OOP

PROJECT INTERMEDIATE REPORT

GARAGE MANAGEMENT SYSTEM

Team Members:

Ahmad Waleed Akhtar	BSCE22003
Muhammad Arham	BSCE22007
Hadia Ahmad	BSCE22017

Task Done:

- Filing
- Operator Overloading
- Inheritance
- Composition
- Polymorphism
- Singleton

Task Left:

- Association
- Aggregation
- Template Class

Class’s created:

LogIn.h:

```
#include <iostream>
#include <windows.h>
using namespace std;

class LogIn{
    static LogIn* ptr;
    string userName;
    string password;
    string matchUser;
    string matchPass;
    LogIn(){
        userName = "";
        password = "";
        matchUser = "WAH";
        matchPass = "WAH";
    }
public:
    static LogIn* getInstance();

    static void releaseInstance();

    string getUserName();
    void setUserName();

    string getPassword();
    void setPassword();

    string getMatchUser();
    string getMatchPass();


};

int userLogIn();
```

LogIn.cpp:

```cpp
#include "LogIn.h"

string LogIn::getUserName() {
    return userName;
}

void LogIn::setUserName() {
    cin>>userName;
}

string LogIn::getPassword() {
    return password;
}

void LogIn::setPassword() {
    cin>>password;
}

string LogIn::getMatchUser() {
    return matchUser;
}

string LogIn::getMatchPass() {
    return matchPass;
}

LogIn *LogIn::getInstance() {
    if (ptr == nullptr){
        ptr = new LogIn;
    }
    return ptr;
}

void LogIn::releaseInstance() {
    if (ptr != nullptr) {
        delete ptr;
        ptr = nullptr;
    }
}

int userLogIn() {
    LogIn* logIn = LogIn::getInstance();
    static int tryCounter = 3;
    reLog:
    cout<<"Enter User Name:\n";
    logIn->setUserName();
    cout<<"Enter Password:\n";
    logIn->setPassword();
    if (logIn->getUserName() != logIn->getMatchUser() && logIn->getPassword() !=
logIn->getMatchPass()){
        if (tryCounter != 0){
            --tryCounter;
            cout<<"Invalid User Name or Password!\n";
            cout<<tryCounter<<" more try left.\n";
            goto reLog;
        }
        else{
            char asking[] = "LOCKING SYSTEM........";
            for (int i = 0; asking[i] != '\0' ; i++)
            {
                cout << asking[i];
                cout.flush();
                if (asking[i] == '\n')
                    Sleep(500);
                else
                    Sleep(5);
            }
            LogIn::releaseInstance();
            return 0;
        }
    }
    else
        return 1;
}

LogIn* LogIn::ptr = nullptr;
```

Address.h:

```
#include <iostream>
using namespace std;

class Address {
private:
    string city_name;
    string area;
    int street_number;
    int house_number;

public:
    Address(){
        city_name="";
        area="";
        street_number=0;
        house_number=0;
    }
    void setCityName(string x);
    void setArea(string a);
    void setStreetNumber(int c);
    void setHouseNumber(int y);
    string getCityName() const;
    string getArea() const;
    int getStreetNumber() const;
    int getHouseNumber() const;

};


#endif //GARAGE_ADDRESS_H
```

Address.cpp:

```
#include "Address.h"
void Address:: setCityName(string x){
    city_name=x;
}
void Address:: setArea(string a){
    area=a;
}
void  Address::setStreetNumber(int c){
    street_number=c;
}
void Address::setHouseNumber(int y){
    house_number=y;
}
string Address:: getCityName() const {
    return city_name;
}
string Address::getArea() const{
    return area;
}
int Address::getStreetNumber() const{
    return street_number;
}
int Address::getHouseNumber() const{
    return house_number;
}
```

Person.h:

```
#include "Address.h"
#include "fstream"

class Person {
protected:
    string first_name;
    string second_name;
    string  contact_no;
    Address address;
public:
    Person(){
        first_name= "";
        second_name="";
        contact_no="";
    }

    string setFirstName(string n);
    string getFirstName();

    string setSecondName(string m);
    string getSecondName();

    string setContactNo(string y);
    string getContactNo();

};


#endif //GARAGE_PERSON_H
```

Person.cpp:

```
#include "Person.h"

string Person::setFirstName( string n) {
    first_name =n;
    return first_name;
}

string Person::setSecondName(string m) {
    second_name=m;
    return second_name;
}

string Person::setContactNo(string y){
    contact_no=y;
    return contact_no;
}

string Person::getFirstName(){
    return first_name;
}

string Person::getSecondName(){
    return second_name;
}

string Person::getContactNo(){
    return contact_no;
}
```

Customer.h:

```
class Customer:public Person{
    string number_plate;
public:
    Customer(){
        number_plate = "";
    }

    string setNumberPlate(string x);
    string getNumberPlate();

    void writeToFile();
    friend void readFromFileCustomer();

    friend istream &operator >> (istream& input,Customer& person);
    friend ostream &operator << (ostream& output,const Customer& person);

};

void readFromFileCustomer();
```

Customer.cpp:

```
#include "Customer.h"

string Customer::getNumberPlate(){
    return number_plate;
}

string Customer::setNumberPlate(string x){
    number_plate=x;
    return number_plate;
}

istream &operator>>(istream &input, Customer& person) {
    int i;string s;
    cout<<"Enter Customer First Name:"<<endl;
    input>>person.first_name;
    cout<<"Enter Customer Second Name:"<<endl;
```

```cpp
    input>>person.second_name;
    cout<<"Enter Customer Contact Number:"<<endl;
    input>>person.contact_no;
    cout<<"Enter Customer Vehicle Registration Number:"<<endl;
    input>>person.number_plate;
    cout<<"Enter Customer City Name:"<<endl;
    input>>s; person.address.setCityName(s);
    cout<<"Enter Customer Area Name:"<<endl;
    cin.ignore();
    getline(input,s); person.address.setArea(s);
    cout<<"Enter Customer Street Number:"<<endl;
    input>>i; person.address.setStreetNumber(i);
    cout<<"Enter Customer House Number:"<<endl;
    input>>i; person.address.setHouseNumber(i);
    return input;
}

ostream &operator<<(ostream &output, const Customer &person) {
    output<<"First Name: "<<person.first_name<<endl;
    output<<"Second Name: "<<person.second_name<<endl;
    output<<"Contact Number: "<<person.contact_no<<endl;
    output<<"Vehicle Registration Number: "<<person.number_plate<<endl;
    output<<"City Name: "<<person.address.getCityName()<<endl;
    output<<"Area Name: "<<person.address.getArea()<<endl;
    output<<"Street Number: "<<person.address.getStreetNumber()<<endl;
    output<<"House Number: "<<person.address.getHouseNumber()<<endl;
    return output;
}

void Customer::writeToFile() {
    fstream write("Customer Record.txt",ios::out | ios::app);
    write<<first_name<<" "<<second_name<<" "<<contact_no<<" "<<number_plate<<"
"<<address.getCityName()
    <<" "<<address.getStreetNumber()<<" "<<address.getHouseNumber()<<"
"<<address.getArea()<<endl;
}

void readFromFileCustomer() {
    Customer temp;
    string counter;
    fstream read("Customer Record.txt",ios::in);
    int noOfEntries = 0;while (!read.eof()) {
        getline(read, counter);
        noOfEntries++;
    }
    read.close();
    string nameOne,nameTwo;
    cout<<"Enter First Name:\n";
    cin>>nameOne;
    cout<<"Enter Second Name:\n";
    cin>>nameTwo;
    read.open("Customer Record.txt",ios::in);
    string s;int j;
    for (int i = 0; i < noOfEntries; ++i) {
        read >> temp.first_name;
        read >> temp.second_name;
        read >> temp.contact_no;
        read >> temp.number_plate;
        read >> s; temp.address.setCityName(s);
        read >> j; temp.address.setStreetNumber(j);
        read >> j; temp.address.setHouseNumber(j);
        getline(read,s);
        temp.address.setArea(s);
        if (temp.first_name == nameOne){
            if (temp.second_name == nameTwo){
                cout<<temp;
                return;
            }
        }
    }
    cout<<"Record Not Found!\n";
}
```

Employee.h:

```
#include "Customer.h"

class Employee:public Person{
    double salary;
public:
    Employee(){
        salary = 0;
    }

    void writeToFile();
    friend void readFromFileEmployee();

    friend istream &operator >> (istream& input,Employee& person);
    friend ostream &operator << (ostream& output,const Employee& person);
};

void readFromFileEmployee();
```

Employee.cpp:

```
#include "Employee.h"

istream &operator>>(istream &input, Employee &person) {
    int i;string s;
    cout<<"Enter Employee First Name:"<<endl;
    input>>person.first_name;
    cout<<"Enter Employee Second Name:"<<endl;
    input>>person.second_name;
    cout<<"Enter Employee Contact Number:"<<endl;
    input>>person.contact_no;
    cout<<"Enter Employee Monthly Salary:"<<endl;
    input>>person.salary;
    cout<<"Enter Employee City Name:"<<endl;
    input>>s; person.address.setCityName(s);
    cout<<"Enter Employee Area Name:"<<endl;
    cin.ignore();
    getline(input,s); person.address.setArea(s);
    cout<<"Enter Employee Street Number:"<<endl;
    input>>i; person.address.setStreetNumber(i);
    cout<<"Enter Customer House Number:"<<endl;
    input>>i; person.address.setHouseNumber(i);
    return input;
}

ostream &operator<<(ostream &output, const Employee &person) {
    output<<"First Name: "<<person.first_name<<endl;
    output<<"Second Name: "<<person.second_name<<endl;
    output<<"Contact Number: "<<person.contact_no<<endl;
    output<<"Salary: "<<person.salary<<" $"<<endl;
    output<<"City Name: "<<person.address.getCityName()<<endl;
    output<<"Area Name: "<<person.address.getArea()<<endl;
    output<<"Street Number: "<<person.address.getStreetNumber()<<endl;
    output<<"House Number: "<<person.address.getHouseNumber()<<endl;
    return output;
}

void Employee::writeToFile() {
    fstream write("Employee Record.txt",ios::out | ios::app);
    write<<first_name<<" "<<second_name<<" "<<contact_no<<" "<<salary<<"
"<<address.getCityName()
         <<" "<<address.getStreetNumber()<<" "<<address.getHouseNumber()<<"
"<<address.getArea()<<endl;
}

void readFromFileEmployee() {
    Employee temp;
    string counter;
    fstream read("Employee Record.txt",ios::in);
    int noOfEntries = 0;while (!read.eof()) {
        getline(read, counter);
        noOfEntries++;
    }
    read.close();
    string nameOne,nameTwo;
    cout<<"Enter First Name:\n";
    cin>>nameOne;
    cout<<"Enter Second Name:\n";
    cin>>nameTwo;
    read.open("Employee Record.txt",ios::in);
    string s;int j;
    for (int i = 0; i < noOfEntries; ++i) {
        read >> temp.first_name;
        read >> temp.second_name;
        read >> temp.contact_no;
        read >> temp.salary;
        read >> s; temp.address.setCityName(s);
        read >> j; temp.address.setStreetNumber(j);
        read >> j; temp.address.setHouseNumber(j);
        getline(read,s);
        temp.address.setArea(s);
        if (temp.first_name == nameOne){
            if (temp.second_name == nameTwo){
                cout<<temp;
                return;
            }
        }
    }
    cout<<"Record Not Found!\n";
}
```

Vehicle.h:

```
#include<iostream>
#include <fstream>
using namespace std;

class Vehicle{
protected:
    int noOfDoors;
    int noOfTyres;
    int noOfSeats;
    int engineCC;

    string numPlate;
    string color;
    string transmissionType;
    string fault;
    string vehicleType;
public:
    Vehicle(){
        noOfDoors = 0;
        noOfSeats = 0;
        noOfTyres = 0;
        engineCC = 0;
        numPlate = "";
        color = "";
        transmissionType = "";
        fault = "";
        vehicleType = "";
    }

    void generalInput();
    void generalOutput() const;

    virtual void vehicleReturn() = 0;
};
```

Vehicle.cpp:

```
#include "Vehicle.h"

void Vehicle::generalInput(){
    cout<<"\nEnter no of Doors:\n";//bat suno apni apni Car k operator overloading m
apni car k mutabik
    cin>>noOfDoors;                    // doors, seats, or tyres k check laga dena
refrence k liye SportsCar ki cpp
    cout<<"Enter no of Seats:\n";      // file m cin ki operator over loading check
karo....
    cin>>noOfSeats;                    // :D I believe in you ( ´••)╯(._.`)
    cout<<"Enter no of Tyres:\n";
    cin>>noOfTyres;
    cout<<"Enter Engine Capacity:\n";
    cin>>engineCC;
    gI:
    cout<<"Enter Transmission Type:\n";
    cin>>transmissionType;
    if (transmissionType != "auto" && transmissionType != "Auto" && transmissionType
!= "manual" && transmissionType != "Manual"){
        cout<<"Enter \"Auto\" or \"Manual\" only.\n";
        goto gI;
    }
    cout<<"Enter Registration Number:\n";
    cin>>numPlate;
    cout<<"Enter Color:\n";
    cin>>color;
    cin.ignore();
    cout<<"Enter briefly about Faults in "<<vehicleType<<":\n";
    getline(cin,fault);
}

void Vehicle::generalOutput() const {
    cout<<"\nVehicle Type: "<<vehicleType<<endl;
    cout<<"Engine Capacity: "<<engineCC<<" CC"<<endl;
    cout<<"Transmission Type: "<<transmissionType<<endl;
    cout<<"Number of Doors: "<<noOfDoors<<endl;
    cout<<"Number of Seats: "<<noOfSeats<<endl;
    cout<<"Number of Tyres: "<<noOfTyres<<endl;
    cout<<"Registration Number: "<<numPlate<<endl;
    cout<<"Faults: "<<fault<<endl;
}
```

SportsCar.h:

```
#include "Vehicle.h"
//#pragma once

class SportsCar:public Vehicle{
    string turboType;
    string spoilerType;
public:
    SportsCar(){
        turboType = "";
        spoilerType = "";
        vehicleType = "Sports Car";
    }

    void vehicleReturn();

    void saveData();

    friend void readDataSpecific();

    friend ostream &operator << (ostream&,const SportsCar&);
    friend istream &operator >> (istream&,SportsCar&);

};

void readDataSpecific();
```

SportsCar.cpp:

```
#include "SportsCar.h"

ostream &operator << (ostream& o,const SportsCar& temp) {
    temp.generalOutput();
    o<<"Turbo Type: "<<temp.turboType<<endl;
    o<<"Spoiler Type: "<<temp.spoilerType<<endl;
    return o;
}

istream &operator >> (istream& i,SportsCar& temp) {
    temp.generalInput();
    cout<<"Enter Turbo Type:\n";
    i>>temp.turboType;
    cout<<"Enter Spoiler Type:\n";
    i>>temp.spoilerType;

    sc0:
    if (temp.noOfDoors <= 0 || temp.noOfDoors == 3 || temp.noOfDoors > 4){
        cout<<"Invalid Number of Doors, Enter again.\n";
        cout<<"Enter no of Doors:\n";
        i>>temp.noOfDoors;
        goto sc0;
    }

    sc1:
    if (temp.noOfSeats <= 0 || temp.noOfSeats == 3 || temp.noOfSeats > 4){
        cout<<"Invalid Number of Seats, Enter again.\n";
        cout<<"Enter no of Seats:\n";
        i>>temp.noOfSeats;
        goto sc1;
    }

    sc2:
    if (temp.noOfTyres != 4){
        cout<<"Invalid Number of Tyres, Enter again.\n";
        cout<<"Enter no of Tyres:\n";
        i>>temp.noOfTyres;
        goto sc2;
    }
    return i;
}

void SportsCar::vehicleReturn() {
    cout<<"Classification of Car: "<<vehicleType<<endl;
    cout<<"Registration Number: "<<numPlate<<endl;
    cout<<"Color: "<<color<<endl;
}


void SportsCar::saveData() {
    fstream in("SportsCar Record.txt",ios::out | ios::app);
    in<<numPlate<<" "<<noOfDoors<<" "<<noOfTyres<<" "<<noOfSeats<<" "<<engineCC<<"
"<<color<<" "
    <<transmissionType<<" "<<vehicleType<<" "<<turboType<<" "<<spoilerType<<"
"<<fault;
    in.close();
}

void readDataSpecific() {
    SportsCar sc;
    string counter;
    fstream out("SportsCar Record.txt", ios::in);
    int noOfEntries = 0;
    while (!out.eof()) {
        getline(out, counter);
        noOfEntries++;
    }
    out.close();
    string temp, type;
    cout << "Enter registration number:\n";
    cin >> temp;
    out.open("SportsCar Record.txt", ios::in);
    for (int i = 0; i < noOfEntries; ++i) {
        out >> sc.numPlate;
        out >> sc.noOfDoors;
        out >> sc.noOfTyres;
        out >> sc.noOfSeats;
        out >> sc.engineCC;
        out >> sc.color;
        out >> sc.transmissionType;
        out >> sc.vehicleType;
        out >> type;
        out >> sc.turboType;
        out >> sc.spoilerType;
        sc.vehicleType = sc.vehicleType + " " + type;
        getline(out, sc.fault);
        if (sc.numPlate == temp) {
            cout << sc;
            return;
        }
    }
    cout << "Record Not Found.\n";
}
```

HatchBack.h:

```
#include "Vehicle.h"
#include "fstream"
class Hatchback: public Vehicle {
protected:
    int airBags;
    string powerLocks;

public:
    Hatchback()
    {
        airBags=0;
        powerLocks=" ";
        vehicleType="HatchBack";

    }


    void vehicleReturn();  // virtual function by polymerization
    void dataRecord(); // data record
    void dataReading(); // data output

    friend ostream &operator << (ostream& ,const Hatchback&); // operator overloading
of hatchback
    friend istream &operator >> (istream& ,Hatchback&);

};
```

HatchBack.cpp:

```
#include "Hatchback.h"
void Hatchback:: vehicleReturn() {   //virtual function call as in other car classes
    cout<<"Classification of Car: "<<vehicleType<<endl;
    cout<<"Registration Number: "<<numPlate<<endl;
    cout<<"Color: "<<color<<endl;
}

istream & operator >>(istream& in ,Hatchback& obj){
    obj.generalInput();
    cout<<" enter number of airbags"<<endl;
    in>>obj.airBags;

    cout<<" Enter type of Power locks"<<endl;  // door locking ability of car(single
basically automatic system k bahir se lock hojaty hain or dual ka either way )

    in>>obj.powerLocks;
    if(obj.powerLocks!=" single"&& obj.powerLocks!="Single"&&obj.powerLocks!=" dual"&&
obj.powerLocks!="Dual")
    {
        cout<<"Enter \"single \" or \"dual\" only.\n";
        in>>obj.powerLocks;
    }

    while(true) {
        if (obj.noOfDoors != 4) {
            cout << "Invalid Number of Doors, Enter again.\n";
            cout << "Enter no of Doors:\n";
            in >> obj.noOfDoors;

        }
        else
            break;
    }
    while(true) {
        if (obj.noOfSeats != 5) {
            cout << "Invalid Number of Seats, Enter again.\n";
            cout << "Enter no of Seats:\n";
            in >> obj.noOfSeats;

        }
        else
            break;
    }
    while(true) {
        if (obj.noOfTyres != 4) {
            cout << "Invalid Number of Tyres, Enter again.\n";
            cout << "Enter no of Tyres:\n";
            in >> obj.noOfTyres;

        }
        else
            break;
    }
    return in;
}
ostream &operator << (ostream& out,const Hatchback& obj) {
    obj.generalOutput();
    out<<"PowerLocks Type: "<<obj.powerLocks<<endl;
    out<<"AirBags Numbers: "<<obj.airBags<<endl;
    return out;
}


void Hatchback::dataRecord() {
    ofstream input("Hatchback Record.txt",ios::out | ios::app);
    input<<numPlate<<" "<<noOfDoors<<" "<<noOfTyres<<" "<<noOfSeats<<" "<<engineCC<<"
"
          <<color<<" "<<transmissionType<<" "<<vehicleType<<" "<<airBags<<"
"<<powerLocks<<" "<<fault<<endl;

    input.close();
}

void Hatchback:: dataReading() {
    Hatchback car;
    string input;
    ifstream output("Hatchback Record.txt");
    int noOfEntries = 0;
    while (!output.eof()) {
        getline(output, input);
        noOfEntries++;
    }
    output.close();
    string temp, type;
    cout << "Enter registration number:\n";
    cin >> temp;
    output.open("Hatchback Record.txt", ios::in);
    for (int i = 0; i < noOfEntries; ++i) {
        output >> car.numPlate;
        output >> car.noOfDoors;
        output >> car.noOfTyres;
        output >> car.noOfSeats;
        output >> car.engineCC;
        output >> car.color;
        output >> car.transmissionType;
        output >> car.vehicleType;
        output >> car.airBags;
        output >> car.powerLocks;
        getline(output, car.fault);
        if (car.numPlate == temp) {
            cout << car;
            return;
        }
    }
    cout << "Record Not Found.\n";
}
```

CustomerLinkVehicle.h:

```
#include "Customer.h"
#include "SportsCar.h"

class CusVeh{ //This class will be used for association and aggregation
    Customer customer;
    SportsCar sc;
public:
    void input();
};
```