```
/*Aufgabe Nr./Task No.: H5
Nachname/Last,Family Name: Dewan
Vorname/First,Given Name: Sadek
Matr.-nr./Matr.-no.: 3056001
Uni-Email: sadek.dewan@stud.uni-due.de
Studiengang/Course of Studies: ISE CE*/
#define _GLIBCXX_USE_CXX11_ABI 0
#include <iostream>
#include<string>
#include <sstream>
#include <cstdlib>
#include <iomanip>
#include <regex>
using namespace std;

const int daysInMonth[12] = { 31,28,31,30,31,30,31,31, 30, 31, 30, 31 };
class Date {
private:
        unsigned int day, month, year;
public:
        Date() {};
        Date(int a_day, int a_month, int a_year)
        {
                day = a_day;
                month = a_month;
                year = a_year;
        };
        string toString()
        {
                std::ostringstream ss;
                ss << day << '.' << month << '.' << year;
                return ss.str();
        };
        friend Date operator+(Date date, int n);

};

Date operator+(Date date, int n)
{
        date.day += n;
        if (date.day > daysInMonth[date.month-1])
        {
                date.day -= daysInMonth[date.month-1];
                if (date.month == 12)
                {
                        date.year++;
                        date.month = 1;
                }
                else
                        date.month++;
        }
        return date;
};

enum Board { NoMeal, Breakfast, HalfPension, AllInclusive };

class Hotel {
private:
        string name;
        int nights, singles, doubles;
        Board board;
        float priceNightSingle, priceNightDouble;
        bool parking;
        Date arrivalDate;
public:
        ~Hotel()
        {
                std::cout << "destructor Hotel" + name + " at " <<
arrivalDate.toString() <<  " for " << singles+2*doubles << " guests done" << endl;
        };
        Hotel(string a_name, int a_nights, int a_singles, int a_doubles, Board
a_board, float a_priceNightSingle, float a_priceNightDouble, bool a_parking, Date
a_arrivalDate)
        {
                name = a_name;
                nights = a_nights;
                singles = a_singles;
                doubles = a_doubles;
                board = a_board;
                priceNightSingle = a_priceNightSingle;
                priceNightDouble = a_priceNightDouble;
                parking = a_parking;
                arrivalDate = a_arrivalDate;
        };
        float get_price()
        {
                float price = ((priceNightSingle * singles) + (priceNightDouble *
doubles)) * nights;
                if (parking)
                        price += 10 * nights;
                return price;
        }
        Date get_arrival()
        {
                return arrivalDate;
        }
        Date get_checkout()
        {
                Date d = arrivalDate + nights;
                return d;
        }
        int get_guests()
        {
                return singles + 2 * doubles;
        }
        void print()
        {
                string s;
                switch (board)
                {
                case AllInclusive:
                                s = "all inclusive";
                                break;
                case Breakfast:
                        s = "breakfast";
                        break;
                case NoMeal:
                        s = "no meal";
                        break;
                case HalfPension:
                        s = "half pension";
                        break;
                }
                string sparking = parking ? ", parking included" : "";
                cout << arrivalDate.toString() << " " << name << " for " << nights
<< " night(s) "
                << singles << " single bed room(s) " << doubles << " double bed
room(s) " << endl;
                cout << "               " << s << sparking << endl;
        }
};

class Transport
{
public:
        virtual float get_price() = 0;
        virtual bool withTransfer() = 0;
        virtual void print() = 0;
        virtual ~Transport()
        {
                cout << "destructor Transport done" << endl;
        };
};

class Selforganised : public Transport
{
public:
        Selforganised() {};
        virtual ~Selforganised()
        {
                cout << "destructor SelfOrganized done" << endl;
        }
        virtual bool withTransfer()
        {
                return false;
        }
        virtual float get_price()
        {
                return 0.0;
        }
        virtual void print()
        {
                cout << "self organized transport" << endl;
        }
};

class PublicTransport : public Transport
{
private:
        Date departure;
        string code, from, to;
protected:
        float priceOneSeat;
        bool firstClass;
public:
        PublicTransport(Date a_departure, string a_code, string a_from, string a_to,
float a_priceOneSeat, bool a_firstClass = false)
        {
                departure = a_departure;
                code = a_code;
                from = a_from;
                to = a_to;
                priceOneSeat = a_priceOneSeat;
                firstClass = a_firstClass;
        };
        ~PublicTransport()
        {
                cout << "destructor PublicTransport " << code << " at " <<
departure.toString() << endl;
        };
        bool get_firstclass() { return firstClass; }
        virtual void print()
        {
                cout <<  departure.toString() << " " << code << " from: " << from <<
" to: " << to << endl;
        }
};

class Flight : public PublicTransport
```

```
{
private:
        bool transfer;
public:
        Flight(Date a_departure, string a_code, string a_from, string a_to, float
a_priceOneSeat, bool a_transfer, bool a_firstClass = false) :
PublicTransport(a_departure, a_code, a_from, a_to, a_priceOneSeat, a_firstClass =
false)
        {
                transfer = a_transfer;
        }
        ~Flight()
        {
                cout << "destructor Flight done" << endl;
        };
        virtual bool withTransfer() { return transfer; }
        virtual float get_price()
        {
                int i = firstClass ? 2 : 1;
                float f = i * priceOneSeat;
                return f;
        }
        virtual void print()
        {
                cout << "flight ";
                PublicTransport::print();
        }
};

class Train : public PublicTransport
{
public:
        Train(Date a_departure, string a_code, string a_from, string a_to, float
a_priceOneSeat, bool a_firstClass = false) : PublicTransport(a_departure, a_code,
a_from, a_to, a_priceOneSeat, a_firstClass = false)
        {

        };
        virtual ~Train()
        {
                cout << "distructor train done" << endl;
        }
        virtual float get_price()
        {
                float mult = firstClass ? 1.5f : 1;
                return priceOneSeat * mult;
        }
        virtual void print()
        {
                cout << "train ";
                PublicTransport::print();
        }
        virtual bool withTransfer() { return false; }

};

class Trip
{
private:
        const unsigned int no;
        static unsigned int lastNo;
        unsigned int travelers;
        Hotel* hotel;
        Transport* transportOutward;
        Transport* transportBack;
        Trip* next;
public:
        Trip(unsigned int a_travalers, Hotel* a_hotel = NULL, Transport* a_out =
NULL, Transport* a_back = NULL, Trip* a_next = NULL):no(lastNo)
        {
                lastNo++;
                travelers = a_travalers;
                hotel = a_hotel;
                transportBack = a_back;
                transportOutward = a_out;
                next = a_next;
        };
        ~Trip()
        {
                delete hotel;
                delete transportOutward;
                delete transportBack;
                cout << "distructor trip done" << endl;
        }
        unsigned int get_no()
        {
                return no;
        }
        Trip* get_next()
        {
                if (next != NULL)
                        return next;
                else return NULL;
        }
        void set_next(Trip* t)
        {
                next = t;
        }
        float get_price()
        {
                float sum = 0;
                sum += hotel->get_price();
                sum += transportOutward->get_price() * travelers;
                sum += transportBack->get_price() * travelers;
                return sum;
        }
        void print()
        {
                cout << "trip inquiry " << no << " for " << travelers << "
person(s)" << endl;
                cout << "check-in: ";
                hotel->print();
                cout << "outward journey: ";
                transportOutward->print();
                cout << "journey back: ";
                transportBack->print();
                if ((transportOutward->withTransfer()) ||
(transportBack->withTransfer()))
                        cout << "transfer required" << endl;
                else
                        cout << "no transfer" << endl;
                cout << "       price: " << fixed << setprecision(2) << get_price()
<< " EUR" << endl;
        }
};

class TravelAgency
{
private:
        Trip* trips;
public:
        TravelAgency()
        {
                trips = NULL;
        };
        void add(Trip* newtrip)
        {
                newtrip->set_next(trips);
                trips = newtrip;
        }
        void remove()
        {
                Trip* trip_to_delete = trips;
                trips = trips->get_next();
                delete trip_to_delete;
        }
        void remove(Trip* t)
        {
                if (t == trips)
                {
                        remove();
                        return;
                }
                Trip* current = trips;
                if (t != trips)
                {
                        while (current->get_next() != t)
                        {
                                current = current->get_next();
                        }
                }
                current->set_next(t->get_next());
                delete t;
        }
        Trip* search(unsigned int number)
        {
                if (trips == NULL)
                        return NULL;
                if (trips->get_no() == number)
                        return trips;
                Trip* current = trips;
                do
                {
                        current = current->get_next();
                        if (current->get_no() == number)
                                return current;
                } while (current->get_next());
                return NULL;
        }
        void printAllTrips()
        {
                if (trips == NULL)
                        return;
                Trip* current = trips;
                do
                {
                        current->print();
                        current = current->get_next();
                } while (current != NULL);
        }
};

Flight* add_flight(Date departure)
{

        string code, from, to;
        float priceOneSeat;
        bool firstClass, transfer;

        std::string sprice, sdate;
        cout << "Please enter flight code: ";
        getline(cin, code);
        cout << "Please enter departure airport: ";
        getline(cin,from);
        cout << "Please enter arrival airport: ";
        getline(cin,to);
        cout << "Please enter price for single passanger: ";
        getline(cin, sprice);
        priceOneSeat = atof(sprice.c_str());
        string c;
        cout << "first class required (y(es) or n(o)): ";
        getline(cin, c);
        if (c[0] == 'y') firstClass = true;
        if (c[0] == 'n') firstClass = false;
        c = "";
        cout << "Transfer to or from airport required (y(es) or n(o)): ";
        getline(cin,c);
        if (c[0] == 'y') transfer = true;
        if (c[0] == 'n') transfer = false;
        Flight* f = new Flight(departure, code, from, to, priceOneSeat, transfer, 
firstClass);
        return f;
}

Train* add_train(Date departure)
{

        string code, from, to;
        float priceOneSeat;
        bool firstClass;

        string sprice;

        cout << "code of train: ";
        getline(cin, code);
        cout << "main train station of departure: ";
        getline(cin, from);
        cout << "main train station of arrival: ";
        getline(cin, to);
        cout << "price for one passenger: ";
        getline(cin, sprice);
        priceOneSeat = atof(sprice.c_str());
        string c;
//      cout << "First class required (y(es) or n(o)): ";
//      getline(cin, c);
//      if (c[0] == 'y') firstClass = true;
//      if (c[0] == 'n') firstClass = false;
        firstClass = false;
        Train* t = new Train(departure, code, from, to, priceOneSeat, firstClass);
        return t;
}

Transport* add_transport(Date date)
{
        int choise;
        do
        {
                cout << "0  self organised" << endl;
                cout << "1  by flight" << endl;
                cout << "2  by train" << endl;
                string schoise;
                getline(cin, schoise);
                choise = atoi(schoise.c_str());
                cout << "your choise: " << schoise << endl;

                switch (choise)
                {
                case 0:
                        return new Selforganised();
                        break;
                case 2:
                        return add_train(date);
                        break;
                case 1:
                        return add_flight(date);
                        break;
                }
        } while (choise != 0);
}

Hotel* add_hotel()
{
        string name;
        int nights, singles, doubles;
        Board board;
        float priceNightSingle, priceNightDouble;
        bool parking;
        string snights, ssingles, sdoubles, ssingleprice, sdoubleprice, sdate;
        cout << "name of hotel: ";
        getline(cin, name);
        bool b;
        int day, month, year;
        do
        {
                b = false;
                cout << "arrival on (DD.MM.YYYY): ";
                int i = 0;
                        getline(cin, sdate);
                        if (isdigit(sdate[1]))
```

```
                                day = (sdate[0] - '0') * 10 + (sdate[1] - '0');
                        else
                        {
                                day = sdate[0] - '0';
                                i++;
                        }
                        if (isdigit(sdate[4-i]))
                                month = (sdate[3-i] - '0') * 10 + (sdate[4-i] -
'0');
                        else
                        {
                                month = sdate[3 - i] - '0';
                                i++;
                        }
                        year = (sdate[6-i] - '0') * 1000 + (sdate[7-i] - '0') * 100
+ (sdate[8-i] - '0') * 10 + (sdate[9-i] - '0');
                        if ((month < 1) || (month > 12))
                        {
                                if ((day < 1) || (day > daysInMonth[month-1]) ||
(year < 1978) || (year > 2100))
                                {
                                        b = true;
                                        cout << "Please enter correct date" << endl;
                                }
                                b = true;
                                cout << "Please enter correct date" << endl;
                        }
        } while (b == true);
        cout << "how many nights: ";
        getline(cin, snights);
        cout << "how many single bedrooms: ";
        getline(cin, ssingles);
        cout << "how many double bedrooms: ";
        getline(cin, sdoubles);
        string s = "";
        while ((s[0] != 'a') && (s[0] != 'b') && (s[0] != 'h') && (s[0] != 'w'))
        {
                cout << "a all inclusive" << endl;
                cout << "b breakfast" << endl;
                cout << "h half board" << endl;
                cout << "w without meals" << endl;
                getline(cin,s);
                switch (s[0])
                {
                case 'a':
                        board = AllInclusive;
                        break;
                case 'h':
                        board = HalfPension;
                        break;
```

```
		case 'b':
			board = Breakfast;
			break;
		case 'w':
			board = NoMeal;
			break;
		}
		cout << "price one night for single room: ";
		getline(cin,ssingleprice);
		cout << "price one night for double room: ";
		getline(cin, sdoubleprice);
		cout << "With parking (y(es) or n(o)):";
		s = "";
		getline(cin, s);
		if (s[0] == 'y') parking = true;
		if (s[0] == 'n') parking = false;
		nights = atoi(snights.c_str());
		singles = atoi(ssingles.c_str());
		doubles = atoi(sdoubles.c_str());
		priceNightSingle = atof(ssingleprice.c_str());
		priceNightDouble = atof(sdoubleprice.c_str());



		Date arrivalDate(day, month, year);
		Hotel* h = new Hotel(name, nights, singles, doubles, board,
priceNightSingle, priceNightDouble, parking, arrivalDate);
		return h;
	}
}


Trip* add_trip()
{
	Hotel *h = add_hotel();

	cout << "please choose transport for outward journey: " << endl;
	Transport* outward = add_transport(h->get_arrival());
	cout << "please choose transport for back journey: " << endl;
	Transport* back = add_transport(h->get_checkout());

	Trip* t = new Trip(h->get_guests(), h, outward, back, NULL);
	return t;
}


unsigned int Trip::lastNo = 1;
int main()
{
	TravelAgency* ta = new TravelAgency();
```

```
string schoise="";
Trip* tr;
do
{
        cout << "HOTLINE TRAVEL AGENCY" << endl;
        cout << "0 exit" << endl;
        cout << "1 add new trip" << endl;
        cout << "2 search trip" << endl;
        cout << "3 view all trip offers" << endl;

        getline(cin,schoise);
        cout << "Your choise: " << schoise << endl;
        switch (schoise[0])
        {
        case '1':
                ta->add(add_trip());
                break;

        case '2':
        {
                cout << "Please enter trip number: " << endl;
                string sno;
                unsigned int no;
                getline(cin, sno);
                no = atoi(sno.c_str());
                tr = ta->search(no);
                if (tr == NULL)
                        cout << "Not found" << endl;
                else
                {
                        tr->print();
                        string c;
                        cout << "(d)elete or (n)o: ";
                        getline(cin, c);
                        switch (c[0])
                        {
                        case 'd':
                                ta->remove(tr);
                        }
                }
                break;
        }
        case '3':
        {
                ta->printAllTrips();
                break;
        }
        }
} while (schoise[0] != '0');
```

```
}
```