TASK 01

#include <iostream>

using namespace std;

class Wallet {

private:

string ownerName;

double totalBalance;

string transactionHistory[100];

int transactionCount;

public:

Wallet(string name, double balance) {

ownerName = name;

totalBalance = balance;

transactionCount = 0;

transactionHistory[transactionCount++] = "Wallet started with balance: " + to\_string(balance);

}

void addMoney(double amount) {

if (amount > 0) {

totalBalance += amount;

transactionHistory[transactionCount++] = "Added: " + to\_string(amount) + " | Balance: " + to\_string(totalBalance);

cout << "Added " << amount << ". New balance: " << totalBalance << endl;

} else {

cout << "Invalid amount." << endl;

}

}

void spendMoney(double amount) {

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

totalBalance -= amount;

transactionHistory[transactionCount++] = "Spent: " + to\_string(amount) + " | Balance: " + to\_string(totalBalance);

cout << "Spent " << amount << ". Remaining balance: " << totalBalance << endl;

if (totalBalance < 100) {

cout << "Warning: Low balance! Only " << totalBalance << " left." << endl;

}

} else {

cout << "Not enough balance or invalid amount." << endl;

}

}

void displayTransactionHistory() {

cout << "\nTransaction History for " << ownerName << ":\n";

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

cout << transactionHistory[i] << endl;

}

}

void displayBalance() {

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

}

};

int main() {

Wallet aliWallet("ali", 500);

aliWallet.addMoney(200);

aliWallet.spendMoney(150);

aliWallet.spendMoney(600);

aliWallet.spendMoney(50);

aliWallet.displayTransactionHistory();

aliWallet.displayBalance();

return 0;

}

TASK 02

#include <iostream>

using namespace std;

class FitnessTracker {

private:

string userName;

int dailyStepGoal;

int stepsTaken;

double caloriesBurned;

public:

FitnessTracker(string name, int stepGoal) {

userName = name;

dailyStepGoal = stepGoal;

stepsTaken = 0;

caloriesBurned = 0;

}

void logSteps(int steps) {

if (steps > 0) {

stepsTaken += steps;

caloriesBurned += steps \* 0.04;

cout << steps << " steps logged. Total steps today: " << stepsTaken << endl;

} else {

cout << "Invalid step count." << endl;

}

}

void displayProgress() {

cout << "\nFitness Progress for " << userName << ":\n";

cout << "Steps Taken: " << stepsTaken << "/" << dailyStepGoal << endl;

cout << "Calories Burned: " << caloriesBurned << " kcal" << endl;

if (stepsTaken >= dailyStepGoal) {

cout << "Great job! You've reached your step goal for today!\n";

} else {

cout << "Keep going! " << (dailyStepGoal - stepsTaken) << " more steps to reach your goal.\n";

}

}

};

int main() {

FitnessTracker ahmed("ahmed", 10000);

ahmed.logSteps(3000);

ahmed.logSteps(4500);

ahmed.logSteps(2800);

ahmed.displayProgress();

return 0;

}

TASK 03

#include <iostream>

using namespace std;

class Library {

private:

string bookList[100];

bool availableBooks[100];

int totalBooks;

public:

Library() {

totalBooks = 0;

}

void addBook(string bookName) {

if (totalBooks < 100) {

bookList[totalBooks] = bookName;

availableBooks[totalBooks] = true;

totalBooks++;

cout << "Book added: " << bookName << endl;

} else {

cout << "Library is full!" << endl;

}

}

void lendBook(string bookName) {

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

if (bookList[i] == bookName && availableBooks[i]) {

availableBooks[i] = false;

cout << "Book borrowed: " << bookName << endl;

return;

}

}

cout << "Book not available." << endl;

}

void returnBook(string bookName) {

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

if (bookList[i] == bookName && !availableBooks[i]) {

availableBooks[i] = true;

cout << "Book returned: " << bookName << endl;

return;

}

}

cout << "Invalid return. Book wasn't borrowed." << endl;

}

void displayBooks() {

cout << "\nLibrary Books:\n";

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

cout << bookList[i] << (availableBooks[i] ? " (Available)" : " (Borrowed)") << endl;

}

}

};

int main() {

Library hamzaLibrary;

hamzaLibrary.addBook("C++ Programming");

hamzaLibrary.addBook("Data Structures");

hamzaLibrary.addBook("Operating Systems");

hamzaLibrary.displayBooks();

hamzaLibrary.lendBook("Data Structures");

hamzaLibrary.displayBooks();

hamzaLibrary.returnBook("Data Structures");

hamzaLibrary.displayBooks();

return 0;

}

TASK 04

#include <iostream>

using namespace std;

class Car {

private:

string brand;

string model;

double fuelCapacity;

double currentFuelLevel;

public:

Car(string b, string m, double capacity) {

brand = b;

model = m;

fuelCapacity = capacity;

currentFuelLevel = capacity;

}

void drive(double distance) {

double fuelConsumed = distance \* 0.1;

if (fuelConsumed <= currentFuelLevel) {

currentFuelLevel -= fuelConsumed;

cout << "Drove " << distance << " km. Fuel left: " << currentFuelLevel << " liters.\n";

if (currentFuelLevel < 5) {

cout << "Warning: Low fuel! Please refuel soon.\n";

}

} else {

cout << "Not enough fuel to drive " << distance << " km.\n";

}

}

void refuel(double amount) {

if (amount > 0 && (currentFuelLevel + amount) <= fuelCapacity) {

currentFuelLevel += amount;

cout << "Refueled " << amount << " liters. Current fuel: " << currentFuelLevel << " liters.\n";

} else {

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

}

}

void checkFuelStatus() {

cout << "Current fuel level: " << currentFuelLevel << " liters.\n";

if (currentFuelLevel < 5) {

cout << "Warning: Low fuel! Consider refueling.\n";

}

}

};

int main() {

Car aliCar("Honda", "Accord", 50);

aliCar.drive(300);

aliCar.checkFuelStatus();

aliCar.refuel(10);

aliCar.drive(150);

aliCar.checkFuelStatus();

return 0;

}

TASK 05

#include <iostream>

using namespace std;

class MusicPlayer

{

private:

string playlist[100];

int currentlyPlayingIndex;

public:

MusicPlayer()

{

currentlyPlayingIndex = -1;

}

void addSong(string songName)

{

for (int i = 0; i < 100; i++)

{

if (playlist[i].empty())

{

playlist[i] = songName;

cout << "Added: " << songName << " to playlist.\n";

return;

}

}

cout << "Playlist is full!\n";

}

void removeSong(string songName)

{

for (int i = 0; i < 100; i++)

{

if (playlist[i] == songName)

{

playlist[i] = "";

if (currentlyPlayingIndex == i)

currentlyPlayingIndex = -1;

cout << "Removed: " << songName << " from playlist.\n";

return;

}

}

cout << "Song not found in playlist.\n";

}

void playSong(string songName)

{

for (int i = 0; i < 100; i++)

{

if (playlist[i] == songName)

{

currentlyPlayingIndex = i;

cout << "Now playing: " << songName << endl;

return;

}

}

cout << "Song not found in playlist.\n";

}

void displayPlaylist()

{

cout << "\nPlaylist:\n";

bool empty = true;

for (int i = 0; i < 100; i++)

{

if (!playlist[i].empty())

{

cout << playlist[i];

if (i == currentlyPlayingIndex)

cout << " (Now Playing)";

cout << endl;

empty = false;

}

}

if (empty)

cout << "No songs in playlist.\n";

}

};

int main()

{

MusicPlayer aliPlayer;

aliPlayer.addSong("Shape of You");

aliPlayer.addSong("Blinding Lights");

aliPlayer.addSong("Senorita");

aliPlayer.displayPlaylist();

aliPlayer.playSong("Blinding Lights");

aliPlayer.removeSong("Senorita");

aliPlayer.displayPlaylist();

return 0;

}