

The University of Faisalabad

SUBMITTED BY : M.Wahaaj Siddique

SUBMITTED TO : M.Javed

REG # : BS-AI-007

DEPARTMENT : CS

SUBJECT : OOP

Problem 1: Publishing Company Class Hierarchy

Statement:

Create a class Publication that stores the title (a string) and price (type float) of a publication. Derive two classes from Publication: Book, which adds a page count (type int), and Tape, which adds a playing time in minutes (type float). Each class should have getData() and putData() functions to get data from the user and display it, respectively. Write a main program to test the Book and Tape classes by creating instances, filling in data, and displaying it.

SYNTAX:

// File: P1.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-BS-AI-007 // Create a class Publication that stores the title (a string) and price (type float) of a publication. Derive two classes from Publication: Book, which adds a page count (type int), and Tape, which adds a playing time in minutes (type float). Each class should have getData() and putData() functions to get data from the user and display it, respectively. Write a main program to test the Book and Tape classes by creating instances, filling in data, and displaying it.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Publication {
```

```
protected:
```

```
    string title;
```

```
    float price;
```

```
public:
```

```
    void getData() {
```

```
        cout << "Enter title: ";
```

```
        cin.ignore(); // To clear the input buffer
```

```
        getline(cin, title);
```

```
        cout << "Enter price: ";
```

```
        cin >> price;
```

```
    }
```

```
    void putData() const {
```

```
        cout << "Title: " << title << "\nPrice: " << 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 data for book:\n";  
    myBook.getData();  
    cout << "\nEnter data for tape:\n";  
    myTape.getData();  
  
    cout << "\nBook data:\n";  
    myBook.putData();  
    cout << "\nTape data:\n";  
    myTape.putData();  
  
    return 0;  
}
```

OUTPUT:

```
D:\OOP\Assignment no 3\pp1.exe
Enter data for book:
Enter title: English
Enter price: 234
Enter page count: 233

Enter data for tape:
Enter title: Singham
Enter price: 110
Enter playing time (in minutes): 120

Book data:
Title: nglish
Price: 234
Page Count: 233

Tape data:
Title: Singham
Price: 110
Playing Time: 120 minutes

-----
Process exited after 30.38 seconds with return value 0
Press any key to continue . . .
```

Problem 2: Adding Sales Data to Publications

Statement:

Start with the Publication, Book, and Tape classes from Problem 1. Add a base class Sales that holds an array of three floats to record the dollar sales of a particular publication for the last three months. Include `getData()` and `putData()` functions to get sales amounts from the user and display them. 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 function to create a Book object and a Tape object and exercise their input/output capabilities.

SYNTAX:

```
// File: P2.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-BS-AI-007 // Start with the Publication, Book, and Tape classes from Problem 1. Add a base class Sales that holds an array of three floats to record the dollar sales of a particular publication for the last three months. Include getData() and putData() functions to get sales amounts from the user and display them. 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 function 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 clear the input buffer
        getline(cin, title);
        cout << "Enter price: ";
        cin >> price;
    }
    void putData() const {
        cout << "Title: " << title << "\nPrice: " << price << endl;
    }
};
```

```
class Sales {
private:
    float sales[3];
public:
    void getData() {
        cout << "Enter sales for last three months:\n";
        for(int i = 0; i < 3; i++) {
            cout << "Month " << i+1 << ": ";
            cin >> sales[i];
        }
    }
};
```

```

    }
    void putData() const {
        cout << "Sales for last three months:\n";
        for(int i = 0; i < 3; i++) {
            cout << "Month " << i+1 << ": " << sales[i] << endl;
        }
    }
};

```

```

class Book : public Publication, public Sales {
private:
    int pageCount;
public:
    void getData() {
        Publication::getData();
        Sales::getData();
        cout << "Enter page count: ";
        cin >> pageCount;
    }
    void putData() const {
        Publication::putData();
        Sales::putData();
        cout << "Page Count: " << pageCount << endl;
    }
};

```

```

class Tape : public Publication, public Sales {
private:
    float playingTime;
public:

```

```

void getData() {
    Publication::getData();
    Sales::getData();
    cout << "Enter playing time (in minutes): ";
    cin >> playingTime;
}

void putData() const {
    Publication::putData();
    Sales::putData();
    cout << "Playing Time: " << playingTime << " minutes" << endl;
}
};

```

```

int main() {
    Book myBook;
    Tape myTape;

    cout << "Enter data for book:\n";
    myBook.getData();
    cout << "\nEnter data for tape:\n";
    myTape.getData();

    cout << "\nBook data:\n";
    myBook.putData();
    cout << "\nTape data:\n";
    myTape.putData();

    return 0;
}

```

OUTPUT:


```

D:\OOP\Assignment no 3\P1.exe
Enter data for tape:
Enter title: Singham
Enter price: 120
Enter sales for last three months:
Month 1: 49
Month 2: 66
Month 3: 54
Enter playing time (in minutes): 120

Book data:
Title: nglsh
Price: 234
Sales for last three months:
Month 1: 44
Month 2: 33
Month 3: 55
Page Count: 234

Tape data:
Title: Singham
Price: 120
Sales for last three months:
Month 1: 49
Month 2: 66
Month 3: 54
Playing Time: 120 minutes

-----
Process exited after 54.81 seconds with return value 0
Press any key to continue . . .

```

Problem 3: Adding Disk Distribution to Publications

Statement:

Extend the Publication class from Problems 1 and 2 to add a third way to distribute books: on computer disk. 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 unique data item for this class is the disk type: either CD or DVD. Use an enum type to store this item. The user should select the type by typing 'c' for CD or 'd' for DVD.

SUNTAX:

// File: P4.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-BS-AI-007 // Extend the Publication class from Problems 1 and 2 to add a third way to distribute books: on computer disk. 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 unique data item for this class is the disk type: either CD or DVD. Use an enum type to store this item. The user should select the type by typing 'c' for CD or 'd' for DVD.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
enum DiskType { CD, DVD };
```

```
class Publication {  
protected:  
    string title;  
    float price;  
public:  
    void getData() {  
        cout << "Enter title: ";  
        cin.ignore(); // To clear the input buffer  
        getline(cin, title);  
        cout << "Enter price: ";  
        cin >> price;  
    }  
    void putData() const {  
        cout << "Title: " << title << "\nPrice: " << price << endl;  
    }  
};
```

```
class Sales {  
private:  
    float sales[3];  
public:  
    void getData() {  
        cout << "Enter sales for the last three months:\n";  
        for(int i = 0; i < 3; i++) {  
            cout << "Month " << i+1 << ": ";  
            cin >> sales[i];  
        }  
    }  
};
```

```

void putData() const {
    cout << "Sales for the last three months:\n";
    for(int i = 0; i < 3; i++) {
        cout << "Month " << i+1 << ": " << sales[i] << endl;
    }
}
};

```

```

class Book : public Publication, public Sales {
private:
    int pageCount;
public:
    void getData() {
        Publication::getData();
        Sales::getData();
        cout << "Enter page count: ";
        cin >> pageCount;
    }
    void putData() const {
        Publication::putData();
        Sales::putData();
        cout << "Page Count: " << pageCount << endl;
    }
};

```

```

class Tape : public Publication, public Sales {
private:
    float playingTime;
public:
    void getData() {

```

```

        Publication::getData();
        Sales::getData();
        cout << "Enter playing time (in minutes): ";
        cin >> playingTime;
    }
    void putData() const {
        Publication::putData();
        Sales::putData();
        cout << "Playing Time: " << playingTime << " minutes" << endl;
    }
};

class Disk : public Publication {
private:
    DiskType diskType;
public:
    void getData() {
        Publication::getData();
        char type;
        cout << "Enter disk type (c for CD, d for DVD): ";
        cin >> type;
        diskType = (type == 'c') ? CD : DVD;
    }
    void putData() const {
        Publication::putData();
        cout << "Disk Type: " << (diskType == CD ? "CD" : "DVD") << endl;
    }
};

int main() {

```

```
Book myBook;
Tape myTape;
Disk myDisk;

cout << "Enter data for book:\n";
myBook.getData();
cout << "\nEnter data for tape:\n";
myTape.getData();
cout << "\nEnter data for disk:\n";
myDisk.getData();

cout << "\nBook data:\n";
myBook.putData();
cout << "\nTape data:\n";
myTape.putData();
cout << "\nDisk data:\n";
myDisk.putData();

return 0;
}
```

OUTPUT:

```
D:\OOP\Assignment no 3\p3.exe
Enter data for book:
Enter title: English
Enter price: 350
Enter sales for the last three months:
Month 1: 222
Month 2: 333
Month 3: 234
Enter page count: 267

Enter data for tape:
Enter title: Singham
Enter price: 150
Enter sales for the last three months:
Month 1: 440
Month 2: 543
Month 3: 345
Enter playing time (in minutes): 120

Enter data for disk:
Enter title: ALL In One
Enter price: 230
Enter disk type (c for CD, d for DVD): c

Book data:
Title: nglish
Price: 350
Sales for the last three months:
Month 1: 222
Month 2: 333
Month 3: 234
Page Count: 267

Tape data:
Title: Singham
Price: 150
Sales for the last three months:
Month 1: 440
Month 2: 543
Month 3: 345
Playing Time: 120 minutes

Disk data:
Title: ALL In One
Price: 230
Disk Type: CD

-----
Process exited after 55.89 seconds with return value 0
Press any key to continue . . .
```

Problem 4: Employee Compensation and Payment Periods

Statement:

Derive a class called Employee2 from the Employee class. This new class should add a double data item called compensation and an enum type called Period to indicate whether the employee is paid hourly, weekly, or monthly. Modify the Manager, Scientist, and Laborer classes so they are derived from Employee2 instead of Employee. Write a main function to create instances of Manager, Scientist, and Laborer, get their data, and display it.

SYNTAX:

// File: P4.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-BS-AI-007 // Derive a class called Employee2 from the Employee class. This new class should add a double data item called compensation and an enum type called Period to indicate whether the employee is paid hourly, weekly, or monthly. Modify the Manager, Scientist, and Laborer classes so they are derived from Employee2 instead of Employee. Write a main function to create instances of Manager, Scientist, and Laborer, get their data, and display it.

```
#include <iostream>
```

```

#include <string>

using namespace std;

enum Period { HOURLY, WEEKLY, MONTHLY };

class Employee {
protected:
    string name;
    int number;
public:
    virtual void getData() {
        cout << "Enter employee name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter employee number: ";
        cin >> number;
    }
    virtual void putData() const {
        cout << "Name: " << name << "\nNumber: " << number << endl;
    }
};

class Employee2 : public Employee {
protected:
    double compensation;
    Period period;
public:
    void getData() override {
        Employee::getData();
        cout << "Enter compensation: ";
    }
};

```

```

    cin >> compensation;

    char per;

    cout << "Enter period (h for hourly, w for weekly, m for monthly): ";
    cin >> per;

    switch (per) {
        case 'h': period = HOURLY; break;
        case 'w': period = WEEKLY; break;
        case 'm': period = MONTHLY; break;
        default: period = HOURLY; // Default case to handle invalid input
    }
}

void putData() const override {
    Employee::putData();
    cout << "Compensation: " << compensation << "\nPeriod: "
        << (period == HOURLY ? "Hourly" : period == WEEKLY ? "Weekly" : "Monthly")
<< endl;
}
};

```

```

class Manager : public Employee2 {
    // No need to override getData and putData since they don't add new functionality
};

```

```

class Scientist : public Employee2 {
    // No need to override getData and putData since they don't add new functionality
};

```

```

class Laborer : public Employee2 {
    // The previous code was incomplete, adding the full method
public:
    void getData() override {

```



```

        Employee2::getData();
    }
    void putData() const override {
        Employee2::putData();
    }
};

int main() {
    Manager manager;
    Scientist scientist;
    Laborer laborer;

    cout << "Enter details for manager:\n";
    manager.getData();
    cout << "Enter details for scientist:\n";
    scientist.getData();
    cout << "Enter details for laborer:\n";
    laborer.getData();

    cout << "\nManager details:\n";
    manager.putData();
    cout << "\nScientist details:\n";
    scientist.putData();
    cout << "\nLaborer details:\n";
    laborer.putData();

    return 0;
}

```

OUTPUT:

```

D:\OOP\Assignment no 3\P4.exe
Enter details for manager:
Enter employee name: Wahaaj
Enter employee number: 2223432
Enter compensation: 2300
Enter period (h for hourly, w for weekly, m for monthly): h
Enter details for scientist:
Enter employee name: Faisal
Enter employee number: 3563242
Enter compensation: 4500
Enter period (h for hourly, w for weekly, m for monthly): w
Enter details for laborer:
Enter employee name: Haseeb
Enter employee number: 3454244
Enter compensation: 5600
Enter period (h for hourly, w for weekly, m for monthly): m

Manager details:
Name: ahaaj
Number: 2223432
Compensation: 2300
Period: Hourly

Scientist details:
Name: Faisal
Number: 3563242
Compensation: 4500
Period: Weekly

Laborer details:
Name: Haseeb
Number: 3454244
Compensation: 5600
Period: Monthly

-----
Process exited after 43.42 seconds with return value 0
Press any key to continue . . .

```

Problem 5: Shape Inheritance Hierarchy

Statement: 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: Attributes: color (type std::string). Methods: a constructor to initialize the color and a method **printColor** to display the color.
- **Circle** Class: Attributes: radius (type double). Methods: a constructor to initialize the color and radius, a method **calculateArea** to calculate the area of the circle (area = $\pi * \text{radius} * \text{radius}$), and a method **printArea** to display the area.
- **Rectangle** Class: Attributes: 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.

SYNTAX:

```

// File: P4.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-
BS-AI-007 // 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: Attributes: color (type std::string). Methods: a constructor to initialize the color and a method printColor to display the color.

Circle Class: Attributes: 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: Attributes: 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>
```

```
using namespace std;
```

```
class Shape {
```

```
protected:
```

```
    string color;
```

```
public:
```

```
    Shape(const string& c) : color(c) {}
```

```
    void printColor() const {
```

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

```
    }
```

```
};
```

```
class Circle : public Shape {
```

```
private:
```

```
    double radius;
```

```
public:
```

```
    Circle(const string& c, double r) : Shape(c), radius(r) {}
```

```
    double calculateArea() const {
```

```
        return 3.14159 * radius * radius;
```

```
    }
```

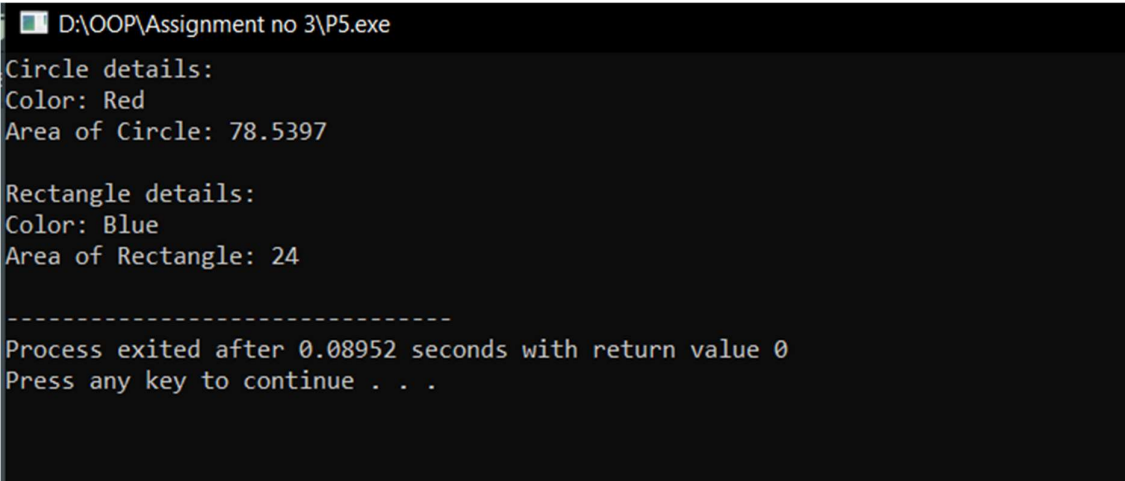
```
void printArea() const {  
    cout << "Area of Circle: " << calculateArea() << endl;  
}  
};
```

```
class Rectangle : public Shape {  
private:  
    double length, width;  
public:  
    Rectangle(const string& c, double l, double w) : Shape(c), length(l), width(w) {}  
    double calculateArea() const {  
        return length * width;  
    }  
    void printArea() const {  
        cout << "Area of Rectangle: " << calculateArea() << endl;  
    }  
};
```

```
int main() {  
    Circle myCircle("Red", 5.0);  
    Rectangle myRectangle("Blue", 4.0, 6.0);  
  
    cout << "Circle details:\n";  
    myCircle.printColor();  
    myCircle.printArea();  
  
    cout << "\nRectangle details:\n";  
    myRectangle.printColor();  
    myRectangle.printArea();  
}
```

```
    return 0;
}
```

OUTPUT:



```
D:\OOP\Assignment no 3\P5.exe
Circle details:
Color: Red
Area of Circle: 78.5397

Rectangle details:
Color: Blue
Area of Rectangle: 24

-----
Process exited after 0.08952 seconds with return value 0
Press any key to continue . . .
```

Problem 6: Employee Management System

Statement: 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.
- **SalariedEmployee:** 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.
- **CommissionEmployee:** 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.

SYNTAX:

```
// File: P6.cpp // Date: 22-05-2024 // Name: Muhammad Wahaaj // Registration No: 2023-BS-AI-007//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.

SalariedEmployee: 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.

CommissionEmployee: 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 {
```

```
protected:
```

```
    string name;
```

```
    int employeeID;
```

```
    string department;
```

```
public:
```

```
    void getData() {
```

```
        cout << "Enter name: ";
```

```
        cin.ignore();
```

```
        getline(cin, name);
```

```
        cout << "Enter employee ID: ";
```

```
        cin >> employeeID;
```

```
        cin.ignore();
```

```
        cout << "Enter department: ";
```

```
        getline(cin, department);
```

```
    }
```

```
    void putData() const {
```

```
        cout << "Name: " << name << "\nEmployee ID: " << employeeID << "\nDepartment: " << department << endl;
```

```
    }
```

```
};
```

```
class SalariedEmployee : public Employee {
```

```

private:
    double annualSalary;
public:
    void getData() {
        Employee::getData();
        cout << "Enter annual salary: ";
        cin >> annualSalary;
    }
    void putData() const {
        Employee::putData();
        cout << "Annual Salary: " << annualSalary << "\nMonthly Pay: " <<
calculateMonthlyPay() << endl;
    }
    double calculateMonthlyPay() const {
        return annualSalary / 12;
    }
};

```

```

class CommissionEmployee : public Employee {
private:
    double sales;
    double commissionRate;
public:
    void getData() {
        Employee::getData();
        cout << "Enter sales: ";
        cin >> sales;
        cout << "Enter commission rate: ";
        cin >> commissionRate;
    }
    void putData() const {

```

```

        Employee::putData();

        cout << "Sales: " << sales << "\nCommission Rate: " << commissionRate << "\nTotal
Pay: " << calculateTotalPay() << endl;
    }
    double calculateTotalPay() const {
        return sales * commissionRate;
    }
};

```

```

int main() {
    SalariedEmployee mySalariedEmployee;
    CommissionEmployee myCommissionEmployee;

    cout << "Enter data for salaried employee:\n";
    mySalariedEmployee.getData();
    cout << "\nEnter data for commission employee:\n";
    myCommissionEmployee.getData();

    cout << "\nSalaried Employee data:\n";
    mySalariedEmployee.putData();
    cout << "\nCommission Employee data:\n";
    myCommissionEmployee.putData();

    return 0;
}

```

OUTPUT:

D:\OOP\Assignment no 3\P6.exe

Enter data for salaried employee:

Enter name: Wahaaaj

Enter employee ID: 34245

Enter department: CS

Enter annual salary: 234500

Enter data for commission employee:

Enter name: Faisal

Enter employee ID: 32425

Enter department: CS

Enter sales: 34567

Enter commission rate: 10

Salaried Employee data:

Name: ahaaj

Employee ID: 34245

Department: CS

Annual Salary: 234500

Monthly Pay: 19541.7

Commission Employee data:

Name: Faisal

Employee ID: 32425

Department: CS

Sales: 34567

Commission Rate: 10

Total Pay: 345670

Process exited after 36.05 seconds with return value 0

Press any key to continue . . .