**Assignment 04**

**Q1)**

#include <iostream>

#include <string.h>

using namespace std;

class Employee

{

private:

    int id;

    char name[20];

    double salary;

    // Conclassor

public:

    Employee()

    {

        // cout << "\nDefault conclassor called\n";

        this->id = 0;

        strcpy(this->name, "No Name");

        this->salary = 0;

    }

    Employee(char \*name, int id, double salary)

    {

        // cout << "\nParameterized Conclassor for Employee called";

        strcpy(this->name, name);

        this->id = id;

        this->salary = salary;

    }

    // Setters

    void setId(int Id)

    {

        this->id = Id;

    }

    void setName(char \*name)

    {

        strcpy(this->name, name);

    }

    void setSalary(double salary)

    {

        this->salary = salary;

    }

    // Getters

    int getId()

    {

        return this->id;

    }

    char \*getName()

    {

        return this->name;

    }

    double getSalary()

    {

        return this->salary;

    }

    // Calculate salary

    virtual double CalculateSalary()

    {

        return salary; // Basic salary for general employees

    }

    // Display

    virtual void display()

    {

        cout << "\nId : " << this->id << "\tName :" << this->name << "\t Salary :" << this->salary;

    }

};

class SalesManager : public Employee

{

private:

    double incentive;

    int target;

public:

    // Conclassor

    SalesManager()

    {

        // cout << "\nDefault conclassor called\n";

        this->incentive = 0;

        this->target = 0;

    }

    SalesManager(char \*name, int id, double salary, double incentive, int target) : Employee(name, id, salary)

    {

        // cout << "\nParameterized Conclassor for SalesManager called";

        this->incentive = incentive;

        this->target = target;

    }

    // CalculateSalary

    double CalculateSalary()

    {

        return getSalary() + incentive; // Total salary = Basic Salary + Incentive

    }

public:

    // Setters

    void setIncentive(double incentive)

    {

        this->incentive = incentive;

    }

    void setTarget(int target)

    {

        this->target = target;

    }

    // Getters

    double getIncentive()

    {

        return this->incentive;

    }

    int getTarget()

    {

        return this->target;

    }

    // Display

    void display()

    {

        Employee::display();

        cout << "\nIncentive : " << this->incentive << "\tTarget : " << this->target;

    }

};

class Admin : public Employee

{

    // id,name,salary,allowence

private:

    double allowence;

public:

    // Construuctor

    Admin()

    {

        // cout << "\nDefault conclassor called Admin\n";

        this->allowence = 00;

    }

    Admin(char \*name, int id, double salary, double allowence) : Employee(name, id, salary)

    {

        // cout << "\nParameterized Conclassor for Admin called";

        this->allowence = allowence;

    }

    // Setters

    void setAllowence(double allowence)

    {

        this->allowence = allowence;

    }

    // CalculateSalary

    double CalculateSalary()

    {

        return getSalary() + allowence; // Total salary = Basic Salary + Allowance

    }

    // Display

    void display()

    {

        Employee::display();

        cout << "\nAllowence :" << this->allowence;

    }

    // getters

    double getAllowence()

    {

        return this->allowence;

    }

};

class HR : public Employee

{

private:

    double commission;

public:

    // Conclassor

    HR()

    {

        // cout << "\nDefault conclassor called HR\n";

        this->commission = 0;

    }

    HR(char \*name, int id, double salary, double commission) : Employee(name, id, salary)

    {

        // cout << "\nParameterized Conclassor for HR called";

        this->commission = commission;

    }

    // Setters

    void setCommission(double commission)

    {

        this->commission = commission;

    }

    // Getters

    double getCommission()

    {

        return this->commission;

    }

    // CalculateSalary

    double CalculateSalary()

    {

        return getSalary() + (getSalary() \* commission / 100); // Total salary = Basic Salary + Commission

    }

    // Display

    void display()

    {

        Employee::display();

        cout << "\nCommission : " << this->commission;

    }

};

class AreaSalesManager : public SalesManager

{

private:

    char location[20];

public:

    AreaSalesManager()

    {

        // cout << "\nDefault Conclassor for AreaSalesManager called";

        strcpy(location, "Not Given");

    }

    AreaSalesManager(char \*name, int id, double salary, double incentive, int target, char \*location) : SalesManager(name, id, salary, incentive, target)

    {

        // cout << "\nParameterized Conclassor for AreaSalesManager called";

        strcpy(this->location, location);

    }

    // CalculateSalary

    double CalculateSalary()

    {

        return SalesManager::CalculateSalary(); // Inherits from SalesManager

    }

    void display()

    {

        SalesManager::display();

        cout << "\nLocation :" << this->location;

    }

};

int main()

{

    Employee \*employee[5];

    employee[0] = new SalesManager("Bhagvat", 123, 500000, 1200, 22);

    employee[1] = new AreaSalesManager("Bhagvat", 123, 690000, 1200, 2, "Pune");

    employee[2] = new HR("Pinto", 124, 560000, 345);

    employee[3] = new Admin("Teja", 122, 780000, 3233);

    for (int i = 0; i < 4; i++)

    {

        employee[i]->display();

        cout << "\nTotal Salary: " << employee[i]->CalculateSalary(); // Display total salary

    }

    // cout << "\n\nSales Manager Data :\n";

    // SalesManager s1("Bhagvat", 123, 690000, 1200, 2);

    // s1.display();

    // cout << "\n\nArea Sales Manager Data :\n";

    // AreaSalesManager As1("Bhagvat", 123, 690000, 1200, 2, "Pune");

    // As1.display();

    // cout << "\n\nHR Data :\n";

    // HR hr("Pinto", 124, 560000, 345);

    // hr.display();

    // cout << "\n\nAdmin Data :\n";

    // Admin admin("Teja", 122, 780000, 3233);

    // admin.display();

    return 1;

}

Output:

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output> & .\'q1Employee.exe'

Id : 123 Name :Bhagvat Salary :500000

Incentive : 1200 Target : 22

Total Salary: 501200

Id : 123 Name :Bhagvat Salary :690000

Incentive : 1200 Target : 2

Location :Pune

Total Salary: 691200

Id : 124 Name :Pinto Salary :560000

Commission : 345

Total Salary: 2.492e+06

Id : 122 Name :Teja Salary :780000

Allowence :3233

Total Salary: 783233

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output>

**Q2)**

#include <iostream>

using namespace std;

class Shape

{

private:

    double area;

public:

    Shape()

    {

        area = 0;

    }

    virtual void calculateArea()

    {

        area = 0;

    }

    void setArea(double area) { this->area = area; }

    double getArea() { return this->area; }

    virtual void display()

    {

        cout << "Area: " << this->area << endl;

    }

};

class Circle : public Shape

{

private:

    double radius;

public:

    // Constructor

    Circle(double r)

    {

        radius = r;

        calculateArea();

    }

    Circle()

    {

        radius = 0;

        calculateArea();

    }

    void calculateArea()

    {

        this->setArea(3.14 \* radius \* radius); // Area of the circle

    }

    void display()

    {

        cout << "Circle with radius: " << radius << endl;

        Shape::display(); // Call base class display

    }

};

class Triangle : public Shape

{

private:

    double base;

    double height;

public:

    // Constructor

    Triangle(double b, double h)

    {

        base = b;

        height = h;

        calculateArea();

    }

    Triangle()

    {

        base = 0;

        height = 0;

        calculateArea();

    }

    void calculateArea()

    {

        this->setArea(0.5 \* base \* height);

    }

    void display()

    {

        cout << "Triangle with base: " << base << " and height: " << height << endl;

        Shape::display();

    }

};

class Rectangle : public Shape

{

private:

    double length;

    double width;

public:

    // Constructor

    Rectangle(double l, double w)

    {

        length = l;

        width = w;

        calculateArea();

    }

    Rectangle()

    {

        length = 0;

        width = 0;

        calculateArea();

    }

    void calculateArea()

    {

        this->setArea(length \* width);

    }

    void display()

    {

        cout << "Rectangle with length: " << length << " and width: " << width << endl;

        Shape::display();

    }

};

int main()

{

    int choice;

    do

    {

        cout << "\n\nWhat do you want to do: "

             << "\n1) Calculate area of Triangle "

             << "\n2) Calculate area of Circle "

             << "\n3) Calculate area of Rectangle "

             << "\n0) Exit"

             << "\nEnter Your Choice: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

        {

            double base, height;

            cout << "\nEnter Base: ";

            cin >> base;

            cout << "\nEnter Height: ";

            cin >> height;

            Triangle triangle(base, height);

            triangle.display();

            break;

        }

        case 2:

        {

            double radius;

            cout << "\nEnter radius: ";

            cin >> radius;

            Circle circle(radius);

            circle.display();

            break;

        }

        case 3:

        {

            double length, width;

            cout << "\nEnter length: ";

            cin >> length;

            cout << "\nEnter width: ";

            cin >> width;

            Rectangle rectangle(length, width);

            rectangle.display();

            break;

        }

        default:

        {

            if (choice != 0)

            {

                cout << "\nInvalid Choice....! ";

            }

            break;

        }

        }

    } while (choice != 0);

    return 0;

}

Output: PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output> & .\'q2Shapes.exe'

What do you want to do:

1) Calculate area of Triangle

2) Calculate area of Circle

3) Calculate area of Rectangle

0) Exit

Enter Your Choice: 1

Enter Base: 34

Enter Height: 2

Triangle with base: 34 and height: 2

Area: 34

What do you want to do:

1) Calculate area of Triangle

2) Calculate area of Circle

3) Calculate area of Rectangle

0) Exit

Enter Your Choice: 2

Enter radius: 6.56

Circle with radius: 6.56

Area: 135.126

What do you want to do:

1) Calculate area of Triangle

2) Calculate area of Circle

3) Calculate area of Rectangle

0) Exit

Enter Your Choice: 3

Enter length: 12

Enter width: 30

Rectangle with length: 12 and width: 30

Area: 360

What do you want to do:

1) Calculate area of Triangle

2) Calculate area of Circle

3) Calculate area of Rectangle

0) Exit

Enter Your Choice: 5

Invalid Choice....!

What do you want to do:

1) Calculate area of Triangle

2) Calculate area of Circle

3) Calculate area of Rectangle

0) Exit

Enter Your Choice: 0

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output>

**3)**

#include <iostream>

#include <string.h>

using namespace std;

struct Vehicle

{

    virtual void start() { cout << "\nVehicle Start"; }

    virtual void stop() { cout << "\nVehicle Stop"; }

    virtual void brake() { cout << "\nVehicle Brake"; }

};

struct Car : public Vehicle

{

    void start() { cout << "\nCar Start"; }

    void brake() { cout << "\nCar Brake"; }

};

struct Bus : public Vehicle

{

    void start() { cout << "\nBus Start"; }

    void brake() { cout << "\nBus Brake"; }

};

struct Bike : Vehicle

{

    void start() { cout << "\nBike Start"; }

    void stop() { cout << "\nBike Stop"; }

};

int main()

{

    Vehicle \*vehicles[5];

    for (int i = 0; i < 5; i++)

    {

        if (i % 2 == 0)

        {

            vehicles[i] = new Car;

        }

        else if (i % 3 == 0)

        {

            vehicles[i] = new Bus;

        }

        else

        {

            vehicles[i] = new Bike;

        }

    }

    for (int i = 0; i < 5; i++)

    {

        vehicles[i]->start();

        vehicles[i]->brake();

        vehicles[i]->stop();

        cout << endl;

    }

    // Car car;

    // Bus Bus;

    // Bike bike;

    // car.start();

    // Bus.start();

    // bike.start();

    return 0;

}

Output:

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output> & .\'q3Vehicle.exe'

Car Start

Car Brake

Vehicle Stop

Bike Start

Vehicle Brake

Bike Stop

Car Start

Car Brake

Vehicle Stop

Bus Start

Bus Brake

Vehicle Stop

Car Start

Car Brake

Vehicle Stop

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output>

**Q4) 1)**

#include <iostream>

using namespace std;

class BankAccount

{

public:

    // void withdrow()

    virtual void withdrow()

    {

        cout << "\n Bank Withdrow";

    }

};

class Savings : public BankAccount

{

public:

    void withdrow()

    {

        cout << "\nSaving  Withdrow";

    }

};

class Current : public BankAccount

{

public:

    void withdrow()

    {

        cout << "\nCurrent  Withdrow";

    }

};

class Loan : public BankAccount

{

public:

    void withdrow()

    {

        cout << "\nLoan Withdrow";

    }

};

int main()

{

    BankAccount \*bankAccounts[5];

    for (int i = 0; i < 5; i++)

    {

        if (i / 2 == 0)

        {

            bankAccounts[i] = new Savings();

        }

        else if (i % 2 == 0)

        {

            bankAccounts[i] = new Current();

        }

        else

        {

            bankAccounts[i] = new Loan();

        }

    }

    for (int i = 0; i < 5; i++)

    {

        bankAccounts[i]->withdrow();

    }

    return 0;

}

Output: PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output> & .\'q4\_1BankAccount.exe'

Saving Withdrow

Saving Withdrow

Current Withdrow

Loan Withdrow

Current Withdrow

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output>

**Q4) 2)**

#include <iostream>

using namespace std;

class GameCharacter

{

public:

    // void attack()

    virtual void attack()

    {

        cout << "\nGame Character attack";

    }

};

class Worrier : public GameCharacter

{

public:

    void attack()

    {

        cout << "\nWorrier  attack";

    }

};

class Mage : public GameCharacter

{

public:

    void attack()

    {

        cout << "\nMage  attack";

    }

};

class Archer : public GameCharacter

{

public:

    void attack()

    {

        cout << "\nArcher attack";

    }

};

class Trickster : public GameCharacter

{

public:

    void attack()

    {

        cout << "\nTrickster attack";

    }

};

int main()

{

    GameCharacter \*gameCharacters[10];

    for (int i = 0; i < 10; i++)

    {

        if (i / 2 == 0)

        {

            gameCharacters[i] = new Worrier();

        }

        else if (i % 2 == 0)

        {

            gameCharacters[i] = new Mage();

        }

        else if (i % 3 == 0)

        {

            gameCharacters[i] = new Archer();

        }

        else

        {

            gameCharacters[i] = new Trickster();

        }

    }

    for (int i = 0; i < 10; i++)

    {

        gameCharacters[i]->attack();

    }

    return 0;

}

Output: PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output> & .\'q4\_2GameCharacter.exe'

Worrier attack

Worrier attack

Mage attack

Archer attack

Mage attack

Trickster attack

Mage attack

Trickster attack

Mage attack

Archer attack

PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment04\output>