Lab 7 tasks

Hamza Farhan

24K-0576

Task 1

#include <iostream>

#include <string>

using namespace std;

class Account {

protected:

    int accountNumber;

    string accountHolderName, accountType;

    float balance;

    float \*transaction;

    int transactionCount = 0;

public:

    Account(string n, int num, string type, float bal) : accountNumber(num), accountHolderName(n), accountType(type), balance(bal), transaction(nullptr), transactionCount(0) {}

    virtual ~Account() {

        delete[] transaction;

    }

    void deposit(float amount) {

        if (amount > 0) {

            balance += amount;

            float \*temp = new float[transactionCount + 1];

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

                temp[i] = transaction[i];

            }

            temp[transactionCount] = amount;

            delete[] transaction;

            transaction = temp;

            transactionCount++;

            cout << "Deposited: " << amount << "\n";

        } else {

            cout << "Invalid deposit amount.\n";

        }

    }

    virtual void withdraw(float amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            float \*temp = new float[transactionCount + 1];

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

                temp[i] = transaction[i];

            }

            temp[transactionCount] = -amount;

            delete[] transaction;

            transaction = temp;

            transactionCount++;

            cout << "Withdrawn: " << amount << "\nRemaining balance: " << balance << "\n";

        } else {

            cout << "Invalid or insufficient funds.\n";

        }

    }

    void getAccountInfo() {

        cout << "account number: " << accountNumber << "\nholders name: " << accountHolderName << "\naccount type: " << accountType << "\nBalance: " << balance << "\n";

    }

    void printStatement() {

        cout << "transaction history for account " << accountNumber << ":\n";

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

            if (transaction[i] > 0) {

                cout << "deposit: "<< +transaction[i] << "\n";

            } else {

                cout << "withdrawal: " << -transaction[i] << "\n";

            }

        }

    }

    virtual float calculateInterest() {

        return 0.1f \* balance;

    }

};

class SavingsAccount : public Account {

    float interestRate;

    float minimumBalance;

public:

    SavingsAccount(string n, int num, float bal, float ir, float mb)

        : Account(n, num, "Savings", bal), interestRate(ir), minimumBalance(mb) {}

    float calculateInterest() override {

        if (balance >= minimumBalance) {

            return interestRate \* balance;

        }

        return 0;

    }

    void withdraw(float amount) override {

        if (balance - amount >= minimumBalance) {

            Account::withdraw(amount);

        } else {

            cout << "Cannot withdraw minimum balance required\n";

        }

    }

};

class CheckingAccount : public Account {

public:

    CheckingAccount(string n, int num, float bal) : Account(n, num, "Checking", bal) {}

    float calculateInterest() override {

        return 0.0f;

    }

};

class FixedDepositAccount : public Account {

    float fixedInterestRate;

    int maturityDate;

public:

    FixedDepositAccount(string n, int num, float bal, float ir, int md)

        : Account(n, num, "Fixed Deposit", bal), fixedInterestRate(ir), maturityDate(md) {}

    float calculateInterest() override {

        return fixedInterestRate \* balance;

    }

    void withdraw(float amount) override {

        cout << "Cant withdraw from fixed deposit before maturity\n";

    }

};

int main() {

    SavingsAccount s("hamza", 12345, 5000.0, 0.05, 1000.0f);

    CheckingAccount c("zaid", 67890, 3000.0);

    FixedDepositAccount f("rayyan", 11111, 10000.0, 0.07, 12);

    cout << "\nSavings Account\n";

    s.getAccountInfo();

    s.deposit(2000);

    s.withdraw(4000);

    cout << "Interest: " << s.calculateInterest() << "\n";

    s.getAccountInfo();

    s.printStatement();

    cout << "\nChecking Account\n";

    c.getAccountInfo();

    c.deposit(1000);

    c.withdraw(500);

    cout << "Interest: " << c.calculateInterest() << "\n";

    c.printStatement();

    cout << "\nFixed Deposit Account\n";

    f.getAccountInfo();

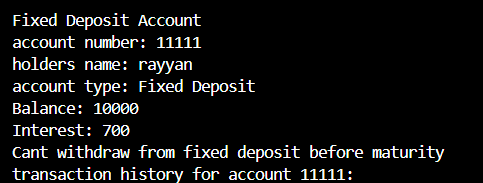
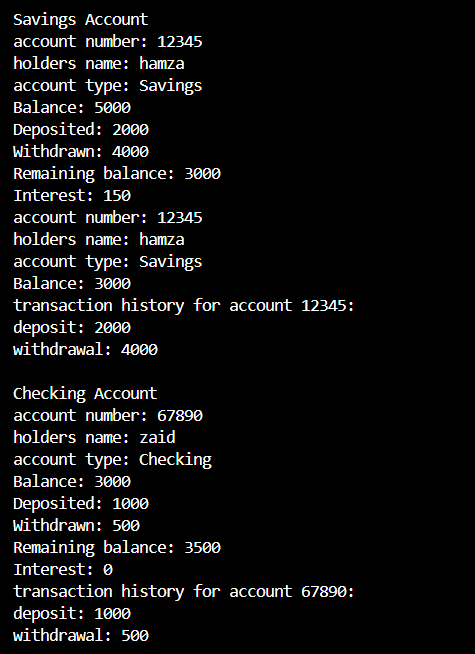
    cout << "Interest: " << f.calculateInterest() << "\n";

    f.withdraw(1000);

    f.printStatement();

}

Output



Task 2

#include <iostream>

#include <string>

using namespace std;

class Shape {

    protected:

    int position;

    string color;

    float thickness;

    public:

    Shape(int pos,string c,float thick):position(pos),color(c),thickness(thick){}

    virtual void draw(){}

    virtual float calculateArea(){

        return 2\*2;

    }

    virtual float calculatePerimeter(){

        return 2+2+2;

    }

};

class Circle:public Shape{

    private:

    float radius;

    int Cposition;

    public:

    Circle(int pos,string c,float thick,float r):Shape(pos,c,thick),Cposition(pos),radius(r){}

    void draw()override{

        cout << "Drawing Rectangle at (" << Cposition <<") with radius "<<radius<< " and color " << color << "\n";

    }

    float calculatePerimeter()override{

        return 2\*3.14\*radius;

    }

    float calculateArea()override{

        return 3.14\*radius\*radius;

    }

};

class Rectangle:public Shape{

    private:

    float width,height;

    int Lposition;

    public:

    Rectangle(int pos,string c,float thick,float w,float h):Shape(pos,c,thick),Lposition(pos),width(w),height(h){}

    void draw()override {

        cout << "Drawing Rectangle at (" << Lposition <<") with width " << width << " and height " << height

             << " and color " << color << "\n";

    }

    float calculatePerimeter()override{

        return 2\*width+2\*height;

    }

    float calculateArea()override{

        return width\*height;

    }

};

class Triangle:public Shape{

    private:

    float a,b,c;

    int Cposition;

    public:

    Triangle(int pos,string col,float thick,float a,float b,float c):Shape(pos,col,thick),Cposition(pos),a(a),b(b),c(c){}

    void draw()override{

        cout << "Drawing Rectangle at (" << Cposition <<") with base "<<a<<" height" <<b <<" adjacent"<<c<<" and color " << color << "\n";

    }

    float calculatePerimeter()override{

        return a+b+c;

    }

    float calculateArea()override{

        return 0.5\*a\*b;

    }

};

int main(){

    Circle c(10,"red",2,5);

    cout << "Area: " << c.calculateArea() << "\n";

    cout << "Perimeter: " <<c.calculatePerimeter() << "\n";

}

Output



Task 3

#include <iostream>

#include <string>

using namespace std;

class Currency{

    protected:

    float amount;

    int currencyCode;

    string currencySymbol;

    float exchangeRate;

    Currency(float amount ,int code,string symbol,float exchange):amount(amount),currencyCode(code),currencySymbol(symbol),exchangeRate(exchange){}

    public:

        virtual double convertToBase() {

            return amount \* exchangeRate;

        }

       virtual float convertTo(Currency &target){

        double baseAmount = convertToBase();

        return baseAmount \* target.exchangeRate;

       }

       virtual void displayCurrency(){

            cout<<"currency code: "<<currencyCode<<endl;

            cout<<"exchange rate: "<<exchangeRate<<"\nsymbol: "<<currencySymbol;

       }

};

class Dollar:public Currency{

    public:

    Dollar(double amt) : Currency(amt, 1122, "$", 1.0) {}

    void displayCurrency() override {

        cout << "$" << amount << " USD" << endl;

    }

};

class Euro:public Currency{

    public:

    Euro(double amt) : Currency(amt, 2211, "@", 0.9) {}

    void displayCurrency() override {

        cout << "@" << amount << " Euro" << endl;

    }

};

class Rupee:public Currency{

    public:

    Rupee(double amt) : Currency(amt, 1212, "R.", 270) {}

    void displayCurrency() override {

        cout << "R. " << amount << " Rupee" << endl;

    }

};

int main (){

    Dollar usd(100);

    Euro eur(0);

    Rupee pkr(0);

    cout << "Original Currencies:" << endl;

    usd.displayCurrency();

    cout << "\nConverting USD to EUR:" << endl;

    double eurAmount = usd.convertTo(eur);

    Euro convertedEuro(eurAmount);

    convertedEuro.displayCurrency();

    cout << "\nConverting USD to PKR:" << endl;

    double inrAmount = usd.convertTo(pkr);

    Rupee convertedRupee(inrAmount);

    convertedRupee.displayCurrency();

    return 0;

}

Output



Task 4

#include <iostream>

#include <string>

using namespace std;

class Person {

    protected:

    string name;

    string address, email;

    int phoneNum;

    public:

    Person(string n,int num,string add,string em):name(n),phoneNum(num),address(add),email(em){}

    virtual void displayInfo(){

        cout<<"name: "<<name<<"\nPhone Number: "<<phoneNum<<"\nAddress: "<<address<<"\nemail: "<<email<<endl;

    }

    string getName() const {

        return name;

    }

    void updateInfo(){

        int ch;

        cout<<"\nenter choice:\n";

        cout<<"press 1 to update name\npress 2 to update address\npress 3 to update number\npress 4 to update email\n";

        cin>>ch;

        string nName,Naddress,Nemail;

        int Nnum;

        switch (ch)

        {

        case 1:

            cout<<"enter new name: ";

            cin>>nName;

            name=nName;

            break;

        case 2:

            cout<<"enter new address: ";

            cin>>Naddress;

            address=Naddress;

            break;

        case 3:

            cout<<"enter new number: ";

            cin>>Nnum;

            phoneNum=Nnum;

            break;

        case 4:

            cout<<"enter new email: ";

            cin>>Nemail;

            email=Nemail;

            break;

            default:

            cout<<"invalid option\n";

            break;

        }

    }

};

class Student:public Person{

    protected:

    float gpa;

    string coursesEnrolled[2];

    int enrollmentYear;

    public:

    Student(string n,int num,string add,string em,float gpa,string c1,string c2,int year):

    Person(n,num, add, em),gpa(gpa),enrollmentYear(year)  {

        coursesEnrolled[0]=c1;

        coursesEnrolled[1]=c2;

    }

    void displayInfo()override{

        cout<<"name: "<<name<<"\nPhone Number: "<<phoneNum<<"\nAddress: "<<address<<"\nemail: "<<email<<endl;

        cout<<"gpa: "<<gpa<<"\nenrollment Year: "<<enrollmentYear<<endl;

        cout<<"courses:\n";

        cout<<coursesEnrolled[0]<<"\n"<<coursesEnrolled[1]<<endl;

    }

};

class Professor:public Person{

    protected:

    float salary;

    string coursesTaught[2],department;

    public:

    Professor(string n,int num,string add,string em,float salary,string c1,string c2,string dep):

    Person(n,num, add, em),salary(salary),department(dep){

        coursesTaught[0]=c1;

        coursesTaught[1]=c2;

    }

    void displayInfo()override{

        cout<<"name: "<<name<<"\nPhone Number: "<<phoneNum<<"\nAddress: "<<address<<"\nemail: "<<email<<endl;

        cout<<"salary: "<<salary<<"department: "<<department<< endl;

        cout<<"courses taught:\n";

        cout<<coursesTaught[0]<<"\n"<<coursesTaught[1]<<endl;

    }

};

class Staff:public Person{

    protected:

    float salary;

    string postion,department;

    public:

    Staff(string n,int num,string add,string em,float salary,string dep,string pos):

    Person(n,num, add, em),salary(salary),department(dep),postion(pos){

    }

    void displayInfo()override{

        cout<<"name: "<<name<<"\nPhone Number: "<<phoneNum<<"\nAddress: "<<address<<"\nemail: "<<email<<endl;

        cout<<"salary: "<<salary<<"\ndepartment: "<<department<<"\nposition: "<<postion<<endl;

    }

};

class Course

{

    private:

    int courseId,credits;

    string courseName,instructor;

    string schedule;

    public:

    Course(int id, string name, string instr, int cred, string sched)

    :courseId(id), courseName(name), instructor(instr), credits(cred), schedule(sched){}

    void displayCourse() {

        cout << "Course ID: " << courseId << "\nCourse Name: " << courseName<< "\nInstructor: " << instructor << "\nCredits: " << credits

             << "\nSchedule: " << schedule << endl;

    }

    void registerStudent(Student &s){

        cout << s.getName() << " has been registered for "<<courseName<<endl;

    }

};

int main (){

    Student s1("hamza",92335,"street 2, house no. 6, DHA phase 8","hamza123@gmail.com",3.8,"OOP","MVC",2024);

    Professor p1("Dr. Smith", 12345, "University Road", "smith@uni.edu", 95000.50, "AI", "Data Science", "CS Department");

    Staff staff1("John Doe", 67890, "Admin Block", "john.doe@uni.edu", 50000.75, "Admin", "Manager");

    Course c1(101, "Object-Oriented Programming", "Dr. Smith", 3, "MWF 10:00 AM - 11:30 AM");

    s1.displayInfo();

    s1.updateInfo();

    s1.displayInfo();

    p1.displayInfo();

    staff1.displayInfo();

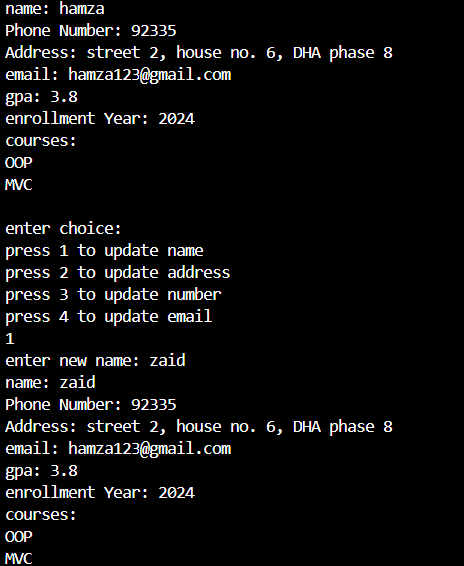
    c1.displayCourse();

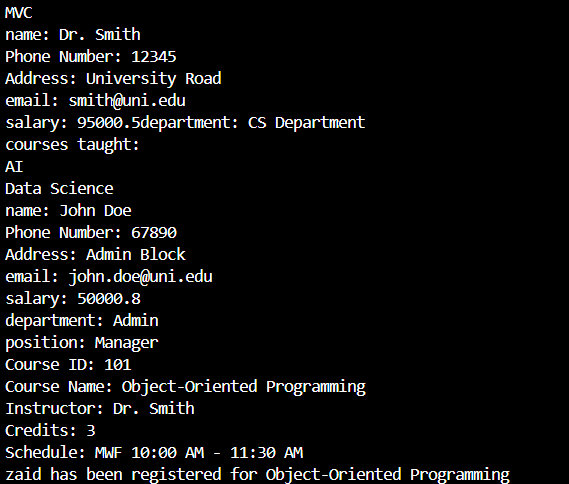
    c1.registerStudent(s1);

    return 0;

}

Output





Task 5

#include <iostream>

#include <string>

using namespace std;

class Media{

    protected:

    string title,publisher;

    int publicationDate,uniqueID;

    public:

    Media(string t,string pub,int date,int id):title(t),publisher(pub),publicationDate(date),uniqueID(id){}

    virtual void displayInfo(){

        cout<<"title: "<<title<<"\npublisher: "<<publisher<<"\npublication date: "<<publicationDate<<"\nID: "<<uniqueID<<endl;

    }

    void checkout(){

        cout<<"checked out\n";

    }

    void returnItem(){

        cout<<"item returned\n";

    }

    string getTitle(){

        return title;

    }

    ~Media(){}

};

class Book:public Media{

    private:

    string author;

    int ISBN,numberOfPages;

    public:

    Book(string t,string pub,int date,int id,string auth,int isbn,int pages):Media(t, pub, date, id),ISBN(isbn),author(auth),

    numberOfPages(pages){}

    void displayInfo()override{

       Media::displayInfo();

        cout<<"author: "<<author<<"\nISBN: "<<ISBN<<"\npages: "<<numberOfPages<<endl;

    }

    void checkout(){

        cout<<"book "<<getTitle()<<" checked out\n";

    }

    ~Book(){}

};

class DVD:public Media{

    private:

    string director;

    float duration,rating;

    public:

    DVD(string t,string pub,int date,int id,string dir,float dur,float r):Media(t, pub, date, id),duration(dur),director(dir),

    rating(r){}

    void displayInfo()override{

        Media::displayInfo();

        cout<<"director: "<<director<<"\nduration: "<<duration<<"\nrating: "<<rating<<endl;

    }

    void checkout(){

        cout<<"DVD "<<getTitle()<<" checked out\n";

    }

    ~DVD(){}

};

class CD:public Media{

    private:

    string artist,genre;

    int numberOfTracks;

    public:

    CD(string t,string pub,int date,int id,string art,string gen,int tracks):Media(t, pub, date, id),artist(art),genre(gen),numberOfTracks(tracks){}

    void displayInfo()override{

        Media::displayInfo();

        cout<<"artist: "<<artist<<"\ngenre: "<<genre<<"\nnumber of tracks: "<<numberOfTracks<<endl;

    }

    void checkout(){

        cout<<"CD "<<getTitle()<<" checked out\n";

    }

    ~CD(){}

};

class Magzine:public Media{

    private:

    int fee;

    public:

    Magzine(string t,string pub,int date,int id,int f):Media(t, pub, date, id),fee(f){}

    void displayInfo()override{

        Media::displayInfo();

        cout<<"fee: "<<fee<<endl;

    }

    void checkout(){

        cout<<"magzine "<<getTitle()<<" checked out\n";

    }

    ~Magzine(){}

};

int main (){

    Magzine m1("daily","vogue",13062024,2211,3000);

    Book b1("harry potter","matt damond",14072023,1122,"JK rowling",522450,556);

    DVD d1("deadpool","rayn renolds",15022022,5566,"sam raimi",2.5,8.9);

    CD c1("Aadat","farooq sattar",14072021,3322,"atif aslam","pop",52);

    c1.displayInfo();

    cout<<endl;

    d1.displayInfo();

    cout<<endl;

    b1.displayInfo();

    cout<<endl;

    m1.displayInfo();

    cout<<endl;

    c1.checkout();

    cout<<endl;

    c1.returnItem();

    cout<<endl;

}

Output

