d. \*/

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

void getData() {

cout << "Enter title: ";

cin.ignore(); // To ignore any leftover newline characters in the input buffer

getline(cin, title);

cout << "Enter price: ";

cin >> price;

}

void putData() const {

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Sales {

protected:

float sale[3];

public:

void getData() {

cout << "\nEnter the sales for the last three months:\n";

for(int i = 0; i < 3; i++) {

cout << "Month " << i + 1 << ": ";

cin >> sale[i];

}

}

void putData() const {

cout << "Sales for the last three months:\n";

for(int i = 0; i < 3; i++) {

cout << "Month " << i + 1 << ": $" << sale[i] << endl;

}

}

};

class Book : public Publication, public Sales {

private:

int pageCount;

public:

void getData() {

Publication::getData();

cout << "Enter page count: ";

cin >> pageCount;

Sales::getData();

}

void putData() const {

Publication::putData();

cout << "Page count: " << pageCount << endl;

Sales::putData();

}

};

class Tape : public Publication, public Sales {

private:

float playingTime;

public:

void getData() {

Publication::getData();

cout << "Enter playing time in minutes: ";

cin >> playingTime;

Sales::getData();

}

void putData() const {

Publication::putData();

cout << "Playing time: " << playingTime << " minutes" << endl;

Sales::putData();

}

};

class Disk : public Publication, public Sales {

private:

enum DiskType {CD, DVD} diskType;

public:

void getData() {

Publication::getData();

char diskTypeInput;

cout << "Enter disk type (c for CD, d for DVD): ";

cin >> diskTypeInput;

if (diskTypeInput == 'c' || diskTypeInput == 'C') {

diskType = CD;}

else {

diskType = DVD;}

Sales::getData();

}

void putData() const {

Publication::putData();

cout << "Disk type: " << (diskType == CD ? "CD" : "DVD") << endl;

Sales::putData();

}

};

int main() {

Book myBook;

Tape myTape;

Disk myDisk;

cout << "Enter details for book:" << endl;

myBook.getData();

cout << "\nEnter details for tape:" << endl;

myTape.getData();

cout << "\nEnter details for disk:" << endl;

myDisk.getData();

system("CLS");

cout << "\nBook details:" << endl;

myBook.putData();

cout << "\nTape details:" << endl;

myTape.putData();

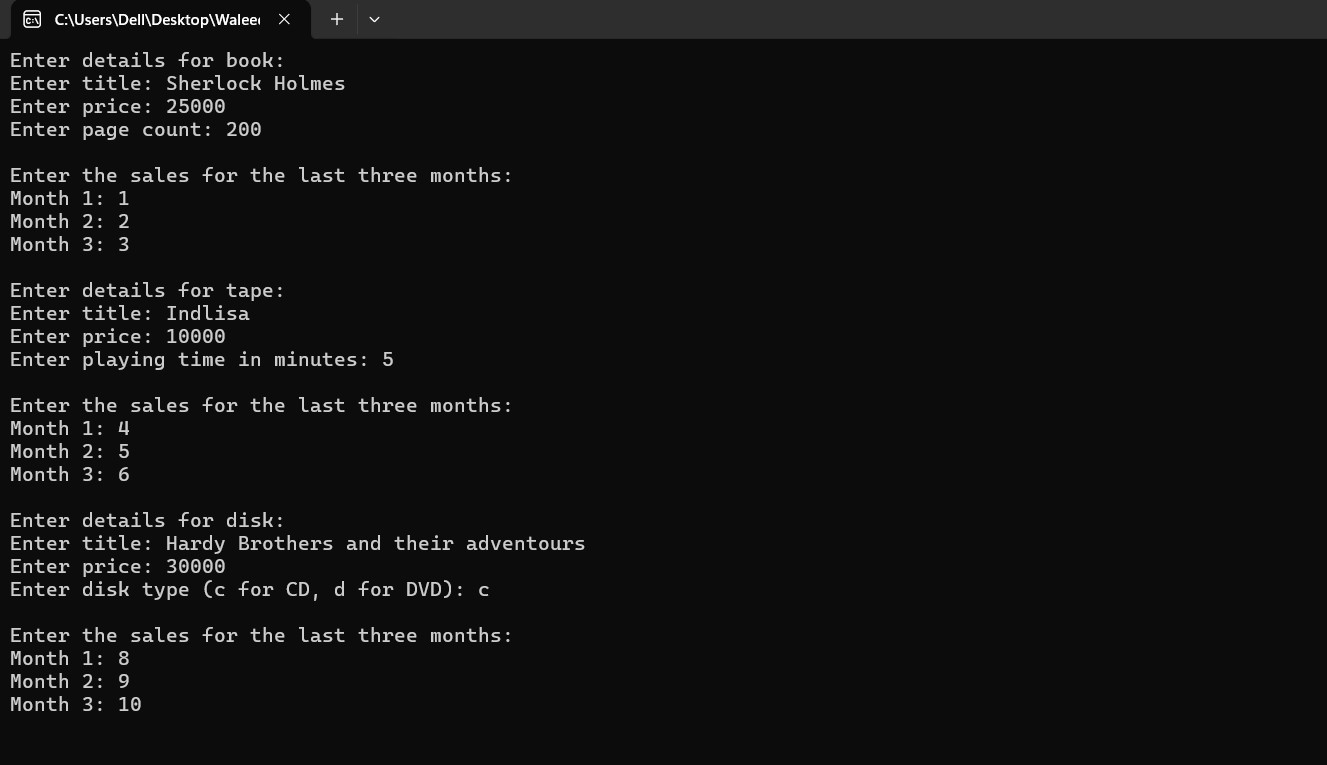
cout << "\nDisk details:" << endl;

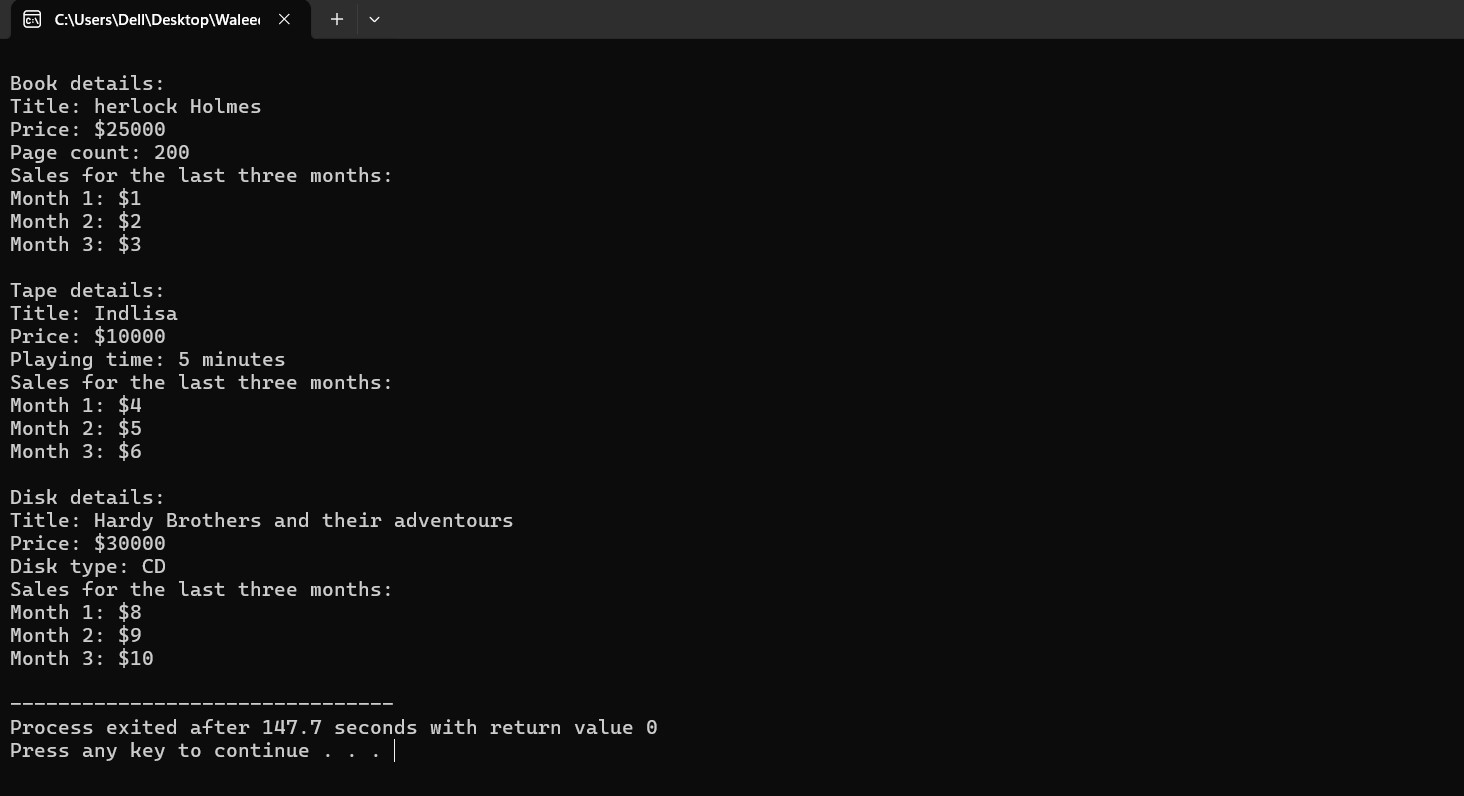
myDisk.putData();

return 0;

}

**Output:**





**Problem 4:**

**Code:**

// File: Problem 4 (Data of Employee).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\*Derive a class called employee2 from the employee class in the EMPLOY program in this chapter. This new

class should add a type double data item called compensation, and also an enum type called period to

indicate whether the employee is paid hourly, weekly, or monthly. For simplicity you can change the

manager, scientist, and laborer classes so they are derived from employee2 instead of employee. However,

note that in many circumstances it might be more in the spirit of OOP to create a separate base class called

compensation and three new classes manager2, scientist2, and laborer2, and use multiple inheritance to

derive these three classes from the original manager, scien st, and laborer classes and from

compensation. This way none of the original classes needs to be modified \*/

#include <iostream>

#include <string>

using namespace std;

// Base class for compensation

class Compensation {

protected:

double compensation;

public:

enum Period { HOURLY, WEEKLY, MONTHLY };

Period period;

};

// Base class for employee

class Employee {

protected:

string name;

long id;

public:

void getData() {

cout << "Enter name: ";

cin.ignore(); // To ignore any leftover newline characters in the input buffer

getline(cin, name);

cout << "Enter ID: ";

cin >> id;

}

void putData() const {

cout << "Name: " << name << endl;

cout << "ID: " << id << endl;

}

};

// Derived class from Employee with compensation details

class Employee2 : public Employee, public Compensation {

public:

void getData() {

Employee::getData();

cout << "Enter compensation: ";

cin >> compensation;

char periodInput;

cout << "Enter pay period (h for Hourly, w for Weekly, m for Monthly): ";

cin >> periodInput;

switch(periodInput) {

case 'h':

case 'H':

period = HOURLY;

break;

case 'w':

case 'W':

period = WEEKLY;

break;

case 'm':

case 'M':

period = MONTHLY;

break;

default:

cout << "Invalid pay period input!" << endl;

}

}

void putData() const {

Employee::putData();

cout << "Compensation: $" << compensation << " per ";

switch(period) {

case HOURLY:

cout << "hour" << endl;

break;

case WEEKLY:

cout << "week" << endl;

break;

case MONTHLY:

cout << "month" << endl;

break;

}

}

};

// Derived classes from Employee2 for specific job roles

class Manager2 : public Employee2 {};

class Scientist2 : public Employee2 {};

class Laborer2 : public Employee2 {};

int main() {

Manager2 manager;

Scientist2 scientist;

Laborer2 laborer;

cout << "Enter details for manager:" << endl;

manager.getData();

cout << "\nEnter details for scientist:" << endl;

scientist.getData();

cout << "\nEnter details for laborer:" << endl;

laborer.getData();

cout << "\nManager details:" << endl;

manager.putData();

cout << "\nScientist details:" << endl;

scientist.putData();

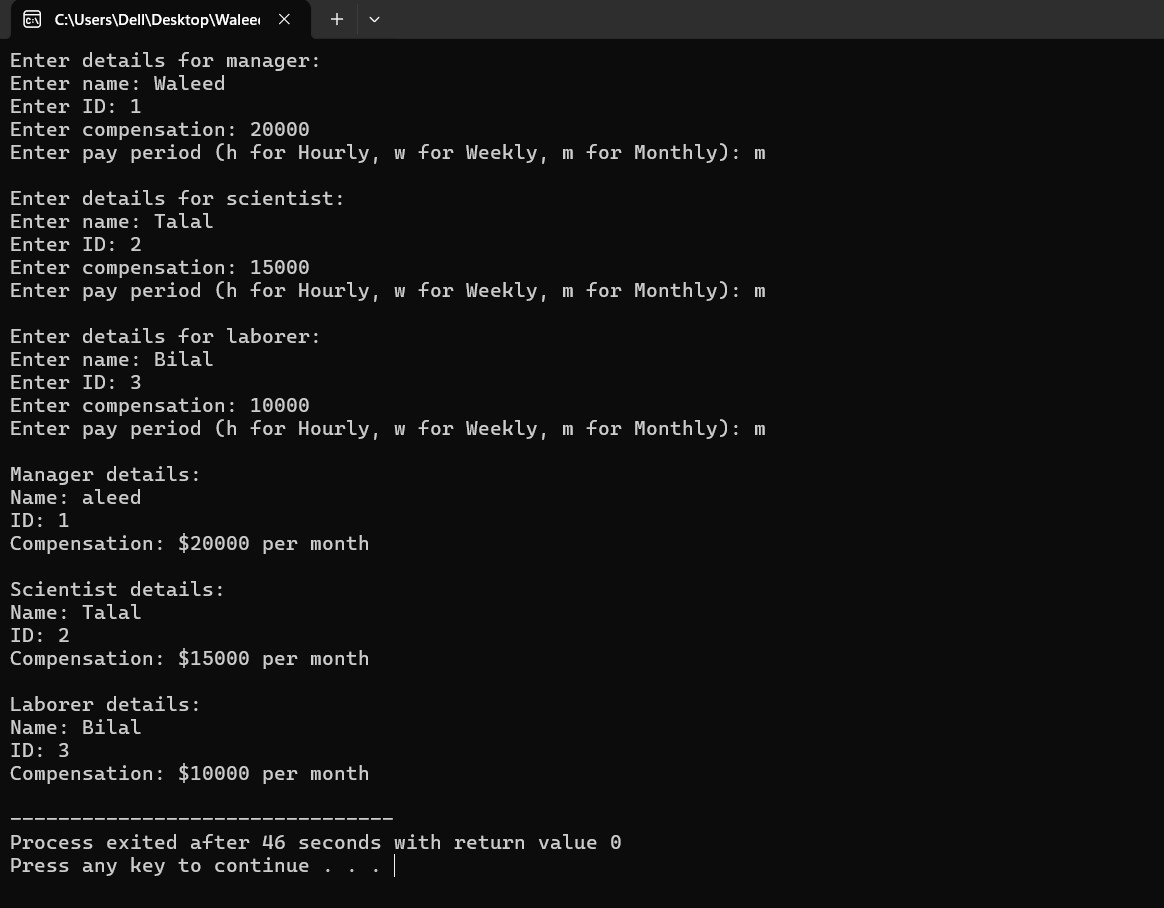
cout << "\nLaborer details:" << endl;

laborer.putData();

return 0;

}

**Output:**



**Problem 5:**

**Code:**

// File: Problem 5 (Hierarchy of Shapes,Rectangle and Circle).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\*Create a simple inheritance hierarchy for a Shape class, Circle class, and Rectangle class. The Shape class

should be the base class, and Circle and Rectangle should be derived classes. Implement the following in

C++:

Shape Class:

Atributes: color (type std::string).

Methods: A constructor to initialize the color and a method printColor to display the color.

Circle Class:

Atributes: radius (type double).

Methods: A constructor to initialize the color and radius, a method calculateArea to calculate the area of

the circle (area = π \* radius \* radius), and a method printArea to display the area.

Rectangle Class:

Atributes: length and width (type double).

Methods: A constructor to initialize the color, length, and width, a method calculateArea to calculate the

area of the rectangle (area = length \* width), and a method printArea to display the area. \*/

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

const double PI=3.142;

//Base Class

class Shape{

protected:

string color;

public:

//Constructor to Initialize color

Shape(const string color) : color(color) {}

//method to print color

void printcolor() const{

cout<<"Color: "<<color<<endl;

}

};

class Circle: public Shape{

private:

double radius;

public:

//Constructor to Initialize color and raduis both

Circle(const string color, double radius): Shape(color) , radius(radius){}

//method to calculate area

double calculatearea() const {

return PI\*(radius\*radius);

}

//method to print area

void printarea() const {

cout<<"Area of Circle = "<<calculatearea()<<endl;

}

};

class Rectangle: public Shape{

private:

double length,width;

public:

//Constructor to initialize color, Length and Width

Rectangle(const string color, double length, double width): Shape(color) , length(length) , width(width) {}

//method to calculate area

double calculatearea(){

return length\*width;

}

//method to print the area of the rectangle

void printarea() {

cout<<"Area of the rectangle is: "<<calculatearea()<<endl;

}

};

int main(){

// create circle

Circle circle("Red",10);

// using the print function

circle.printcolor();

circle.printarea();

cout<<endl;

//create rectangle

Rectangle rectangle("Green",5,15);

// using the print funtion

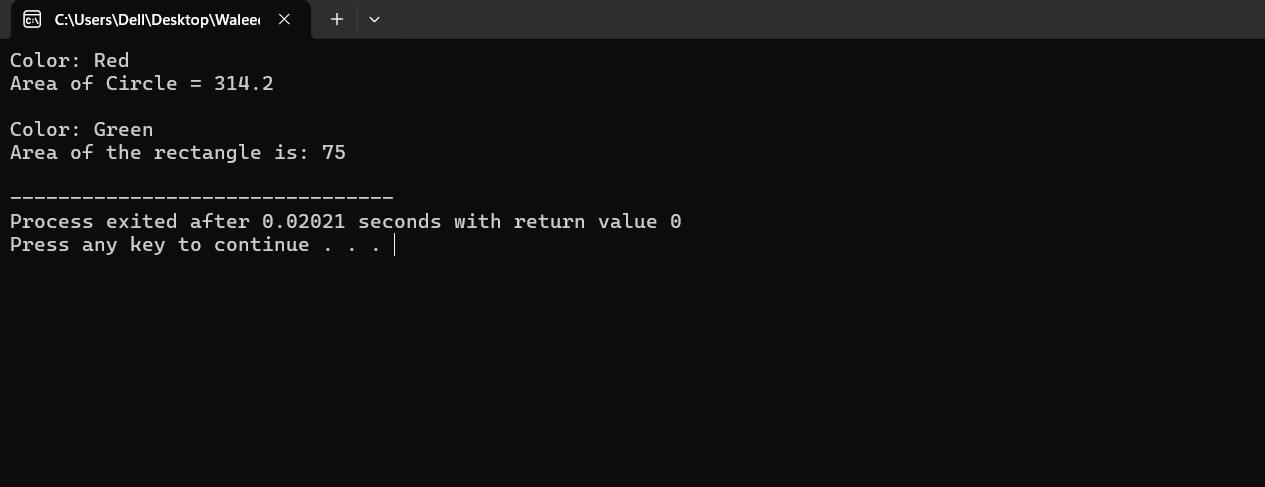
rectangle.printcolor();

rectangle.printarea();

return 0;

}

**Output:**



**Problem 6:**

**Code:**

// File: Problem 6 (Hierarchy of Employee Management system).cpp

// Date: 20-05-2024

// Name: Waleed Amjad

// Registration No: 2023-BS-AI-054

/\*Design a class hierarchy for an Employee management system. The base class should be Employee with

derived classes SalariedEmployee and CommissionEmployee. Each class should have appropriate data

members and member functions to handle the specific attributes and behaviors of each type of employee.

Employee: Should have data members for name, employee ID, and department. It should also have

member functions to get and set these values.

Salaried Employee: Inherits from Employee and adds a data member for annual Salary. It should have

member functions to get and set the salary, and to calculate the monthly pay.

Commission Employee: Inherits from Employee and adds data members for sales and commission Rate. It

should have member functions to get and set these values, and to calculate the total pay based on sales

and commission rate. \*/

#include <iostream>

#include <string>

using namespace std;

class Employee{

private:

string Name;

double ID;

string Department;

public:

//initialize

Employee(const string Name, double ID,const string Department): Name(Name) , ID(ID) , Department(Department){}

//method

string getName() const{

return Name;

}

double getID() const{

return ID;

}

string getDepartment() const{

return Department;

}

};

class SalariedEmployee: public Employee{

private:

double AnnualSalary;

public:

//constructor to initialize

SalariedEmployee(const string Name, double ID , const string Department, double AnnualSalary):

Employee(Name,ID,Department) , AnnualSalary(AnnualSalary){}

//Method to calculate the monthly salary

double MonthlySalary() const {

return AnnualSalary/12;

}

};

class CommissionedEmployee: public Employee{

private:

double sales,Commissionrate;

public:

//Constructor to initialize

CommissionedEmployee(const string Name, double ID , const string Department, double sales, double Commissionrate ):

Employee(Name,ID,Department) , sales(sales) , Commissionrate(Commissionrate){}

//Method to calculate total pay based on salaries and commmission

double TotalPay() const{

return sales\*Commissionrate;

}

};

int main(){

SalariedEmployee salariedemployee("Waleed",01,"AI",50000);

cout<<"Name of the Employee is: "<<salariedemployee.getName()<<endl;

cout<<"ID of the Employee is: "<<salariedemployee.getID()<<endl;

cout<<"The Department of the employee is: "<<salariedemployee.getDepartment()<<endl;

cout<<"The monthly pay of the employee is: "<<salariedemployee.MonthlySalary()<<endl;

cout<<endl;

CommissionedEmployee commissionedemployee("Talal",02,"CyberSecurity",2000,1.5);

cout<<"The name of the employee is: "<<commissionedemployee.getName()<<endl;

cout<<"The Id of the employee is: "<<commissionedemployee.getID()<<endl;

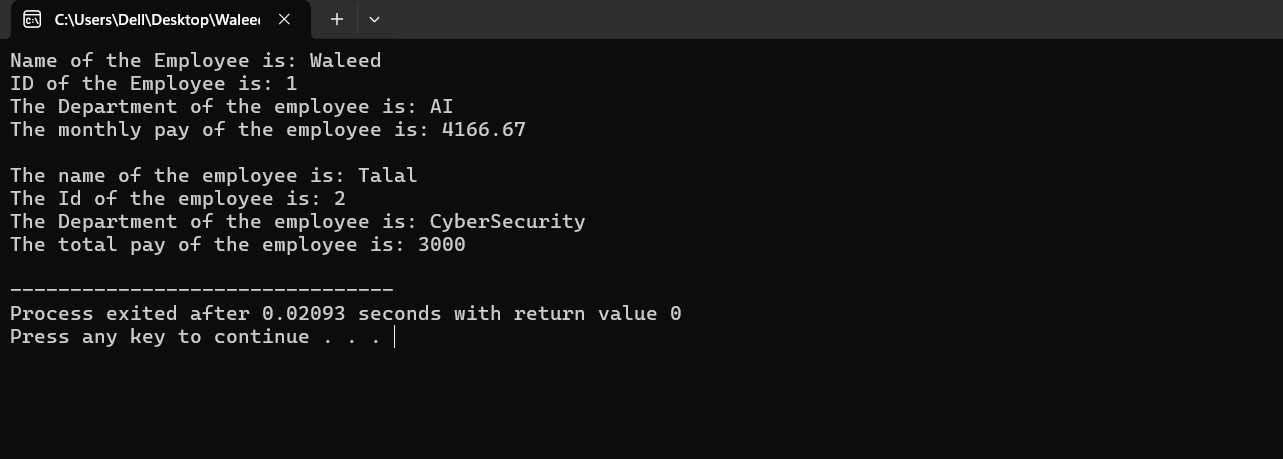
cout<<"The Department of the employee is: "<<commissionedemployee.getDepartment()<<endl;

cout<<"The total pay of the employee is: "<<commissionedemployee.TotalPay()<<endl;

return 0;

}

**Output:**



***End of Assignment***

**-------------------------------------------------------**