|  |
| --- |
| The university of faisalabad |
| Ayesha imran |
| 2023 BSAI 011 |
|  |
| **Monday** |
| **5/19/2024** |

|  |
| --- |
|  |

OOPS assignment no 4

# Program no 1:

Imagine a publishing company that markets both book and audiocasseƩe versions of its works. Create a class publicaƟon that stores the Ɵtle (a string) and price (type float) of a publicaƟon. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing Ɵme in minutes (type float). Each of these three classes should have a getdata () funcƟon to get its data from the user at the keyboard, and a putdata () funcƟon to display its data. Write a main () program to test the book and tape classes by creaƟng instances of them, asking the user to fill in data with getdata (), and then displaying the data with putdata (

# Code

#include <iostream>

using namespace std;

class Publication {

protected:

char title[50];

float price;

public:

void getdata() {

cout << "Enter title: ";

cin >> title; // Using cin to input title

cout << "Enter price: ";

cin >> price;

}

void putdata() const {

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

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

}

};

class Book : public Publication {

private:

int page Count;

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 the book:\n";

myBook.getdata();

cout << "\nEnter details for the tape:\n";

myTape.getdata();

cout << "\nDetails of the book:\n";

myBook.putdata();

cout << "\nDetails of the tape:\n";

myTape.putdata();

return 0;

}

# Program 2:

Imagine a publishing company that markets both book and audiocasseƩe versions of its works. Create a class publicaƟon that stores the Ɵtle (a string) and price (type float) of a publicaƟon. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing Ɵme in minutes (type float). Each of these three classes should have a getdata() funcƟon to get its data from the user at the keyboard, and a putdata() funcƟon to display its data. Write a main() program to test the book and tape classes by creaƟng instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata(

# Code:

#include <iostream>

using namespace std;

// Base class Publication

class Publication {

protected:

char title[50];

float price;

public:

void getdata() {

cout << "Enter title: ";

cin >> title;

cout << "Enter price: ";

cin >> price;

}

void putdata() const {

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

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

}

};

// Base class Sales

class Sales {

protected:

float sales[3];

public:

void getdata() {

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

cout << "Enter sales for month " << i + 1 << ": ";

cin >> sales[i];

}

}

void putdata() const {

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

cout << "Sales for month " << i + 1 << ": " << sales[i] << endl;

}

}

};

// Derived class Book from Publication and Sales

class Book : public Publication, public Sales {

private:

int page\_count;

public:

void getdata() {

Publication::getdata();

cout << "Enter page count: ";

cin >> page\_count;

Sales::getdata();

}

void putdata() const {

Publication::putdata();

cout << "Page Count: " << page\_count << endl;

Sales::putdata();

}

};

// Derived class Tape from Publication and Sales

class Tape : public Publication, public Sales {

private:

float playing\_time;

public:

void getdata() {

Publication::getdata();

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

cin >> playing\_time;

Sales::getdata();

}

void putdata() const {

Publication::putdata();

cout << "Playing Time (minutes): " << playing\_time << endl;

Sales::putdata();

}

};

// Main function to test Book and Tape classes

int main() {

Book book;

Tape tape;

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

book.getdata();

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

tape.getdata();

cout << "\nData for book:" << endl;

book.putdata();

cout << "\nData for tape:" << endl;

tape.putdata();

return 0;

}

# Program 3

Assume that the publisher in QuesƟon 1 and 3 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 publicaƟon. The disk class should incorporate the same member funcƟons 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

# Code:

#include <iostream>

using namespace std;

// Base class Publication

class Publication {

protected:

char title[50];

float price;

public:

void getdata() {

cout << "Enter title: ";

cin >> title;

cout << "Enter price: ";

cin >> price;

}

void putdata() const {

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

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

}

};

// Base class Sales

class Sales {

protected:

float sales[3];

public:

void getdata() {

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

cout << "Enter sales for month " << i + 1 << ": ";

cin >> sales[i];

}

}

void putdata() const {

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

cout << "Sales for month " << i + 1 << ": " << sales[i] << endl;

}

}

};

// Derived class Book from Publication and Sales

class Book : public Publication, public Sales {

private:

int page\_count;

public:

void getdata() {

Publication::getdata();

cout << "Enter page count: ";

cin >> page\_count;

Sales::getdata();

}

void putdata() const {

Publication::putdata();

cout << "Page Count: " << page\_count << endl;

Sales::putdata();

}

};

// Derived class Tape from Publication and Sales

class Tape : public Publication, public Sales {

private:

float playing\_time;

public:

void getdata() {

Publication::getdata();

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

cin >> playing\_time;

Sales::getdata();

}

void putdata() const {

Publication::putdata();

cout << "Playing Time (minutes): " << playing\_time << endl;

Sales::putdata();

}

};

// Derived class Disk from Publication and Sales

class Disk : public Publication, public Sales {

public:

enum DiskType { CD, DVD };

private:

DiskType disk\_type;

public:

void getdata() {

Publication::getdata();

char type;

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

cin >> type;

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

disk\_type = CD;

} else if (type == 'd' || type == 'D') {

disk\_type = DVD;

}

Sales::getdata();

}

void putdata() const {

Publication::putdata();

cout << "Disk Type: " << (disk\_type == CD ? "CD" : "DVD") << endl;

Sales::putdata();

}

};

// Main function to test Book, Tape, and Disk classes

int main() {

Book book;

Tape tape;

Disk disk;

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

book.getdata();

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

tape.getdata();

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

disk.getdata();

cout << "\nData for book:" << endl;

book.putdata();

cout << "\nData for tape:" << endl;

tape.putdata();

cout << "\nData for disk:" << endl;

disk.putdata();

return 0;

}

# Program 4

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 compensaƟon, 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, scienƟst, 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 compensaƟon and three new classes manager2, scienƟst2, and laborer2, and use mulƟple inheritance to derive these three classes from the original manager, scienƟst, and laborer classes and from compensaƟon. This way none of the original classes needs to be modified

# Code

#include <iostream>

using namespace std;

// Base class Employee

class Employee {

protected:

int empID;

public:

Employee() {

empID=0; }

void setEmpID(int id)

{

empID = id;

}

int getEmpID() const

{

return empID;

}

virtual void display() const

{

cout << "Employee ID: " << empID << endl;

}

};

// Derived class Employee2

class Employee2 : public Employee {

public:

enum Period { HOURLY, WEEKLY, MONTHLY };

private:

double compensation;

Period payPeriod;

public:

Employee2()

{

compensation=0.0;

payPeriod=HOURLY;

}

void setCompensation(double comp)

{

compensation = comp;

}

double getCompensation() const

{

return compensation;

}

void setPayPeriod(Period period)

{

payPeriod = period;

}

Period getPayPeriod() const

{

return payPeriod;

}

void display() const override {

Employee::display();

cout << "Compensation: " << compensation << endl;

cout << "Pay Period: " << (payPeriod == HOURLY ? "Hourly" : payPeriod == WEEKLY ? "Weekly" : "Monthly") << endl;

}

};

// Derived classes from Employee2

class Manager : public Employee2 {

public:

void display() const override {

cout << "Manager" << endl;

Employee2::display();

}

};

class Scientist : public Employee2 {

public:

void display() const override {

cout << "Scientist" << endl;

Employee2::display();

}

};

class Laborer : public Employee2 {

public:

void display() const override {

cout << "Laborer" << endl;

Employee2::display();

}

};

// Main function to demonstrate the functionality

int main() {

Manager mgr;

mgr.setEmpID(1);

mgr.setCompensation(5000.0);

mgr.setPayPeriod(Employee2::MONTHLY);

Scientist sci;

sci.setEmpID(2);

sci.setCompensation(4000.0);

sci.setPayPeriod(Employee2::WEEKLY);

Laborer lab;

lab.setEmpID(3);

lab.setCompensation(1000.0);

lab.setPayPeriod(Employee2::HOURLY);

mgr.display();

cout << endl;

sci.display();

cout << endl;

lab.display();

return 0;

}

# Program 5

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: AƩributes: color (type std::string). Methods: A constructor to iniƟalize the color and a method printColor to display the color. Circle Class: AƩributes: radius (type double). Methods: A constructor to iniƟalize 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: AƩributes: length and width (type double). Methods: A constructor to iniƟalize 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.

# Code

#include <iostream>

using namespace std;

class Shape {

protected:

char color[20]; // Fixed-size character array for simplicity

public:

// Constructor to initialize the color

Shape(const char\* c) {

int i = 0;

while (c[i] != '\0' && i < 19) {

color[i] = c[i];

i++;

}

color[i] = '\0'; // Null-terminate the character array

}

// Method to display the color

void printColor() const {

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

}

};

class Circle : public Shape {

private:

double radius;

public:

// Constructor to initialize the color and radius

Circle(const char\* c, double r) {

Shape = c;

radius=r;

}

// Method to calculate the area of the circle

double calculateArea() const {

return 3.14159265358979323846 \* radius \* radius;

}

// Method to display the area of the circle

void printArea() const {

cout << "Area of the Circle: " << calculateArea() << endl;

}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

// Constructor to initialize the color, length, and width

Rectangle(const char\* c, double l, double w) {

Shape= c;

length = l;

width = w; }

// Method to calculate the area of the rectangle

double calculateArea() const {

return length \* width;

}

// Method to display the area of the rectangle

void printArea() const {

cout << "Area of the Rectangle: " << calculateArea() << endl;

}

};

int main() {

Circle circle("Red", 5.0);

circle.printColor();

circle.printArea();

Rectangle rectangle("Blue", 4.0, 6.0);

rectangle.printColor();

rectangle.printArea();

return 0;

}

# Program 6

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 funcƟons to handle the specific AƩributes and behaviors of each type of employee. Employee: Should have data members for name, employee ID, and department. It should also have member funcƟons to get and set these values. Salaried Employee: Inherits from Employee and adds a data member for annual Salary. It should have member funcƟons 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 funcƟons to get and set these values, and to calculate the total pay based on sales and commission rate.

# Code

#include <iostream>

using namespace std;

class Employee {

private:

int employeeID;

string name;

string department;

public:

Employee(int id, string n, string dept) : employeeID(id), name(n), department(dept) {}

void display() const {

cout << "Employee ID: " << employeeID << endl;

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

cout << "Department: " << department << endl;

}

};

class SalariedEmployee : public Employee {

private:

double annualSalary;

public:

SalariedEmployee(int id, string n, string dept, double salary)

: Employee(id, n, dept), annualSalary(salary) {}

double calculateMonthlyPay() const {

return annualSalary / 12.0;

}

void displaySalary() const {

cout << "Annual Salary: $" << annualSalary << endl;

}

};

class CommissionEmployee : public Employee {

private:

double sales;

double commissionRate;

public:

CommissionEmployee(int id, string n, string dept, double salesAmt, double rate)

: Employee(id, n, dept), sales(salesAmt), commissionRate(rate) {}

double calculateTotalPay() const {

return sales \* commissionRate;

}

void displayCommissionInfo() const {

cout << "Total Sales: $" << sales << endl;

cout << "Commission Rate: " << commissionRate << endl;

}

};

int main() {

// Example usage

SalariedEmployee salariedEmp(101, "Ayesha Imran", "Marketing", 20000.0);

salariedEmp.display();

salariedEmp.displaySalary();

cout << "Monthly Pay: $" << salariedEmp.calculateMonthlyPay() << endl;

CommissionEmployee commissionEmp(102, "Amna Imran", "Sales", 700000.0, 0.05);

commissionEmp.display();

commissionEmp.displayCommissionInfo();

cout << "Total Pay: $" << commissionEmp.calculateTotalPay() << endl;

return 0;

}