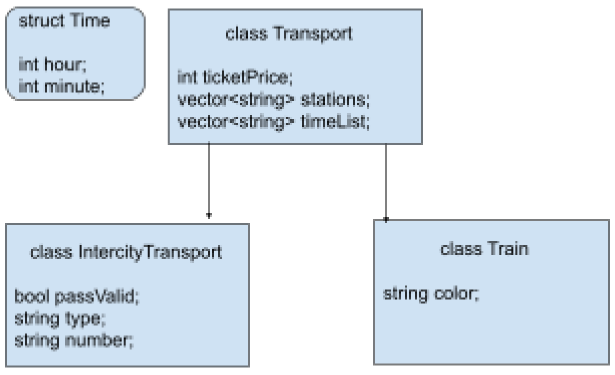
Project: “BudapestGo”
Themes: “File Handling” , “Object Oriented Programming”

This program for recording transport in the city of Budapest, an analogue of Budapest GO. As you know, there are several types of transport in Budapest, for example, trams, metro, buses, and so on. And all of them have specific stops, arrival times, and so on. So, the aim of this program is to store information about transport in Budapest, and also represent this information.

There are one abstract base class “Transport” and two derived classes “Train”, “IntercityTransport”



Files:
“Transport.h”, “IntercityTransport.h”, “Train.h” - contain declaration of attributes and functions of Transport, IntercityTransport, Train objects respectively.
“Transport.cpp”, “IntercityTransport.cpp”, “Train.cpp” - contain implementation of foregoing functions.
“main.cpp” - running program.
“transport.txt” - file created during execution of the program.

“Transport.h” / “Transport.cpp”

void printMenu() - function which prints the menu for the user
The user should have the following menu on the screen:

1 - Show all transport information that are in the file on the screen
2 - Add a transport
3 - Show the travel time of a particular transport line
4 - Show all transports that stop at a given station
5 - Stop the program

typedef struct Time{} - used for calculating the travel time of a particular transport line.

Time stringToTime(string _sTime) - converts string to struct Time. String as a parameter given in the following format “hh:mm”. The function returns Time type value.

void timeDifference(Time _t1, Time _t2) - calculate period of time between two times. Receives value of Time struct and prints the difference in format “hh:mm”. The source code was copied and modified. Reference: https://www.techcrashcourse.com/2017/01/cpp-program-to-find-difference-between-time-periods.html

Attributes:
int ticketPrice - ticket price for any transport.
vector<string> stations - list of stations (stops)
vector<string> timeList - list of arriving time for every station

int get_ticketPrice(); vector<string> get_stations(); vector<string> get_timeList() - getters to access from outside of the scope.

Transport() - default constructor of Transport class.
virtual ~Transport() - destructor. Virtual used to avoid memory leak.
Transport(const Transport& theOther) - copyConstructor.
Transport(int _ticketPrice, vector<string> _stations, vector<string> _timeList) parameterized constructor.

virtual string transportToString() = 0; - converts information about objects to a string. Since every derived class has this function, but implemented differently it is a virtual function. Later used to write objects’ information to a file.

void splitStringToVector(string s, vector<string>& v) - receives a string and splits it to the vector<string> type by space. Reference: https://slaystudy.com/c-split-string-by-space-into-vector/

“IntercityTransport.h” / “IntercityTransport.cpp”

IntercityTransport is derived publicly from the Transport class, therefore contains all the functions of the Transport class.

Attributes:
bool passValid - to check whether passengers need a pass or not. For example, valid for all trams, but not valid for “100E " bus.
string type - can be bus, metro, tram and sct.
string number - number (or ID) for the transport. For example, “2M” or “73” or “4/6”

bool get_passValid(); string get_type(); string get_number() - getters to access information out of the class;

IntercityTransport() - default constructor

IntercityTransport(const IntercityTransport& theOther) - copy constructor
virtual ~IntercityTransport() - destructor
IntercityTransport(int _ticketPrice, vector<string> _stations, vector<string> _timeList, bool _passValid, string _type, string _number) - parameterized constructor

string transportToString() - converts information about objects to a string.
void timeBetweenDestinationsIntercity(string _type, string _number, vector<IntercityTransport*>& _listOfIntercityTransport) - calculates and prints the time period between arriving time to the first station and arriving time to the second station.

void addInterCityTransport(fstream& _file, vector<IntercityTransport*>& _listOfIntercityTransport) - reads and writes the information about the object into the "transport.txt" file. PassValid should be "true" or "false" otherwise an exception will be thrown. Every hour should be less than 24 and every minute should be less than 60, otherwise an exception will be thrown.

void sameNameIntercityTransport(string _station, vector<IntercityTransport*>& _listOfIntercityTransport) - prints the information about the objects with the same station on the screen.

"Train.h" / "Train.cpp"

Train is derived publicly from the Transport class, therefore contains all the functions of the Transport class.

Attributes:
string color - color identifier of the train.

string get_color() - to access information out of the class;

Train() - default constructor
Train(const Train& theOther) - copy constructor
virtual ~Train() - destructor
Train(int _ticketPrice, vector<string> _stations, vector<string> _timeList, string _color) - parameterized constructor

string transportToString() - converts information about objects to a string.

void timeBetweenDestinationsTrain(string _color, vector<Train*>& _listOfTrains) - calculates and prints the time period between arriving time to the first station and arriving time to the second station.

void addTrain(fstream& _file, vector<Train*>& _listOfTrain) - reads and writes the information about the object into the "transport.txt" file. Every hour should be less than 24 and every minute should be less than 60, otherwise an exception will be thrown.

void sameNameTrains(string _station, vector<Train*>& _listOfTrains) - prints the information about the objects with the same station on the screen.

“main.cpp”
case 1: prints all information which “transport.txt” contains.
case 2: add transport to the file and memory database.
case 3: prints time period of the destination of the particular transport
case 4: prints transport with the same station.
case 5: deallocate memory and stop the program.

To test the program:
1) 2->1
To add IntercityTransport:
type: “Tram”
stations: “Oktogon Blaha Corvin”
number: “4/6”
ticketPrice: 340
passValid: “true”
timelist: “10:20 11:30 13:40”

2->2
To add Train:
ticketPrice: 1000
color: “yellow”
stations: “Budapest Debrecen”
timelist: “12:10 14:00”

2) 1
3) 3->1
type: “Tram”
number: “4/6”

4) 4
station: “Budapest”
5) 5