Eman Tahir

2023-BS-AI-015

Section-A

OBJECT ORIENTED PROGRAMMING

LAB TASK

Q1: Imagine a publishing company that markets both book and audiocassetes versions of its works. Create a class publicaion that stores the Ɵtle (a string) and price (type float) of a publicaon. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book and tape classes by creatng instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question statement:Imagine a publishing company that markets both book and audiocassetes versions of its works. Create a class publicaion that stores the ?tle (a string) and price (type float) of a publicaon. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book and tape classes by creatng instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

#include<iostream>

using namespace std;

class publication

{

private:

int pageC;

float pageT;

public:

string title;

float price;

int book()

{

int pageC;

}

int tape()

{

float pageT;

}

int getdata()

{

cout<<"enter title of book: ";

cin>>title;

cout<<"enter price of book: ";

cin>>price;

cout<<"enter number of pages: ";

cin>>pageC;

cout<<"enter time in minutes: ";

cin>>pageT;

}

int putdata()

{

cout<<"publications detail:"<<endl;

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

cout<<"price: "<<price<<" Rs"<<endl;

cout<<"number of pages: "<<pageC<<endl;

cout<<"playing time: "<<pageT<<" minutes"<<endl;

}

};

int main ()

{

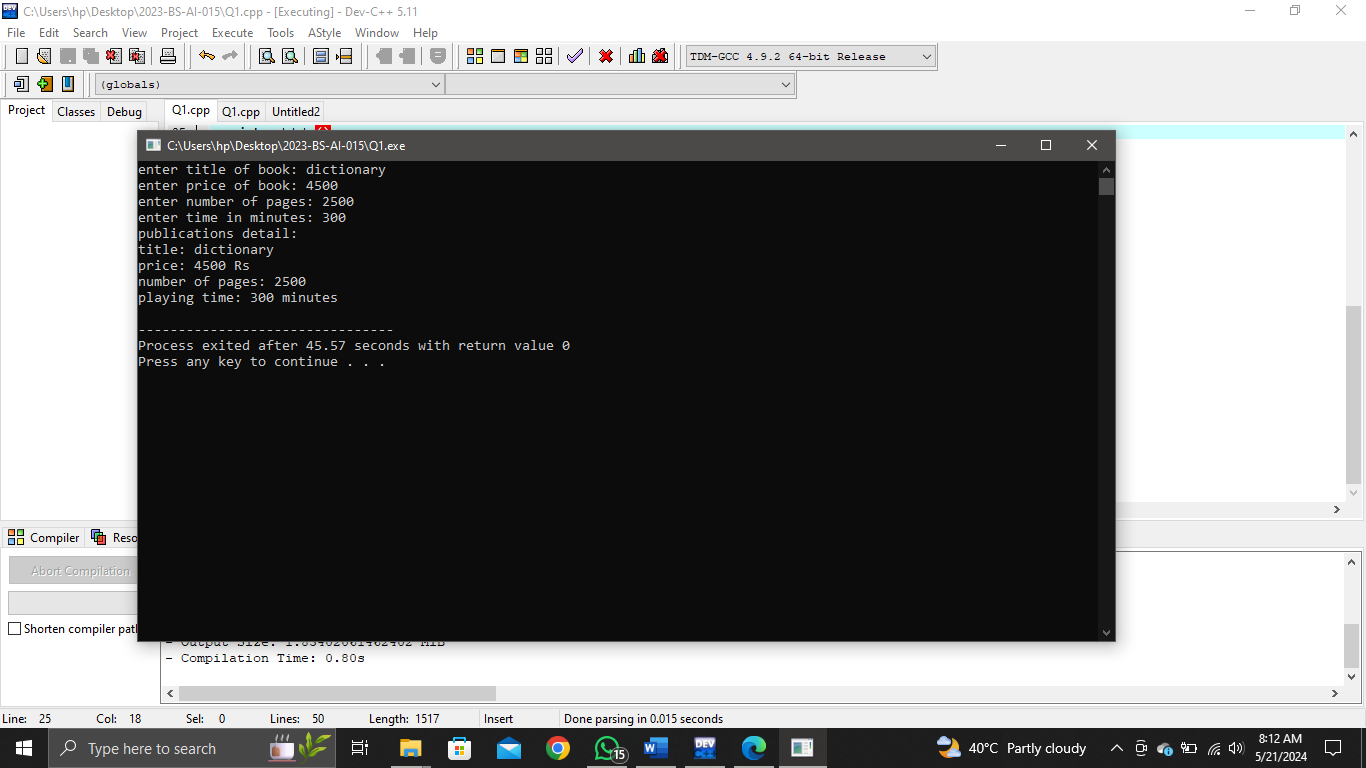
publication p;

p.getdata();

p.putdata();

}

Output:



Q2: Start with the publication, book, and tape classes of Question 1. Add a base class sales that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to get three sales amounts from the user, and a putdata() function to display the sales figures. 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.

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question statement:Start with the publication, book, and tape classes of Question 1. Add a base class sales that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to get three sales amounts from the user, and a putdata() function to display the sales figures. 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>

using namespace std;

class publication

{

private:

int pageC;

float pageT;

float s1, s2, s3;

public:

string title;

float price;

int book()

{

int pageC;

}

int tape()

{

float pageT;

}

int sales()

{

float s1;

float s2;

float s3;

}

int getdata()

{

cout<<"enter title of book: ";

cin>>title;

cout<<"enter price of book: ";

cin>>price;

cout<<"enter number of pages: ";

cin>>pageC;

cout<<"enter time in minutes: ";

cin>>pageT;

cout<<"sales amount in first month: ";

cin>>s1;

cout<<"sales amount in second month: ";

cin>>s2;

cout<<"sales amount in third month: ";

cin>>s3;

}

int putdata()

{

cout<<"publications detail:"<<endl;

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

cout<<"price: "<<price<<" Rs"<<endl;

cout<<"number of pages: "<<pageC<<endl;

cout<<"playing time: "<<pageT<<" minutes"<<endl;

cout<<"sales amount in first three month:"<<endl;

cout<<"sales amount in first month: "<<s1<<"%"<<endl;

cout<<"sales amount in second month: "<<s2<<"%"<<endl;

cout<<"sales amount in third month: "<<s3<<"%"<<endl;

}

};

int main ()

{

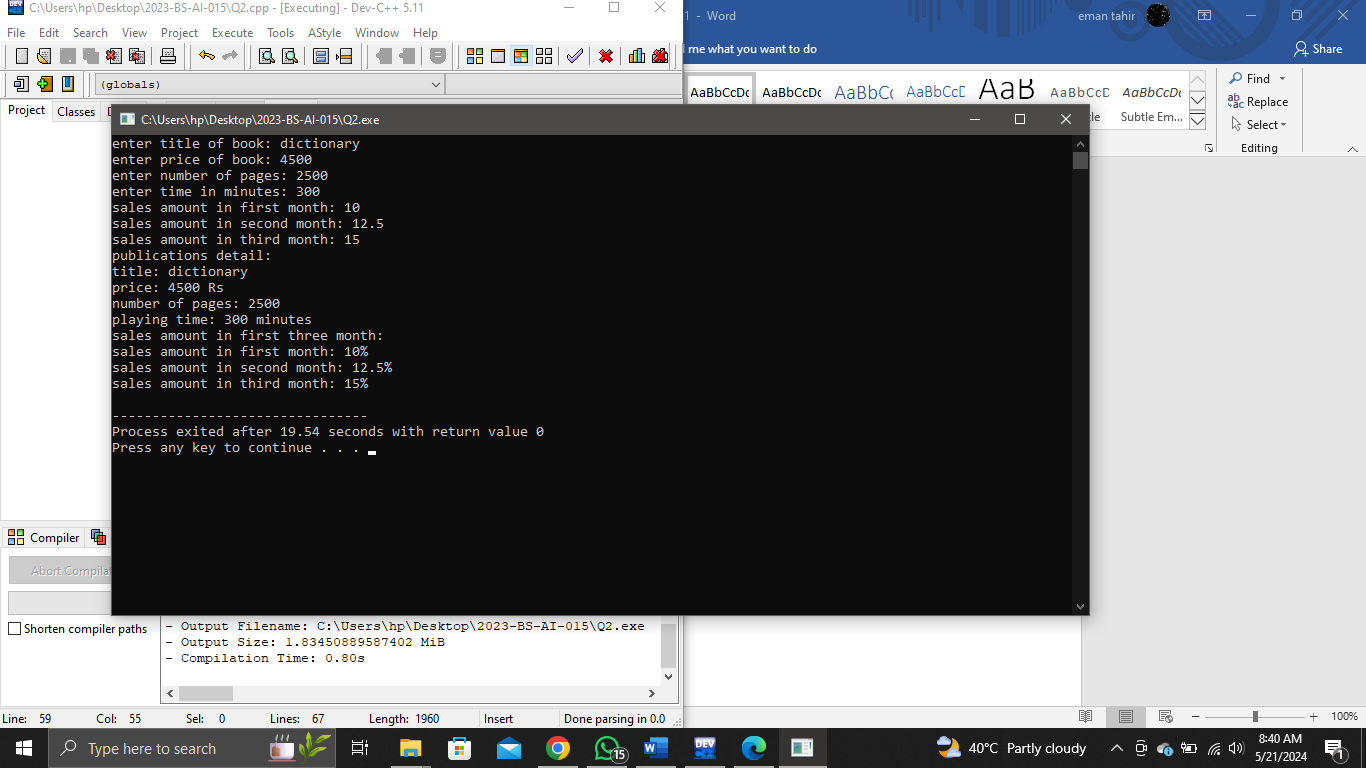
publication p;

p.getdata();

p.putdata();

}

Output:



Q3: Assume that the publisher in Question 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 publication. The disk class should incorporate the same member functions 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.

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question statement: Assume that the publisher in Question 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 publication. The disk class should incorporate the same member functions 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.

#include<iostream>

using namespace std;

class publication

{

private:

int pageC;

float pageT;

float s1, s2, s3;

string type;

public:

string title;

float price;

int book()

{

int pageC;

}

int tape()

{

float pageT;

}

int sales()

{

float s1;

float s2;

float s3;

}

int disk()

{

string type;

}

int getdata()

{

cout<<"enter title of book: ";

cin>>title;

cout<<"enter price of book: ";

cin>>price;

cout<<"enter number of pages: ";

cin>>pageC;

cout<<"enter time in minutes: ";

cin>>pageT;

cout<<"sales amount in first month: ";

cin>>s1;

cout<<"sales amount in second month: ";

cin>>s2;

cout<<"sales amount in third month: ";

cin>>s3;

cout<<"enter c for CD type and d for DVD type: ";

cin>>type;

}

int putdata()

{

cout<<"publications detail:"<<endl;

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

cout<<"price: "<<price<<" Rs"<<endl;

cout<<"number of pages: "<<pageC<<endl;

cout<<"playing time: "<<pageT<<" minutes"<<endl;

cout<<"sales amount in first three month:"<<endl;

cout<<"sales amount in first month: "<<s1<<"%"<<endl;

cout<<"sales amount in second month: "<<s2<<"%"<<endl;

cout<<"sales amount in third month: "<<s3<<"%"<<endl;

cout<<"the disk type u wanted is: "<<type<<"type"<<endl;

}

};

int main ()

{

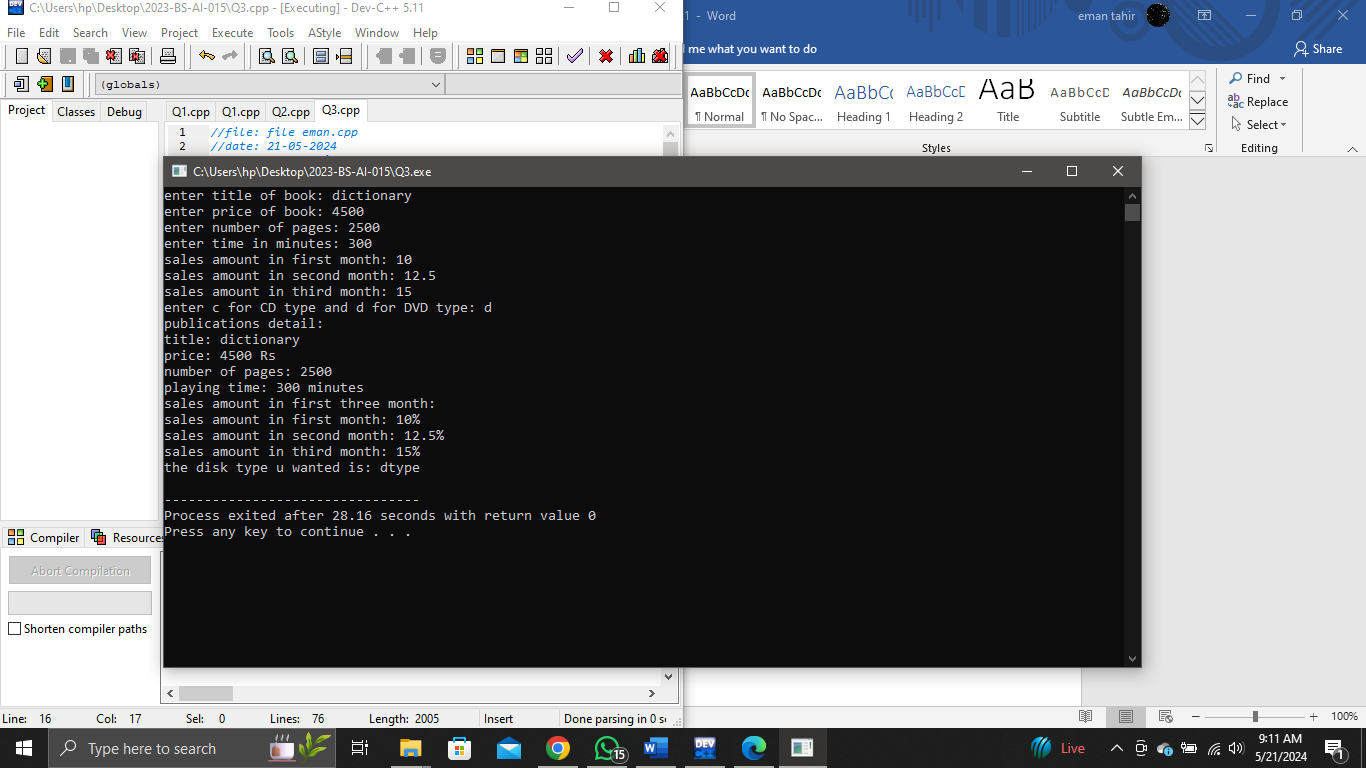
publication p;

p.getdata();

p.putdata();

}

Output:



Q4: 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 compensation, 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, scientist, 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 compensation and three new classes manager2, scientist2, and laborer2, and use multiple inheritance to derive these three classes from the original manager, scientist, and laborer classes and from compensation. This way none of the original classes needs to be modified

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question statement: 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 compensation, 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, scientist, 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 compensation and three new classes manager2, scientist2, and laborer2, and use multiple inheritance to derive these three classes from the original manager, scientist, and laborer classes and from compensation. This way none of the original classes needs to be modified

#include <iostream>

using namespace std;

class Employee {

protected:

int empID;

public:

Employee() {

empID=0; }

void setEmpID(int id)

{

empID = id;

}

int getEmpID() const

{

return empID;

}

virtual void display() const

{

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

}

};

class Employee2 : public Employee {

public:

enum Period { HOURLY, WEEKLY, MONTHLY };

private:

double compensation;

Period payPeriod;

public:

Employee2()

{

compensation=0.0;

payPeriod=HOURLY;

}

void setCompensation(double comp)

{

compensation = comp;

}

double getCompensation() const

{

return compensation;

}

void setPayPeriod(Period period)

{

payPeriod = period;

}

Period getPayPeriod() const

{

return payPeriod;

}

void display() const override {

Employee::display();

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

cout << "Pay Period: " << (payPeriod == HOURLY ? "Hourly" : payPeriod == WEEKLY ? "Weekly" : "Monthly") << endl;

}

};

class Manager : public Employee2 {

public:

void display() const override {

cout << "Manager" << endl;

Employee2::display();

}

};

class Scientist : public Employee2 {

public:

void display() const override {

cout << "Scientist" << endl;

Employee2::display();

}

};

class Laborer : public Employee2 {

public:

void display() const override {

cout << "Laborer" << endl;

Employee2::display();

}

};

int main() {

Manager m;

m.setEmpID(1);

m.setCompensation(6000.0);

m.setPayPeriod(Employee2::MONTHLY);

Scientist s;

s.setEmpID(2);

s.setCompensation(57000.0);

s.setPayPeriod(Employee2::WEEKLY);

Laborer l;

l.setEmpID(3);

l.setCompensation(8000.0);

l.setPayPeriod(Employee2::HOURLY);

m.display();

cout << endl;

s.display();

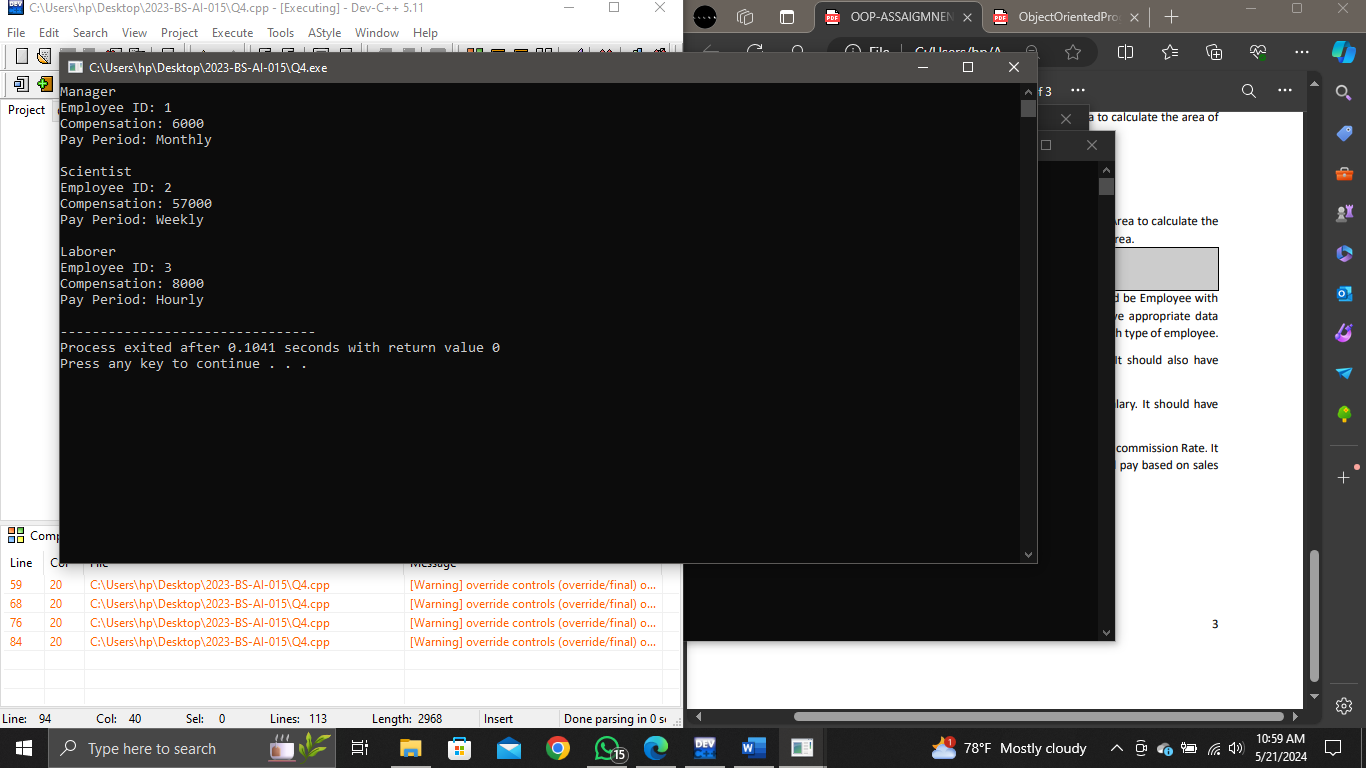
cout << endl;

l.display();

return 0;

}

Output:



Q5: 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:

Atributes: color (type std::string).

Methods: A constructor to initialize the color and a method printColor to display the color.

Circle Class:

Atributes: 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:

Atributes: 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.

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question 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:

//Atributes: color (type std::string).

//Methods: A constructor to initialize the color and a method printColor to display the color.

//Circle Class:

//Atributes: radius (type double).

//Methods: A constructor to initialize the color and radius, a method calculateArea to calculate the area of the circle (area = ? \* radius \* radius), and a method printArea to display the area.

//Rectangle Class:

//Atributes: 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>

using namespace std;

class Shape

{

private:

string color;

public:

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

int 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 3.14 \* radius \* radius;

}

int printArea() const {

cout << "Area of the circle: " << calculateArea() << endl;

}

};

class Rectangle : public Shape {

private:

double length;

double 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 the rectangle: " << calculateArea() << endl;

}

};

int main() {

Circle circle("yellow", 3.0);

circle.printColor();

circle.printArea();

Rectangle rectangle("green", 3.0, 7.0);

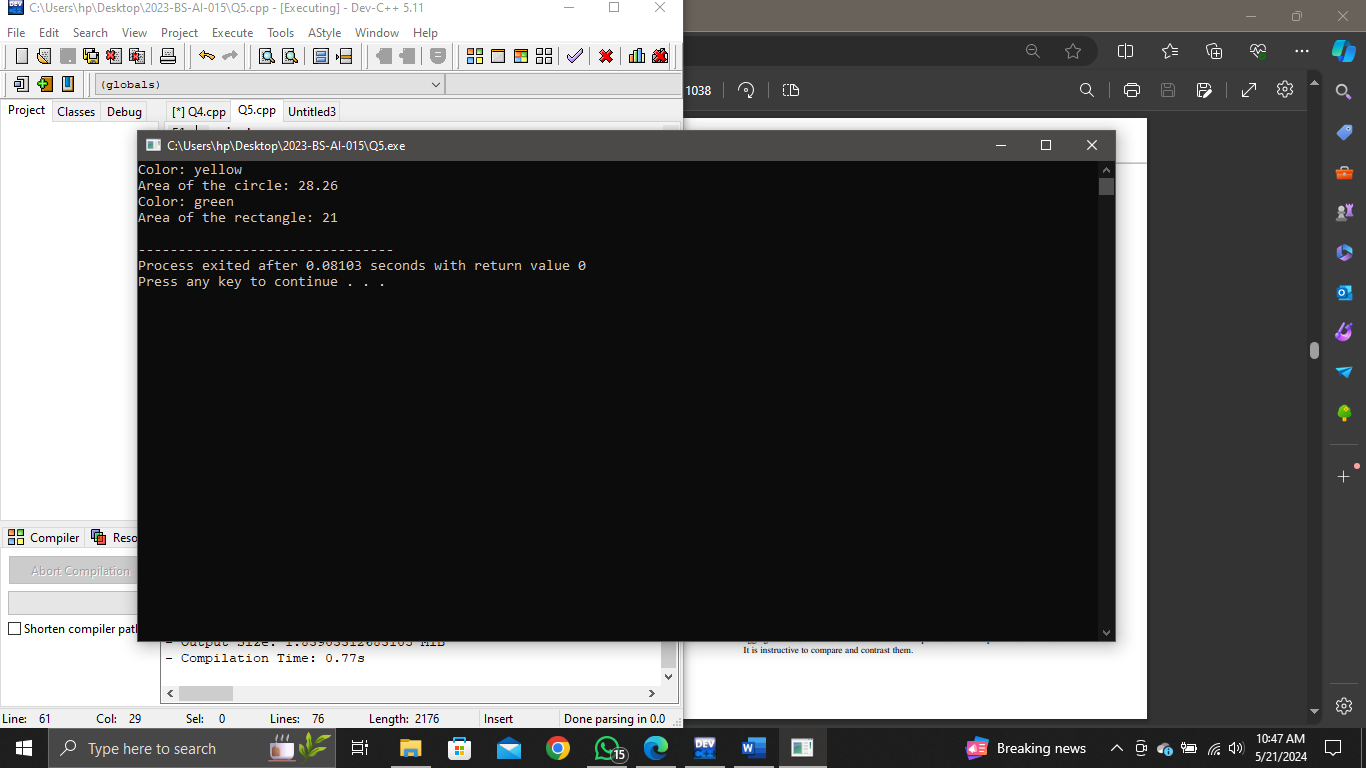
rectangle.printColor();

rectangle.printArea();

return 0;

}

Output:



Q6: 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 atributes 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.

Salaried Employee: 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.

Commission Employee: 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.

Solution:

Input: //file: file eman.cpp

//date: 21-05-2024

//name: eman tahir

//reg no. 2023-bs-ai-015

//question 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 atributes 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.

//Salaried Employee: 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.

//Commission Employee: 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>

using namespace std;

class Employee {

private:

int employeeID;

string name;

string department;

public:

Employee(int id, string n, string dept) : employeeID(id), name(n), department(dept) {}

int display()

const {

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

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

cout << "Department: " << department << endl;

}

};

class SEmployee : public Employee {

private:

double annualSalary;

public:

SEmployee(int id, string n, string dept, double salary)

: Employee(id, n, dept), annualSalary(salary) {}

double calMonthlyPay() const {

return annualSalary / 15.0;

}

void displaySalary() const {

cout << "Annual Salary: $" << annualSalary << endl;

}

};

class CommissionEmployee : public Employee {

private:

double sales;

double commissionRate;

public:

CommissionEmployee(int id, string n, string dept, double salesAmt, double rate)

: Employee(id, n, dept), sales(salesAmt), commissionRate(rate) {}

double calTotalPay() const {

return sales \* commissionRate;

}

void displayCommissionInfo() const {

cout << "Total Sales: $" << sales << endl;

cout << "Commission Rate: " << commissionRate << endl;

}

};

int main() {

SEmployee sEmp(1, "emp1", "Marketing", 30000.0);

sEmp.display();

sEmp.displaySalary();

cout << "Monthly Pay: $" << sEmp.calMonthlyPay() << endl;

CommissionEmployee commissionEmp(2, "emp2", "Sales", 80000.0, 0.07);

commissionEmp.display();

commissionEmp.displayCommissionInfo();

cout << "Total Pay: $" << commissionEmp.calTotalPay() << endl;

return 0;

}

Output:

