OBJECT ORIENTED PROGRAMMING

ABDULLAH KHALID 035

Syed Muhammad Shahraiz

TASK 1

```
#include <iostream>
#include <string>
using namespace std;

class Publish {
protected:
    string title;
    float price;

public:
    void getdata() {
        cout << "Enter title: "<<endl;
        getline(cin, title);
        cout << "Enter price: ";
        cin >> price;
    }

    void putdata() const {
        cout << "Title: " << title << endl;
```

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

class Book : public Publish {
private:
    int page_count;

public:
    void getdata() {

        Publish::getdata();
        cout << "Enter page count: ";
        cin >> page_count;
    }

    void putdata() const {
        Publish::putdata();
        cout << "Page Count: " << page_count << endl;
    }
```

```
};

class Tape : public Publish {
private:
   float playing_time;

public:
   void getdata() {
      Publish::getdata();
      cout << "Enter playing time (in minutes): ";
      cin >> playing_time;
   }

   void putdata() const {
      Publish::putdata();
      cout << "Playing Time: " << playing_time << " minutes" <<
endl;
   }
};
```

```
int main() {
    cout << "Enter details for a book:" << endl;
    Book book;
    book.getdata();

    cout << "Enter details for a tape:" << endl;
    Tape tape;
    tape.getdata();

    cout << "Displaying book details:" << endl;
    book.putdata();

    cout << "Displaying tape details:" << endl;
    tape.putdata();

    return 0;
}
```

```
F:\endless\opp1.exe

Enter details for a book:
Enter title:
hobby
Enter price: 1000
Enter page count: 13
Enter details for a tape:
Enter title:
Enter price: 1000
Enter playing time (in minutes): 5
Displaying book details:
Title: hobby
Price: $1000
Page Count: 13
Displaying tape details:
Title:
Price: $1000
Playing Time: 5 minutes

--------------------------------
Process exited after 25.67 seconds with return value 0
Press any key to continue . . .
```

TASK 2

```
#include <iostream>

#include <string>

using namespace std;


class Publication {

protected:

    string title;

    float price;
```

```
public:
    void getdata() {
        cout << "Enter title: ";
        cin.ignore(); // Ignore any leftover newline character
        getline(cin, title);
        cout << "Enter price: ";
        cin >> price;
    }

    void putdata() const {
        cout << "Title: " << title << endl;
        cout << "Price: $" << price << endl;
    }
};

class Sales {
protected:
    float sales[3];
```

```
public:
    void getdata() {
        cout << "Enter sales for the last three months:" << endl;
        for (int i = 0; i < 3; i++) {
            cout << "Month " << i + 1 << ": ";
            cin >> sales[i];
        }
    }

    void putdata() const {
        cout << "Sales 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 {
private:
    int page_count;
```

```
public:
    void getdata() {
        Publication::getdata();
        cout << "Enter page count: ";
        cin >> page_count;
        Sales::getdata();
    }

    void putdata() const {
        Publication::putdata();
        cout << "Page Count: " << page_count << endl;
        Sales::putdata();
    }
};

class Tape : public Publication, public Sales {
private:
    float playing_time;
```

```
public:
    void getdata() {
        Publication::getdata();
        cout << "Enter playing time (in minutes): ";
        cin >> playing_time;
        Sales::getdata();
    }

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

int main() {
    cout << "Enter details for a book:" << endl;
    Book book;
    book.getdata();
```

```
    cout << "Enter details for a tape:" << endl;
    Tape tape;
    tape.getdata();

    cout << "Displaying book details:" << endl;
    book.putdata();

    cout << "Displaying tape details:" << endl;
    tape.putdata();

    return 0;
}
```

```
Enter title: Muskal mehal
Enter price: 2000
Enter page count: 24
Enter sales for the last three months:
Month 1: 45
Month 2: 40
Month 3: 30
Enter details for a tape:
Enter title: Robbin
Enter price: 3000
Enter playing time (in minutes): 6
Enter sales for the last three months:
Month 1: 30
Month 2: 45
Month 3: 47
Displaying book details:
Title: uskal mehal
Price: $2000
Page Count: 24
Sales for the last three months:
Month 1: $45
Month 2: $40
Month 3: $30
Displaying tape details:
Title: Robbin
Price: $3000
Playing Time: 6 minutes
Sales for the last three months:
Month 1: $30
Month 2: $45
Month 3: $47
```

TASK 3

```
#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(); // Ignore any leftover newline character
        getline(cin, title);
        cout << "Enter price: ";
        cin >> price;
    }

    void putdata() const {
        cout << "Title: " << title << endl;
        cout << "Price: $" << price << endl;
    }
};

class Sales {
```

```
protected:
    float sales[3];

public:
    void getdata() {
        cout << "Enter sales for the last three months:" << endl;
        for (int i = 0; i < 3; i++) {
            cout << "Month " << i + 1 << ": ";
            cin >> sales[i];
        }
    }

    void putdata() const {
        cout << "Sales 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 {
private:
    int page_count;

public:
    void getdata() {
        Publication::getdata();
        cout << "Enter page count: ";
        cin >> page_count;
        Sales::getdata();
    }

    void putdata() const {
        Publication::putdata();
        cout << "Page Count: " << page_count << endl;
        Sales::putdata();
    }
};

class Tape : public Publication, public Sales {
```

```
private:
    float playing_time;

public:
    void getdata() {
        Publication::getdata();
        cout << "Enter playing time (in minutes): ";
        cin >> playing_time;
        Sales::getdata();
    }

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

class Disk : public Publication, public Sales {
```

```
private:
    DiskType disk_type;

public:
    void getdata() {
        Publication::getdata();
        char type;
        cout << "Enter disk type (c for CD, d for DVD): ";
        cin >> type;
        if (type == 'c' || type == 'C') {
            disk_type = CD;
        } else if (type == 'd' || type == 'D') {
            disk_type = DVD;
        } else {
            cout << "Invalid disk type! Defaulting to CD." << endl;
            disk_type = CD;
        }
        Sales::getdata();
    }
```

```
    void putdata() const {
        Publication::putdata();
        cout << "Disk Type: " << (disk_type == CD ? "CD" : "DVD")
<< endl;
        Sales::putdata();
    }
};

int main() {
    cout << "Enter details for a book:" << endl;
    Book book;
    book.getdata();

    cout << "Enter details for a tape:" << endl;
    Tape tape;
    tape.getdata();

    cout << "Enter details for a disk:" << endl;
    Disk disk;
    disk.getdata();
```

```
    cout << "Displaying book details:" << endl;
    book.putdata();

    cout << "Displaying tape details:" << endl;
    tape.putdata();

    cout << "Displaying disk details:" << endl;
    disk.putdata();

    return 0;
}
```

```
F:\endless\opp 3.exe
Enter price: 150
Enter page count: 15
Enter sales for the last three months:
Month 1: 14
Month 2: 15
Month 3: 10
Enter details for a tape:
Enter title: robbin
Enter price: 200
Enter playing time (in minutes): 4
Enter sales for the last three months:
Month 1: 50
Month 2: 4
Month 3: 40
Enter details for a disk:
Enter title: Playing hob
Enter price: 2000
Enter disk type (c for CD, d for DVD): c
Enter sales for the last three months:
Month 1: 13
Month 2: 24
Month 3: 43
Displaying book details:
Title: ath
Price: $150
Page Count: 15
Sales for the last three months:
Month 1: $14
Month 2: $15
Month 3: $10
```

TASK 4

```
#include <iostream>

#include <string>

using namespace std;


class Employee {

protected:

    string name;
```

```
    unsigned long number;

public:
    void getdata() {
        cout << "Enter name: ";
        cin.ignore(); // Ignore any leftover newline character
        getline(cin, name);
        cout << "Enter number: ";
        cin >> number;
    }

    void putdata() const {
        cout << "Name: " << name << endl;
        cout << "Number: " << number << endl;
    }
};
enum Period { HOURLY, WEEKLY, MONTHLY };

class Employee2 : public Employee {
protected:
```

```
    double compensation;
    Period period;

public:
    void getdata() {
        Employee::getdata();
        cout << "Enter compensation: ";
        cin >> compensation;
        char periodChoice;
        cout << "Enter period (h for hourly, w for weekly, m for
monthly): ";
        cin >> periodChoice;
        switch (periodChoice) {
            case 'h': case 'H':
                period = HOURLY;
                break;
            case 'w': case 'W':
                period = WEEKLY;
                break;
            case 'm': case 'M':
```

```
            period = MONTHLY;
            break;
        default:
            cout << "Invalid choice! Defaulting to hourly." << endl;
            period = HOURLY;
            break;
    }
}

void putdata() const {
    Employee::putdata();
    cout << "Compensation: " << compensation << endl;
    cout << "Period: ";
    switch (period) {
        case HOURLY:
            cout << "Hourly" << endl;
            break;
        case WEEKLY:
            cout << "Weekly" << endl;
            break;
```

```
            case MONTHLY:
                cout << "Monthly" << endl;
                break;
        }
    }
};
class Manager : public Employee2 {
public:
    void getdata() {
        cout << "Enter details for Manager:" << endl;
        Employee2::getdata();
    }

    void putdata() const {
        cout << "Manager details:" << endl;
        Employee2::putdata();
    }
};

class Scientist : public Employee2 {
```

```
public:
    void getdata() {
        cout << "Enter details for Scientist:" << endl;
        Employee2::getdata();
    }

    void putdata() const {
        cout << "Scientist details:" << endl;
        Employee2::putdata();
    }
};

class Laborer : public Employee2 {
public:
    void getdata() {
        cout << "Enter details for Laborer:" << endl;
        Employee2::getdata();
    }

    void putdata() const {
```

```
        cout << "Laborer details:" << endl;
        Employee2::putdata();
    }
};
int main() {
    Manager mgr;
    Scientist sci;
    Laborer lab;

    cout << "Writing the  details for Manager:" << endl;
    mgr.getdata();

    cout << "Enter details for Scientist:" << endl;
    sci.getdata();

    cout << "Enter details for Laborer:" << endl;
    lab.getdata();

    cout << "Displaying Manager details:" << endl;
    mgr.putdata();
```

```
    cout << "Displaying Scientist details:" << endl;
    sci.putdata();

    cout << "Displaying Laborer details:" << endl;
    lab.putdata();

    return 0;
}
```

```
F:\endless\opp 4.exe
Writing the  details for Manager:
Enter details for Manager:
Enter name: Abdullah
Enter number: 35
Enter compensation: 15
Enter period (h for hourly, w for weekly, m for monthly): h
Enter details for Scientist:
Enter details for Scientist:
Enter name: Arham
Enter number: 59
Enter compensation: 4
Enter period (h for hourly, w for weekly, m for monthly): m
Enter details for Laborer:
Enter details for Laborer:
Enter name: sharaiz
Enter number: 45
Enter compensation: 10
Enter period (h for hourly, w for weekly, m for monthly): w
Displaying Manager details:
Manager details:
Name: bdullah
Number: 35
Compensation: 15
Period: Hourly
Displaying Scientist details:
Scientist details:
Name: Arham
Number: 59
Compensation: 4
Period: Monthly
```

TASK 5

```
#include <iostream>

#include <string>

#include <cmath>


class Shape {

protected:
```

```
    std::string color;
public:
    Shape(const std::string& color) : color(color) {}

    void printColor() {
        std::cout << "Color: " << color << std::endl;
    }
};

class Circle : public Shape {
private:
    double radius;
public:
     Circle(const std::string& color, double radius) : Shape(color),
radius(radius) {}

    double calculateArea() {
        return M_PI * radius * radius;
    }
```

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

class Rectangle : public Shape {
private:
    double length;
    double width;
public:
        Rectangle(const std::string& color, double length, double width) : Shape(color), length(length), width(width) {}

    double calculateArea() {
        return length * width;
    }

    void printArea() {
            std::cout << "Rectangle Area: " << calculateArea() << std::endl;
    }
```

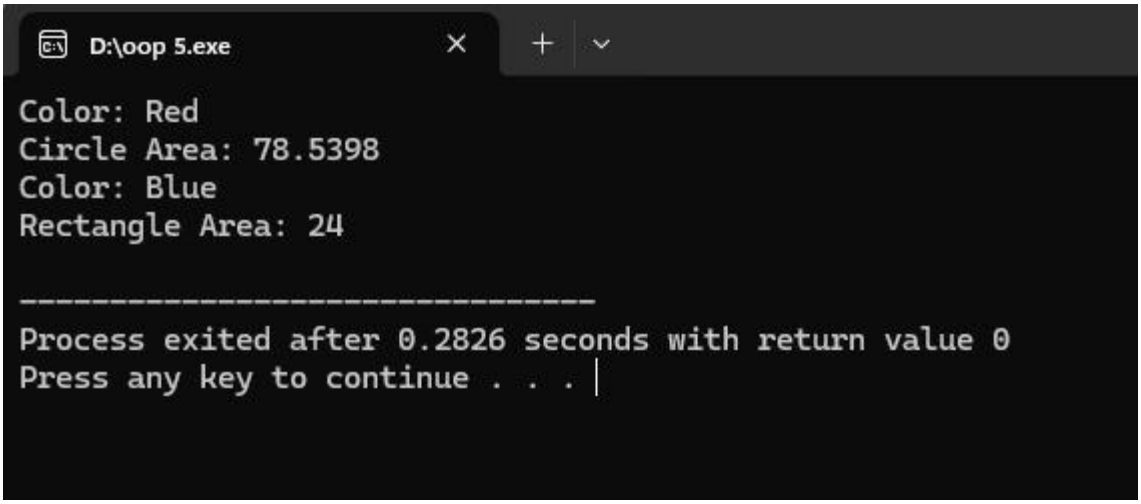
```
};

int main() {
    Circle circle("Red", 5.0);
    Rectangle rectangle("Blue", 4.0, 6.0);

    circle.printColor();
    circle.printArea();

    rectangle.printColor();
    rectangle.printArea();

    return 0;
}
```



TASK 6

```
#include <iostream>
#include <string>

using namespace std;

class Employee {
protected:
    string name;
    int employeeID;
    string department;

public:
    // Constructor
    Employee(const string& n, int id, const string& dept)
        : name(n), employeeID(id), department(dept) {}

    // Member functions to get and set the values
    void setName(const string& n) {
        name = n;
```

```
}

string getName() const {
    return name;
}

void setEmployeeID(int id) {
    employeeID = id;
}

int getEmployeeID() const {
    return employeeID;
}

void setDepartment(const string& dept) {
    department = dept;
}

string getDepartment() const {
    return department;
```

```
    }

    // Virtual function to be overridden by derived classes
    virtual void displayDetails() const {
        cout << "Name: " << name << endl;
        cout << "Employee ID: " << employeeID << endl;
        cout << "Department: " << department << endl;
    }
};
class SalariedEmployee : public Employee {
private:
    double annualSalary;

public:
    // Constructor
    SalariedEmployee(const string& n, int id, const string& dept,
double salary)
        : Employee(n, id, dept), annualSalary(salary) {}

    // Member functions to get and set the salary
```

```
    void setAnnualSalary(double salary) {
        annualSalary = salary;
    }

    double getAnnualSalary() const {
        return annualSalary;
    }

    // Function to calculate the monthly pay
    double calculateMonthlyPay() const {
        return annualSalary / 12;
    }

    // Override displayDetails to include salary information
    void displayDetails() const override {
        Employee::displayDetails();
        cout << "Annual Salary: $" << annualSalary << endl;
        cout << "Monthly Pay: $" << calculateMonthlyPay() << endl;
    }
};
```

```
class CommissionEmployee : public Employee {
private:
    double sales;
    double commissionRate;

public:
    // Constructor
    CommissionEmployee(const string& n, int id, const string&
dept, double s, double rate)
        : Employee(n, id, dept), sales(s), commissionRate(rate) {}

    // Member functions to get and set sales and commission
rate
    void setSales(double s) {
        sales = s;
    }

    double getSales() const {
        return sales;
    }
```

```
    void setCommissionRate(double rate) {
        commissionRate = rate;
    }

    double getCommissionRate() const {
        return commissionRate;
    }

    // Function to calculate the total pay
    double calculateTotalPay() const {
        return sales * commissionRate;
    }

    // Override displayDetails to include sales and commission
information
    void displayDetails() const override {
        Employee::displayDetails();
        cout << "Sales: $" << sales << endl;
        cout << "Commission Rate: " << commissionRate * 100 <<
"%" << endl;
        cout << "Total Pay: $" << calculateTotalPay() << endl;
```

```
    }
};
int main() {
    // Creating a SalariedEmployee object
    SalariedEmployee salariedEmp("John Doe", 101, "Finance",
60000);
    cout << "Salaried Employee Details:" << endl;
    salariedEmp.displayDetails();

    // Creating a CommissionEmployee object
    CommissionEmployee commissionEmp("Jane Smith", 102,
"Sales", 150000, 0.10);
    cout << "\nCommission Employee Details:" << endl;
    commissionEmp.displayDetails();

    return 0;
}
```

F:\endless\oop 6.exe

```
Salaried Employee Details:
Name: John Doe
Employee ID: 101
Department: Finance
Annual Salary: $60000
Monthly Pay: $5000

Commission Employee Details:
Name: Jane Smith
Employee ID: 102
Department: Sales
Sales: $150000
Commission Rate: 10%
Total Pay: $15000

--------------------------------
Process exited after 0.1262 seconds with return value 0
Press any key to continue . . .
```