2024

**ABEER AWAIS [022]**

**OOP( OBJECT-ORIENTED-PROGRAMMING)**

**5/18/2024**

ASSIGNMENT NO.5



**PROGRAM 1**

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0) {}

virtual void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const {

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

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

}

};

class Book : public Publication {

private:

int pageCount;

public:

Book() : Publication(), pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

}

void putdata() const override {

Publication::putdata();

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

}

};

class Tape : public Publication {

private:

float playingTime;

public:

Tape() : Publication(), playingTime(0.0) {}

void getdata() override {

Publication::getdata();

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

cin >> playingTime;

cin.ignore();

}

void putdata() const override {

Publication::putdata();

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

}

};

int main() {

Book myBook;

Tape myTape;

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

myBook.getdata();

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

myTape.getdata();

cout << "\nDisplaying book data:" << endl;

myBook.putdata();

cout << "\nDisplaying tape data:" << endl;

myTape.putdata();

return 0;

}

**PROGRAM 2**

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0) {}

virtual void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const {

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

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

}

};

class Sales {

protected:

float sales[3];

public:

Sales() {

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

sales[i] = 0.0;

}

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;

}

}

};

class Book : public Publication, public Sales {

private:

int pageCount;

public:

Book() : Publication(), Sales(), pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

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

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

private:

float playingTime;

public:

Tape() : Publication(), Sales(), playingTime(0.0) {}

void getdata() override {

Publication::getdata();

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

cin >> playingTime;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

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

Sales::putdata();

}

};

int main() {

Book myBook;

Tape myTape;

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

myBook.getdata();

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

myTape.getdata();

cout << "\nDisplaying book data:" << endl;

myBook.putdata();

cout << "\nDisplaying tape data:" << endl;

myTape.putdata();

return 0;

}

**PROGRAM 3**

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0) {}

virtual void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const {

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

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

}

};

class Sales {

protected:

float sales[3];

public:

Sales() {

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

sales[i] = 0.0;

}

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;

}

}

};

class Book : public Publication, public Sales {

private:

int pageCount;

public:

Book() : Publication(), Sales(), pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

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

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

private:

float playingTime;

public:

Tape() : Publication(), Sales(), playingTime(0.0) {}

void getdata() override {

Publication::getdata();

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

cin >> playingTime;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

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

Sales::putdata();

}

};

enum DiskType { CD, DVD };

class Disk : public Publication, public Sales {

private:

DiskType diskType;

public:

Disk() : Publication(), Sales(), diskType(CD) {}

void getdata() override {

Publication::getdata();

char type;

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

cin >> type;

cin.ignore();

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

diskType = CD;

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

diskType = DVD;

} else {

cout << "Invalid input. Defaulting to CD." << endl;

diskType = CD;

}

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

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

Sales::putdata();

}

};

int main() {

Book myBook;

Tape myTape;

Disk myDisk;

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

myBook.getdata();

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

myTape.getdata();

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

myDisk.getdata();

cout << "\nDisplaying book data:" << endl;

myBook.putdata();

cout << "\nDisplaying tape data:" << endl;

myTape.putdata();

cout << "\nDisplaying disk data:" << endl;

myDisk.putdata();

return 0;

}

**PROGRAM 4**

#include <iostream>

#include <string>

using namespace std;

// Assuming the original employee class is defined as follows:

class employee {

protected:

string name;

unsigned long number;

public:

void setData(const string& n, unsigned long num) {

name = n;

number = num;

}

void showData() const {

cout << "Name: " << name << "\nNumber: " << number << endl;

}

};

// Define the employee2 class

class employee2 : public employee {

public:

enum period { HOURLY, WEEKLY, MONTHLY };

private:

double compensation;

period payPeriod;

public:

void setCompensation(double comp, period per) {

compensation = comp;

payPeriod = per;

}

void showCompensation() const {

cout << "Compensation: " << compensation << "\nPay Period: ";

switch (payPeriod) {

case HOURLY: cout << "Hourly"; break;

case WEEKLY: cout << "Weekly"; break;

case MONTHLY: cout << "Monthly"; break;

}

cout << endl;

}

};

// Define the manager class

class manager : public employee2 {

private:

string title;

public:

void setTitle(const string& t) {

title = t;

}

void showTitle() const {

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

}

};

// Define the scientist class

class scientist : public employee2 {

private:

int publications;

public:

void setPublications(int pub) {

publications = pub;

}

void showPublications() const {

cout << "Publications: " << publications <<endl;

}

};

// Define the laborer class

class laborer : public employee2 {

// Additional attributes specific to laborer can be added here

};

int main() {

manager m;

m.setData("Abeer", 12345);

m.setCompensation(75000, employee2::MONTHLY);

m.setTitle("Project Manager");

scientist s;

s.setData("Mishal", 23456);

s.setCompensation(50000, employee2::MONTHLY);

s.setPublications(15);

laborer l;

l.setData("Aiman", 34567);

l.setCompensation(20, employee2::HOURLY);

// Display data

cout << "Manager Details:\n";

m.showData();

m.showCompensation();

m.showTitle();

cout << "\nScientist Details:\n";

s.showData();

s.showCompensation();

s.showPublications();

cout << "\nLaborer Details:\n";

l.showData();

l.showCompensation();

return 0;

}

**PROGRAM 5**

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

// Base class: Shape

class Shape {

protected:

string color;

public:

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

void printColor() const {

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

}

};

// Derived class: Circle

class Circle : public Shape {

private:

double radius;

public:

Circle(const string& col, double rad) : Shape(col), radius(rad) {}

double calculateArea() const {

return M\_PI \* radius \* radius;

}

void printArea() const {

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

}

};

// Derived class: Rectangle

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(const string& col, double len, double wid) : Shape(col), length(len), width(wid) {}

double calculateArea() const {

return length \* width;

}

void printArea() const {

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

}

};

// Main function to test the implementation

int main() {

Circle circle("Black", 10.0);

Rectangle rectangle("Grey", 6.0, 9.0);

cout << "Circle Details:\n";

circle.printColor();

circle.printArea();

cout << "\nRectangle Details:\n";

rectangle.printColor();

rectangle.printArea();

return 0; }

**PROGRAM 6**

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

string name;

int employeeID;

string departmen

public:

Employee(string name, int employeeID, string department) {

this->name = name; // “This-> is used to point towards the data type used earlier”

this->employeeID = employeeID;

this->department = department;

}

string getName() {

return name;

}

void setName(string name) {

this->name = name;

}

int getEmployeeID() {

return employeeID;

}

void setEmployeeID(int employeeID) {

this->employeeID = employeeID;

}

string getDepartment() {

return department;

}

void setDepartment(string department) {

this->department = department;

}

};

class SalariedEmployee : public Employee {

private:

double annualSalary

public:

SalariedEmployee(string name, int employeeID, string department, double annualSalary)

: Employee(name, employeeID, department) {

this->annualSalary = annualSalary;

}

double getAnnualSalary() {

return annualSalary;

}

void setAnnualSalary(double annualSalary) {

this->annualSalary = annualSalary;

}

double calculateMonthlyPay() {

return annualSalary / 12;

}

};

class CommissionEmployee : public Employee {

private:

double sales;

double commissionRate;

public:

CommissionEmployee(string name, int employeeID, string department, double sales, double commissionRate)

: Employee(name, employeeID, department) {

this->sales = sales;

this->commissionRate = commissionRate;

}

double getSales() {

return sales;

}

void setSales(double sales) {

this->sales = sales;

}

double getCommissionRate() {

return commissionRate;

}

void setCommissionRate(double commissionRate) {

this->commissionRate = commissionRate;

}

double calculateTotalPay() {

return sales \* commissionRate;

}

};

int main() {

SalariedEmployee salariedEmp("Aiman Awais", 62351, "Sales", 50000.0);

CommissionEmployee commissionEmp("Abeer Awais", 58659, "Marketing", 30000.0, 0.05);

cout << "Salaried Employee:" << endl;

cout << "Name: " << salariedEmp.getName() << endl;

cout << "Employee ID: " << salariedEmp.getEmployeeID() << endl;

cout << "Department: " << salariedEmp.getDepartment() << endl;

cout << "Annual Salary: $" << salariedEmp.getAnnualSalary() << endl;

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

cout << endl;

cout << "Commission Employee:" << endl;

cout << "Name: " << commissionEmp.getName() << endl;

cout << "Employee ID: " << commissionEmp.getEmployeeID() << endl;

cout << "Department: " << commissionEmp.getDepartment() << endl;

cout << "Sales: $" << commissionEmp.getSales() << endl;

cout << "Commission Rate: " << commissionEmp.getCommissionRate() << endl;

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

return 0;

}