A logo with blue and red text

AI-generated content may be incorrect.

**WORKSHEET 3**

**Submitted By: Sandhya Aryal**

**Student ID: 24000860**

**Cyber Security and Digital Forensics**

1. **Create a Time class to store hours and minutes. Implement:**
   1. **Overload the + operator to add two Time objects**
   2. **Overload the > operator to compare two Time objects**
   3. **Handle invalid time (>24 hours or >60 minutes) by throwing a custom exception**

#include <iostream>

using namespace std;

class Time

{

private:

int hours;

int minutes;

public:

Time(int h = 0, int m = 0)

{

hours = h;

minutes = m;

validate();

}

void inputTime()

{

cout << "Enter the time in hours (0 - 24): ";

cin >> hours;

cout << "Enter the time in minutes (0 - 60): ";

cin >> minutes;

validate();

}

void displayTime()

{

cout << hours << " hours and " << minutes << " minutes" << endl;

}

void validate()

{

if (hours < 0 || hours > 24 || minutes < 0 || minutes >= 60)

{

cout << "Error: Invalid time! Hours must be <= 24 and Minutes must be <= 60." << endl;

exit(1);

}

}

Time operator+(Time t)

{

Time temp;

temp.minutes = minutes + t.minutes;

temp.hours = hours + t.hours;

if (temp.minutes >= 60)

{

temp.minutes -= 60;

temp.hours++;

}

return temp;

}

bool operator>(Time t)

{

if (hours > t.hours)

{

return true;

}

else if (hours == t.hours && minutes > t.minutes)

{

return true;

}

else

{

return false;

}

}

};

int main()

{

Time t1, t2, sum;

cout << "Enter first time:" << endl;

t1.inputTime();

cout << "Enter second time:" << endl;

t2.inputTime();

cout << "\nFirst Time: ";

t1.displayTime();

cout << "Second Time: ";

t2.displayTime();

sum = t1 + t2;

cout << "\nSum of times: ";

sum.displayTime();

if (t1 > t2)

{

cout << "First time is greater than second time." << endl;

}

else

{

cout << "Second time is greater than or equal to first time." << endl;

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Task 2: 70 marks**

1. **Create a base class Vehicle and two derived classes Car and Bike:**
   1. **Vehicle has registration number and color**
   2. **Car adds number of seats**
   3. **Bike adds engine capacity**
   4. **Each class should have its own method to write its details to a file**
   5. **Include proper inheritance and method overriding**

#include <iostream>

#include <fstream>

using namespace std;

class Vehicle {

protected:

string registrationNumber;

string color;

public:

Vehicle(string regNum = "", string col = "") : registrationNumber(regNum), color(col) {}

virtual void writeToFile() {

ofstream outFile("vehicle\_details.txt", ios::app);

if (outFile.is\_open()) {

outFile << "Vehicle Registration Number: " << registrationNumber << endl;

outFile << "Vehicle Color: " << color << endl;

} else {

cout << "Error opening file!" << endl;

}

outFile.close();

}

virtual void display() {

cout << "Vehicle Registration Number: " << registrationNumber << endl;

cout << "Vehicle Color: " << color << endl;

}

void inputVehicleDetails() {

cout << "Enter Vehicle Registration Number: ";

cin >> registrationNumber;

cout << "Enter Vehicle Color: ";

cin >> color;

}

};

class Car : public Vehicle {

private:

int numberOfSeats;

public:

Car(string regNum = "", string col = "", int seats = 0) : Vehicle(regNum, col), numberOfSeats(seats) {}

void writeToFile() override {

ofstream outFile("vehicle\_details.txt", ios::app);

if (outFile.is\_open()) {

outFile << "\n--- Car Details ---" << endl;

outFile << "Registration Number: " << registrationNumber << endl;

outFile << "Color: " << color << endl;

outFile << "Number of Seats: " << numberOfSeats << endl;

} else {

cout << "Error opening file!" << endl;

}

outFile.close();

}

void display() override {

Vehicle::display();

cout << "Number of Seats: " << numberOfSeats << endl;

}

void inputCarDetails() {

inputVehicleDetails();

cout << "Enter Number of Seats: ";

cin >> numberOfSeats;

}

};

class Bike : public Vehicle {

private:

int engineCapacity;

public:

Bike(string regNum = "", string col = "", int capacity = 0) : Vehicle(regNum, col), engineCapacity(capacity) {}

void writeToFile() override {

ofstream outFile("vehicle\_details.txt", ios::app);

if (outFile.is\_open()) {

outFile << "\n--- Bike Details ---" << endl;

outFile << "Registration Number: " << registrationNumber << endl;

outFile << "Color: " << color << endl;

outFile << "Engine Capacity: " << engineCapacity << " cc" << endl;

} else {

cout << "Error opening file!" << endl;

}

outFile.close();

}

void display() override {

Vehicle::display();

cout << "Engine Capacity: " << engineCapacity << " cc" << endl;

}

void inputBikeDetails() {

inputVehicleDetails();

cout << "Enter Engine Capacity (in cc): ";

cin >> engineCapacity;

}

};

void displayVehicleDetailsFromFile() {

ifstream inFile("vehicle\_details.txt");

if (inFile.is\_open()) {

string line;

while (getline(inFile, line)) {

cout << line << endl;

}

inFile.close();

} else {

cout << "Error opening file!" << endl;

}

}

int main() {

int choice;

char saveOption;

while (true) {

cout << "\n--- Vehicle Management System ---\n";

cout << "1. Add Vehicle Details\n";

cout << "2. Show Vehicle Details\n";

cout << "3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1) {

int vehicleType;

cout << "Enter the type of vehicle (1 for Car, 2 for Bike): ";

cin >> vehicleType;

if (vehicleType == 1) {

Car car;

car.inputCarDetails();

car.display();

cout << "Do you want to save this vehicle's details to a file? (y/n): ";

cin >> saveOption;

if (saveOption == 'y' || saveOption == 'Y') {

car.writeToFile();

cout << "\nDetails have been saved to 'vehicle\_details.txt'." << endl;

} else {

cout << "\nVehicle details were not saved." << endl;

}

} else if (vehicleType == 2) {

Bike bike;

bike.inputBikeDetails();

bike.display();

cout << "Do you want to save this vehicle's details to a file? (y/n): ";

cin >> saveOption;

if (saveOption == 'y' || saveOption == 'Y') {

bike.writeToFile();

cout << "\nDetails have been saved to 'vehicle\_details.txt'." << endl;

} else {

cout << "\nVehicle details were not saved." << endl;

}

} else {

cout << "Invalid vehicle type!" << endl;

}

} else if (choice == 2) {

cout << "\nDisplaying vehicle details from file:\n";

displayVehicleDetailsFromFile();

} else if (choice == 3) {

cout << "Exiting the Vehicle Management System." << endl;

break;

} else {

cout << "Your Choice is Invalid! Please try again." << endl;

}

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Create a program that:**
   1. **Reads student records (roll, name, marks) from a text file**
   2. **Throws an exception if marks are not between 0 and 100**
   3. **Allows adding new records with proper validation**
   4. **Saves modified records back to file**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Student

{

private:

string roll;

string name;

int marks;

public:

Student(string r = "", string n = "", int m = 0) : roll(r), name(n), marks(m) {}

int getMarks() const

{

return marks;

}

void setMarks(int m)

{

if (m >= 0 && m <= 100)

{

marks = m;

}

else

{

cout << "Error: Marks should be between 0 and 100." << endl;

marks = -1;

}

}

void setDetails()

{

cout << "Enter student roll: ";

cin >> roll;

cin.ignore();

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter student marks (0-100): ";

int m;

while (true)

{

cin >> m;

if (m >= 0 && m <= 100)

{

setMarks(m);

break;

}

else

{

cout << "Error: Marks should be between 0 and 100. Please enter again: ";

}

}

}

void displayDetails() const

{

cout << "Roll: " << roll << ", Name: " << name << ", Marks: " << marks << endl;

}

void saveToFile() const

{

ofstream outFile("student\_records.txt", ios::app);

if (outFile.is\_open())

{

outFile << roll << " " << name << " " << marks << endl;

outFile.close();

}

else

{

cout << "Error opening file!" << endl;

}

}

static void readFromFile()

{

ifstream inFile("student\_records.txt");

string roll, name;

int marks;

if (inFile.is\_open())

{

while (inFile >> roll >> name >> marks)

{

cout << "Roll: " << roll << ", Name: " << name << ", Marks: " << marks << endl;

}

inFile.close();

}

else

{

cout << "Unable to open file!" << endl;

}

}

};

int main()

{

int choice;

while (true)

{

cout << "\n--- Student Record System ---\n";

cout << "1. View Student Records\n";

cout << "2. Add New Student Record\n";

cout << "3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1)

{

Student::readFromFile();

}

else if (choice == 2)

{

Student student;

student.setDetails();

if (student.getMarks() >= 0 && student.getMarks() <= 100)

{

student.saveToFile();

cout << "Record added successfully!" << endl;

}

else

{

cout << "Failed to add record due to invalid marks." << endl;

}

}

else if (choice == 3)

{

cout << "Exiting the program!" << endl;

break;

}

else

{

cout << "Invalid choice! Please enter 1, 2, or 3." << endl;

}

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**