**1.Illustrate the following in two ways of object notations.**

**a) Employee**

**b) Teacher**

**c) Car**

**Identify the object of the following class and create object.**

**a) Company**

**b) Class**

**c) Vehicle**

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

string name;

int employeeID;

double salary;

public:

Employee(string n, int id, double sal) : name(n), employeeID(id), salary(sal) {}

void displayInfo() {

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

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

cout << "Employee Salary: $" << salary << endl;

}

};

class Teacher {

private:

string name;

string subject;

public:

Teacher(string n, string sub) : name(n), subject(sub) {}

void displayInfo() {

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

cout << "Teaching Subject: " << subject << endl;

}

};

class Car {

private:

string make;

string model;

int year;

public:

Car(string mk, string mdl, int yr) : make(mk), model(mdl), year(yr) {}

void displayInfo() {

cout << "Car Make: " << make << endl;

cout << "Car Model: " << model << endl;

cout << "Car Year: " << year << endl;

}

};

class Company {

};

class Class{

};

class Vehicle {

};

int main() {

Employee emp("John Doe", 1001, 50000.0);

Teacher teacher("Alice Smith", "Mathematics");

Car car("Toyota", "Corolla", 2022);

cout << "Employee Information:" << endl;

emp.displayInfo();

cout << endl;

cout << "Teacher Information:" << endl;

teacher.displayInfo();

cout << endl;

cout << "Car Information:" << endl;

car.displayInfo();

cout << endl;

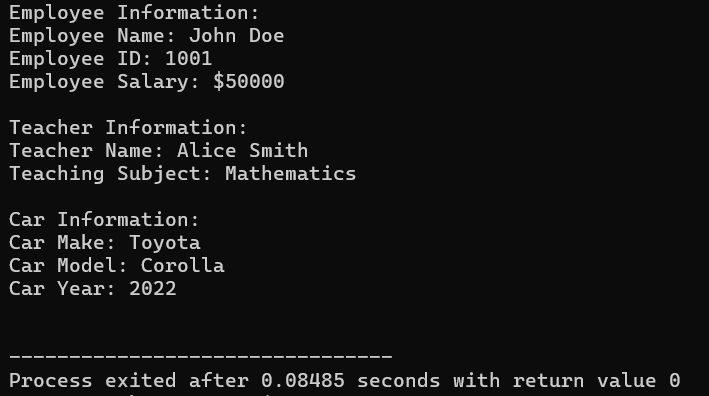
Company companyObj;

Class classObj;

Vehicle vehicleObj;

return 0;

}



**2.Design a class hierarchy to represent a geometric shapes system. Include classes for circles, rectangles, and triangles. Implement methods for calculating the area and perimeter of each shape. Demonstrate the use of abstract classes and pure virtual functions. Discuss how this design supports future additions of new shapes.**

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

double area() const { return 0; }

double perimeter() const { return 0; }

~Shape() {}

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double area() const {

return M\_PI \* radius \* radius;

}

double perimeter() const {

return 2 \* M\_PI \* radius;

}

};

class Rectangle : public Shape {

private:

double length, width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() const {

return length \* width;

}

double perimeter() const {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

double side1, side2, side3;

public:

Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

double area() const {

double s = (side1 + side2 + side3) / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() const {

return side1 + side2 + side3;

}

};

int main() {

Circle circle(5);

Rectangle rectangle(4, 6);

Triangle triangle(3, 4, 5);

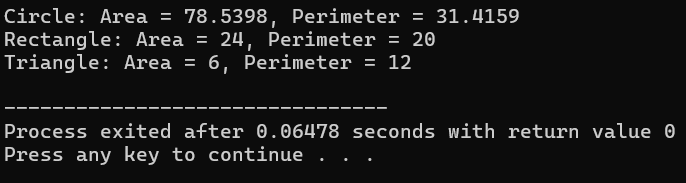
cout << "Circle: Area = " << circle.area() << ", Perimeter = " << circle.perimeter() << endl;

cout << "Rectangle: Area = " << rectangle.area() << ", Perimeter = " << rectangle.perimeter() << endl;

cout << "Triangle: Area = " << triangle.area() << ", Perimeter = " << triangle.perimeter() << endl;

return 0;

}



**3.Design a class hierarchy to represent a university system. Include classes for students, professors, and courses. Use appropriate OOP principles like encapsulation, inheritance, and polymorphism. Provide a brief explanation of how your design promotes code reusability and flexibility**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Person {

protected:

string name;

int age;

string contactInfo;

public:

Person(const string& n, int a, const string& contact) : name(n), age(a), contactInfo(contact) {}

string getName() const { return name; }

int getAge() const { return age; }

string getContactInfo() const { return contactInfo; }

};

class Student : public Person {

private:

string studentID;

float GPA;

public:

Student(const string& n, int a, const string& contact, const string& id, float gpa)

: Person(n, a, contact), studentID(id), GPA(gpa) {}

string getStudentID() const { return studentID; }

float getGPA() const { return GPA; }

};

class Professor : public Person {

private:

string employeeID;

string department;

public:

Professor(const string& n, int a, const string& contact, const string& empID, const string& dept)

: Person(n, a, contact), employeeID(empID), department(dept) {}

string getEmployeeID() const { return employeeID; }

string getDepartment() const { return department; }

};

class Course {

private:

string courseCode;

string title;

string description;

Professor professor;

vector<Student> enrolledStudents;

public:

Course(const string& code, const string& t, const string& desc, const Professor& prof)

: courseCode(code), title(t), description(desc), professor(prof) {}

void addStudent(const Student& student) { enrolledStudents.push\_back(student); }

void removeStudent(const Student& student) { /\* Remove student from enrolledStudents vector \*/ }

string getCourseCode() const { return courseCode; }

string getTitle() const { return title; }

string getDescription() const { return description; }

Professor getProfessor() const { return professor; }

const vector<Student>& getEnrolledStudents() const { return enrolledStudents; }

};

int main() {

Professor prof("John Doe", 45, "john.doe@example.com", "EMP123", "Computer Science");

Student student1("Alice Smith", 20, "alice.smith@example.com", "S123", 3.5);

Student student2("Bob Johnson", 22, "bob.johnson@example.com", "S456", 3.2);

Course course("CS101", "Introduction to Programming", "This course covers basic programming concepts", prof);

course.addStudent(student1);

course.addStudent(student2);

cout << "Course Code: " << course.getCourseCode() << endl;

cout << "Course Title: " << course.getTitle() << endl;

cout << "Course Description: " << course.getDescription() << endl;

cout << "Professor: " << course.getProfessor().getName() << endl;

cout << "Enrolled Students:" << endl;

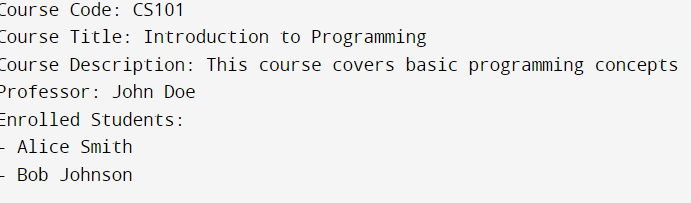
for (const auto& student : course.getEnrolledStudents()) {

cout << "- " << student.getName() << endl;

}

return 0;

}

****

**4.** **Design a class hierarchy to represent a geometric shapes system. Include classes for circles, rectangles, and triangles. Implement methods for calculating the area and perimeter of each shape. Demonstrate the use of abstract classes and pure virtual functions. Discuss 10 K3 CO1 how this design supports future additions of new shapes**

#include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

double area() const { return 0; }

double perimeter() const { return 0; }

~Shape() {}

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double area() const {

return M\_PI \* radius \* radius;

}

double perimeter() const {

return 2 \* M\_PI \* radius;

}

};

class Rectangle : public Shape {

private:

double length, width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() const {

return length \* width;

}

double perimeter() const {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

double side1, side2, side3;

public:

Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

double area() const {

double s = (side1 + side2 + side3) / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() const {

return side1 + side2 + side3;

}

};

int main() {

Circle circle(5);

Rectangle rectangle(4, 6);

Triangle triangle(3, 4, 5);

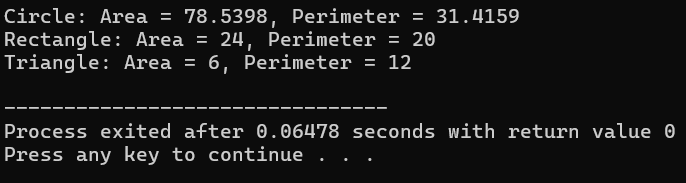
cout << "Circle: Area = " << circle.area() << ", Perimeter = " << circle.perimeter() << endl;

cout << "Rectangle: Area = " << rectangle.area() << ", Perimeter = " << rectangle.perimeter() << endl;

cout << "Triangle: Area = " << triangle.area() << ", Perimeter = " << triangle.perimeter() << endl;

return 0;

}



**5.In an organization they decide to give bonus to all the employees on New Year. A 5% bonus on salary is given to the grade A workers and 10% bonus on salary to the grade B workers. Write a program to enter the salary and grade of the employee. If the salary of the employee is less than $10,000 then the employee gets an extra 2% bonus on salary Calculate the bonus that has to be given to the employee and print the salary that the employee will get.**

**Sample Input & Output:**

**Enter the grade of the employee: B**

**Enter the employee salary: 50000**

**Salary=50000 Bonus=5000.0**

**Total to be paid: 55000.0**

#include<iostream>

using namespace std;

int main(){

int salary;

char grade;

double bonus,orginal;

cout<<"enter the salary of the empolyee:";

cin>>salary;

cout<<"enter the grade of employee:";

cin>>grade;

if(grade=='A'){

if(salary<10000){

bonus=salary\*0.07;

}

else{

bonus=salary\*0.05;

}

orginal=salary+bonus;

}

if(grade=='B'){

if(salary<10000)

{

bonus=salary\*0.12;

}

else{

bonus=salary\*0.10;

}

orginal=salary+bonus;

}

cout<<" salary is :"<<salary<<endl;

cout<<"bonus is:"<<bonus<<endl;

cout<<"orginal is:"<<orginal;

}

