**Assingment No 3:**

**Roll No 056.**

**Masood Bakhtiar.**

**Qno 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:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

#include <string>

using namespace std;

class Publication {

public:

string title;

float price;

void getdata() {

cout << "Enter the title of the publication: ";

cin >> title;

cout << "Enter the price of the publication: ";

cin >> price;

}

void putdata() {

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

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

}

};

class Book : public Publication {

public:

int page\_count;

void getdata() {

Publication::getdata();

cout << "Enter the page count of the book: ";

cin >> page\_count;

}

void putdata() {

Publication::putdata();

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

}

};

class Tape : public Publication {

public:

float playing\_time;

void getdata() {

Publication::getdata();

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

cin >> playing\_time;

}

void putdata() {

Publication::putdata();

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

}

};

int main() {

Book book;

Tape tape;

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

book.getdata();

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

tape.getdata();

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

book.putdata();

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

tape.putdata();

return 0;

}

**Qno 2:**

**Start with the publicaƟon, book, and tape classes of QuesƟon 1. Add a base class sales that holds an array of three floats so that it can record the dollar sales of a parƟcular publicaƟon for the last three months. Include a getdata() funcƟon to get three sales amounts from the user, and a putdata() funcƟon to display the sales figures. Alter the book and tape classes so they are derived from both publicaƟon and sales. An object of class book or tape should input and output sales data along with its other data. Write a main() funcƟon to create a book object and a tape object and exercise their input/output capabiliƟes.**

**Code:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

void getdata() {

cout << "Enter the title of the publication: ";

cin >> title;

cout << "Enter the price of the publication: ";

cin >> price;

}

void putdata() {

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

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

}

};

class Sales {

protected:

float sales[3];

public:

void getdata() {

cout << "Enter the sales figures for the last three months:" << endl;

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

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

cin >> sales[i];

}

}

void putdata() {

cout << "Sales figures for the last three months:" << endl;

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

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

}

}

};

class Book : public Publication, public Sales {

protected:

int page\_count;

public:

void getdata() {

Publication::getdata();

cout << "Enter the page count of the book: ";

cin >> page\_count;

Sales::getdata();

}

void putdata() {

Publication::putdata();

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

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

protected:

float playing\_time;

public:

void getdata() {

Publication::getdata();

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

cin >> playing\_time;

Sales::getdata();

}

void putdata() {

Publication::putdata();

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

Sales::putdata();

}

};

int main() {

Book book;

Tape tape;

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

book.getdata();

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

tape.getdata();

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

book.putdata();

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

tape.putdata();

return 0;

}

**Q no 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:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

void getdata() {

cout << "Enter the title of the publication: ";

cin >> title;

cout << "Enter the price of the publication: ";

cin >> price;

}

void putdata() {

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

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

}

};

class Sales {

protected:

array<float, 3> sales;

public:

void getdata() {

cout << "Enter the sales figures for the last three months:" << endl;

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

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

cin >> sales[i];

}

}

void putdata() {

cout << "Sales figures for the last three months:" << endl;

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

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

}

}

};

class Book : public Publication, public Sales {

protected:

int page\_count;

public:

void getdata() {

Publication::getdata();

cout << "Enter the page count of the book: ";

cin >> page\_count;

Sales::getdata();

}

void putdata() {

Publication::putdata();

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

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

protected:

float playing\_time;

public:

void getdata() {

Publication::getdata();

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

cin >> playing\_time;

Sales::getdata();

}

void putdata() {

Publication::putdata();

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

Sales::putdata();

}

};

enum class DiskType { CD, DVD };

class Disk : public Publication, public Sales {

protected:

DiskType disk\_type;

public:

void getdata() {

Publication::getdata();

char disk\_type\_char;

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

cin >> disk\_type\_char;

if (disk\_type\_char == 'c') {

disk\_type = DiskType::CD;

} else {

disk\_type = DiskType::DVD;

}

Sales::getdata();

}

void putdata() {

Publication::putdata();

cout << "Disk Type: ";

if (disk\_type == DiskType::CD) {

cout << "CD";

} else {

cout << "DVD";

}

cout << endl;

Sales::putdata();

}

};

int main() {

Book book;

Tape tape;

Disk disk;

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

book.getdata();

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

tape.getdata();

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

disk.getdata();

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

book.putdata();

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

tape.putdata();

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

disk.putdata();

return 0;

}

**Q no 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:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string name;

int number;

public:

void getdata() {

cout << "Enter name: ";

cin >> name;

cout << "Enter number: ";

cin >> number;

}

void putdata() {

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

cout << "Number: " << number << endl;

}

};

enum class PayPeriod { HOURLY, WEEKLY, MONTHLY };

class Employee2 : public Employee {

protected:

double compensation;

PayPeriod pay\_period;

public:

void getdata() {

Employee::getdata();

cout << "Enter compensation: ";

cin >> compensation;

int period;

cout << "Enter pay period (0 for hourly, 1 for weekly, 2 for monthly): ";

cin >> period;

pay\_period = static\_cast<PayPeriod>(period);

}

void putdata() {

Employee::putdata();

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

cout << "Pay Period: ";

switch (pay\_period) {

case PayPeriod::HOURLY:

cout << "Hourly";

break;

case PayPeriod::WEEKLY:

cout << "Weekly";

break;

case PayPeriod::MONTHLY:

cout << "Monthly";

break;

}

cout << endl;

}

};

class Manager2 : public Employee2 {

protected:

string title;

public:

void getdata() {

Employee2::getdata();

cout << "Enter title: ";

cin >> title;

}

void putdata() {

Employee2::putdata();

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

}

};

class Scientist2 : public Employee2 {

protected:

int publications;

public:

void getdata() {

Employee2::getdata();

cout << "Enter number of publications: ";

cin >> publications;

}

void putdata() {

Employee2::putdata();

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

}

};

class Laborer2 : public Employee2 {

protected:

string job;

public:

void getdata() {

Employee2::getdata();

cout << "Enter job: ";

cin >> job;

}

void putdata() {

Employee2::putdata();

cout << "Job: " << job << endl;

}

};

int main() {

Manager2 manager;

Scientist2 scientist;

Laborer2 laborer;

cout << "Enter manager data:" << endl;

manager.getdata();

cout << "\nEnter scientist data:" << endl;

scientist.getdata();

cout << "\nEnter laborer data:" << endl;

laborer.getdata();

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

manager.putdata();

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

scientist.putdata();

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

laborer.putdata();

return 0;

}

**Q no 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:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

#include <string>

using namespace std;

class Shape {

protected:

string color;

public:

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

void printColor() {

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

}

};

class Circle : public Shape {

private:

double radius;

public:

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

double calculateArea() {

return 3.14159 \* radius \* radius;

}

void printArea() {

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

}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(const string& col, double l, double w) : Shape(col), length(l), width(w) {}

double calculateArea() {

return length \* width;

}

void printArea() {

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

}

};

int main() {

Circle circle("Red", 5.0);

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

cout << "Circle Details:" << endl;

circle.printColor();

circle.printArea();

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

rectangle.printColor();

rectangle.printArea();

return 0;

}

**Q no 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:**

//Masood Bakhtiar

//Roll no 56

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string name;

int employeeID;

string department;

public:

Employee(const string& empName, int empID, const string& dept) : name(empName), employeeID(empID), department(dept) {}

void setName(const string& empName) {

name = empName;

}

void setEmployeeID(int empID) {

employeeID = empID;

}

void setDepartment(const string& dept) {

department = dept;

}

string getName() const {

return name;

}

int getEmployeeID() const {

return employeeID;

}

string getDepartment() const {

return department;

}

};

class SalariedEmployee : public Employee {

private:

double annualSalary;

public:

SalariedEmployee(const string& empName, int empID, const string& dept, double salary) : Employee(empName, empID, dept), annualSalary(salary) {}

void setAnnualSalary(double salary) {

annualSalary = salary;

}

double getAnnualSalary() const {

return annualSalary;

}

double calculateMonthlyPay() const {

return annualSalary / 12;

}

};

class CommissionEmployee : public Employee {

private:

double sales;

double commissionRate;

public:

CommissionEmployee(const string& empName, int empID, const string& dept, double salesAmt, double commission) : Employee(empName, empID, dept), sales(salesAmt), commissionRate(commission) {}

void setSales(double salesAmt) {

sales = salesAmt;

}

void setCommissionRate(double commission) {

commissionRate = commission;

}

double getSales() const {

return sales;

}

double getCommissionRate() const {

return commissionRate;

}

double calculateTotalPay() const {

return sales \* commissionRate;

}

};

int main() {

SalariedEmployee salariedEmployee("John Doe", 1001, "Finance", 60000.0);

CommissionEmployee commissionEmployee("Jane Smith", 1002, "Sales", 100000.0, 0.05);

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

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

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

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

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

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

cout << "\nCommission Employee Details:" << endl;

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

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

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

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

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

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

return 0;

}