ICT OOP TASKS

QUESTION 2

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Account {

protected:

    string accountNumber;

    string accountHolderName;

    double balance;

    vector<string> transactionHistory;

public:

    Account(string accNum, string accHolder, double bal)

        : accountNumber(accNum), accountHolderName(accHolder), balance(bal) {}

    virtual void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            transactionHistory.push\_back("Deposited: " + to\_string(amount));

        } else {

            cout << "Invalid deposit amount!" << endl;

        }

    }

    virtual void withdraw(double amount) {

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

            balance -= amount;

            transactionHistory.push\_back("Withdrawn: " + to\_string(amount));

        } else {

            cout << "Insufficient funds or invalid withdrawal amount!" << endl;

        }

    }

    void displayTransactionHistory() {

        cout << "Transaction History for " << accountHolderName << " (Account: " << accountNumber << "):" << endl;

        for (const string &trans : transactionHistory) {

            cout << trans << endl;

        }

        cout << "Current Balance: " << balance << endl;

    }

    double getBalance() {

        return balance;

    }

    string getAccountNumber() {

        return accountNumber;

    }

    string getAccountHolderName() {

        return accountHolderName;

    }

};

class SavingsAccount : public Account {

private:

    double interestRate;

public:

    SavingsAccount(string accNum, string accHolder, double bal, double rate)

        : Account(accNum, accHolder, bal), interestRate(rate) {}

    void calculateInterest() {

        double interest = (balance \* interestRate) / 100;

        balance += interest;

        transactionHistory.push\_back("Interest Added: " + to\_string(interest));

        cout << "Interest earned: " << interest << endl;

    }

};

class CurrentAccount : public Account {

private:

    double overdraftLimit;

public:

    CurrentAccount(string accNum, string accHolder, double bal, double overdraft)

        : Account(accNum, accHolder, bal), overdraftLimit(overdraft) {}

    void withdraw(double amount) override {

        if (amount > 0 && (balance - amount) >= -overdraftLimit) {

            balance -= amount;

            transactionHistory.push\_back("Withdrawn: " + to\_string(amount));

        } else {

            cout << "Withdrawal exceeds overdraft limit!" << endl;

        }

    }

};

int main() {

    SavingsAccount savingsAcc("2345", "ALI", 5000.0, 4.0);

    CurrentAccount currentAcc("452345", "Jane AMIR", 400.0, 150.0);

    savingsAcc.deposit(170.0);

    savingsAcc.withdraw(159.0);

    savingsAcc.calculateInterest();

    currentAcc.deposit(700.0);

    currentAcc.withdraw(500.0);

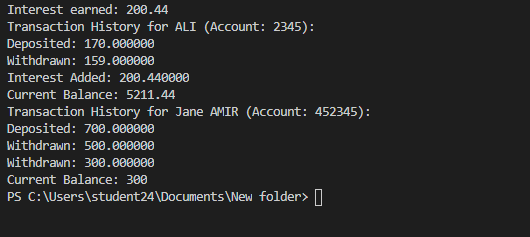
    currentAcc.withdraw(300.0);

    savingsAcc.displayTransactionHistory();

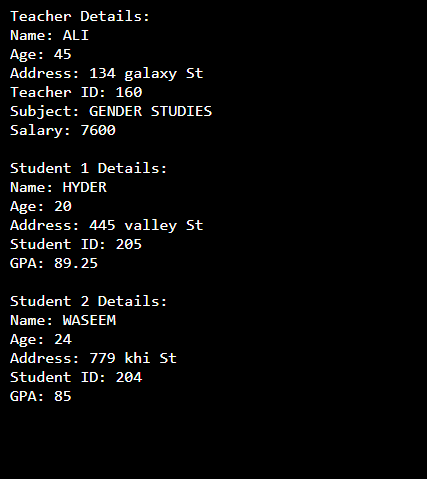
    currentAcc.displayTransactionHistory();

    return 0;

}



QUESTION 1

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Payable {

public:

virtual double calculateSalary() const = 0; // Pure virtual function

};

class Person {

protected:

string name;

int age;

string address;

public:

Person(string n, int a, string addr) : name(n), age(a), address(addr) {}

virtual void displayDetails() const {

cout << "Name: " << name << "\n";

cout << "Age: " << age << "\n";

cout << "Address: " << address << "\n";

}

};

class Student : public Person {

private:

int studentID;

vector<double> grades;

public:

Student(string n, int a, string addr, int id)

: Person(n, a, addr), studentID(id) {}

void addGrade(double grade) {

grades.push\_back(grade);

}

double calculateGPA() const {

if (grades.empty()) return 0.0;

double total = 0.0;

for (double grade : grades) {

total += grade;

}

return total / grades.size();

}

void displayDetails() const override {

Person::displayDetails();

cout << "Student ID: " << studentID << "\n";

cout << "GPA: " << calculateGPA() << "\n";

}

};

class Teacher : public Person, public Payable {

private:

int teacherID;

string subject;

int classesTaught;

double ratePerClass;

public:

Teacher(string n, int a, string addr, int id, string subj, int classes, double rate)

: Person(n, a, addr), teacherID(id), subject(subj), classesTaught(classes), ratePerClass(rate) {}

void assignGrade(Student& student, double grade) {

student.addGrade(grade);

}

double calculateSalary() const override {

return classesTaught \* ratePerClass;

}

void displayDetails() const override {

Person::displayDetails();

cout << "Teacher ID: " << teacherID << "\n";

cout << "Subject: " << subject << "\n";

cout << "Salary: " << calculateSalary() << "\n";

}

};

int main() {

Teacher teacher("ALI", 45, "134 galaxy St", 160, "GENDER STUDIES", 19, 400.0);

Student student1("HYDER", 20, "445 valley St", 205);

Student student2("WASEEM", 24, "779 khi St", 204);

teacher.assignGrade(student1, 98.5);

teacher.assignGrade(student1, 80.0);

teacher.assignGrade(student2, 78.0);

teacher.assignGrade(student2, 92.0);

cout << "Teacher Details:\n";

teacher.displayDetails();

SASS

cout << "\nStudent 1 Details:\n";

student1.displayDetails();

cout << "\nStudent 2 Details:\n";

student2.displayDetails();

return 0;

}