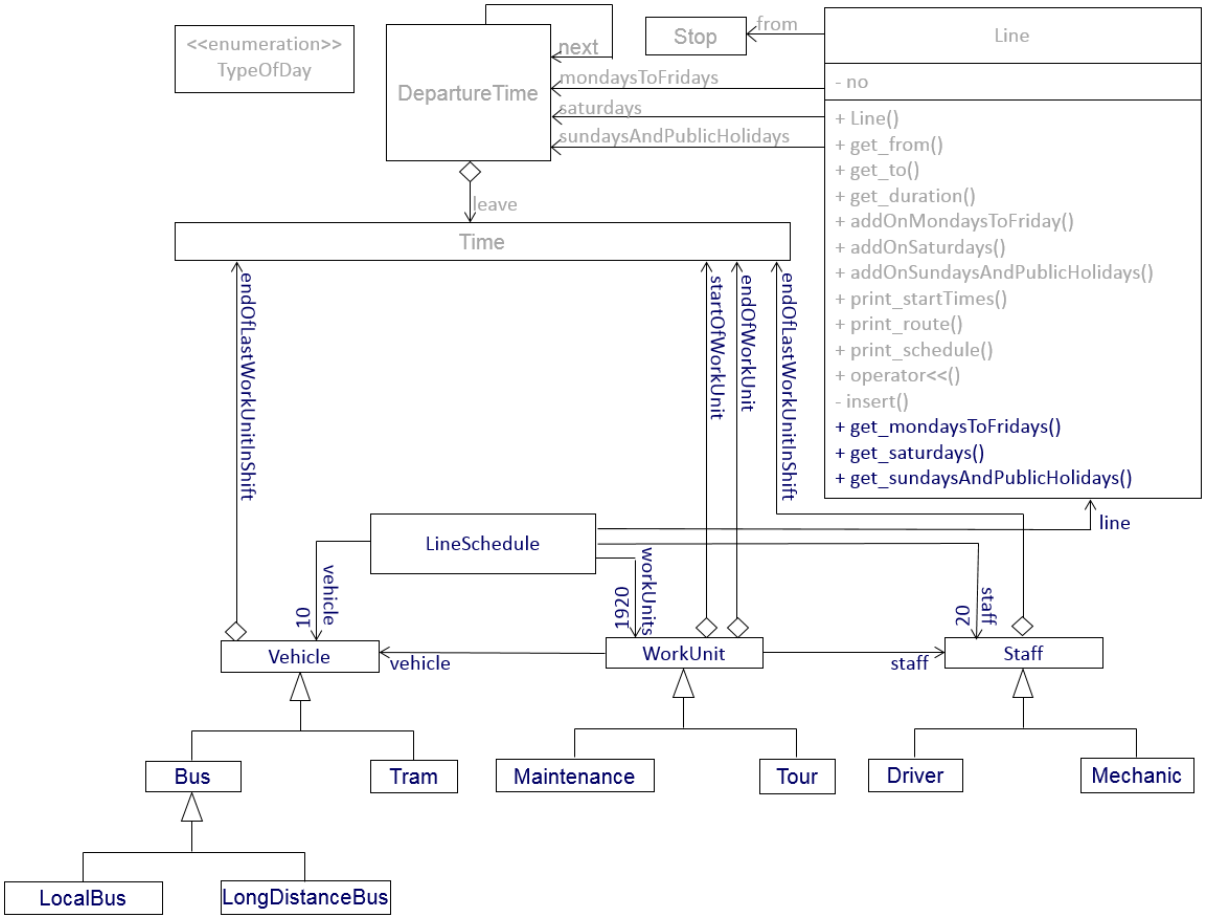
Schedule Vehicles and Staff for Public Transport Line

Your prototype of electronic (bus/tram/tube) line maps has found acceptance from your customer Duisburg Public Transport Company (DVG). Therefore as a follow-up contract also an electronic schedule for vehicles and staff shall be realised by the well known Duisburg software company "Software Nerds". Again the software architects have already extended the given UML class diagram of task H4 (due to easy readability data types, parameters and destructors are left out), which you should implement in C++ and test with example bus line 933.



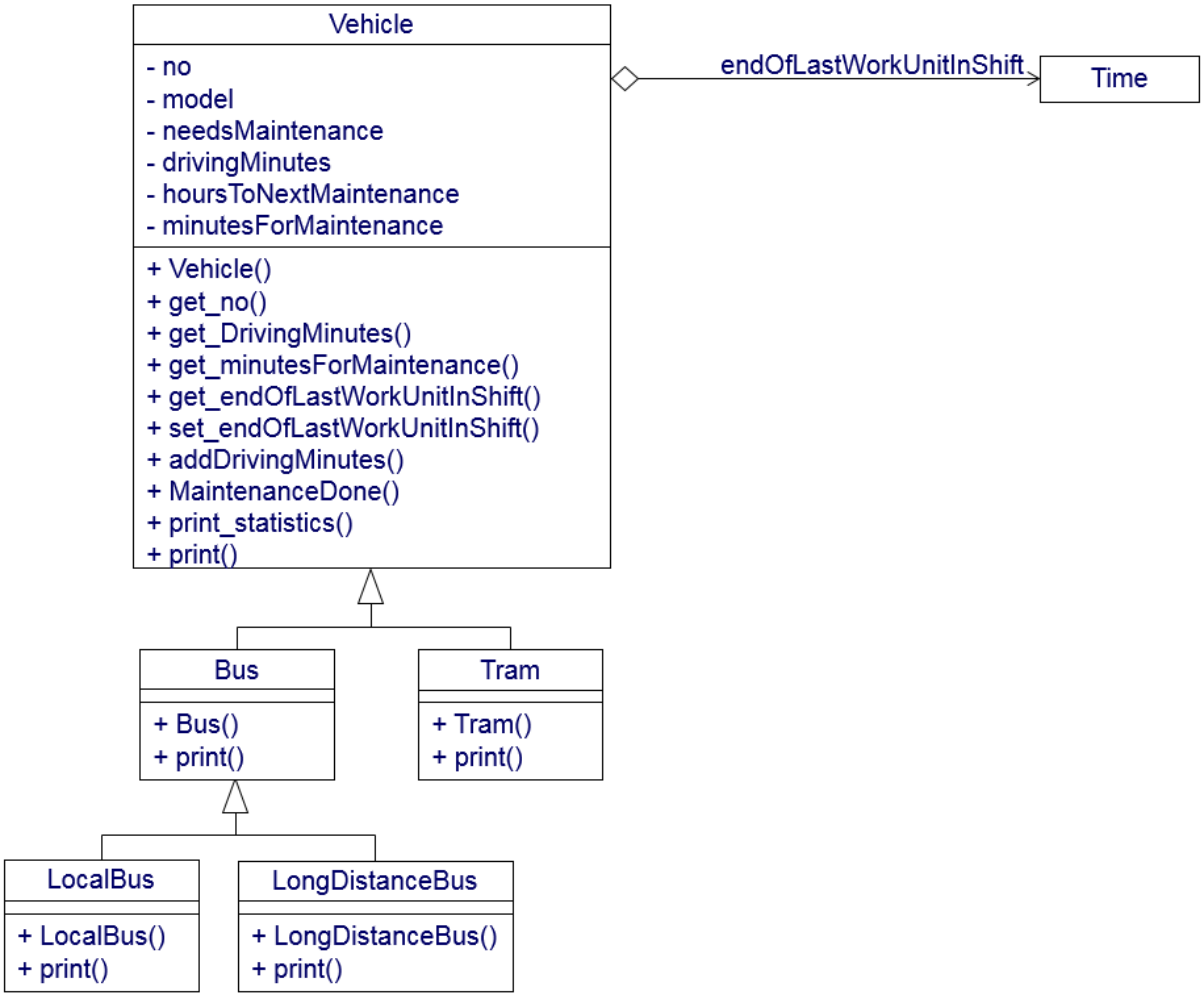
grey text: in task H4 already implemented.

Subtask1:

Extend the class with name **`Line`** for a (bus/tram/tube) line by,

three public methods with names **`get_mondaysToFridays`**, **`get_saturdays`** and **`get_sundaysAndPublicHolidays`** without parameters returning the respective pointers to the lists of departure times.

Subtask2:



Define a small class hierarchy for public transport vehicles.
First define a base class with name **`Vehicle`** with following members:
*a private unsigned integer attribute with name **`no`** for the identification number of a vehicle.
*a private C++ string attribute with name **`model`** storing the model description.
*a private Boolean attribute with name **`needsMaintenance`** whether the vehicle needs a maintenance.
*a private unsigned integer attribute with name **`drivingMinutes`** for the count of driving minutes after last maintenance.
*a private constant unsigned integer attribute with name **`hoursToNextMaintenance`** for the number of driving hours in-between two maintenances.
*a private constant unsigned integer attribute with name **`minutesForMaintenance`** for the minutes how long a maintenance will take.
*a private attribute with name **`endOfLastWorkUnitInShift`** of type **`Time`** to store the time until when the vehicle is busy/in service in a shift work.
*a public constructor with four parameters for the number of the vehicle, the model and the two constants for the number of hours in-between two maintenances and the duration of a maintenance. The respective four attributes shall be initialised by an initialisation list, additionally the driving minutes set to value 0, the Boolean variable to **`false`** and the clock time for the end of last tour of maintenance to 00:00 o'clock.

*four public methods with names **get_no**, **get_drivingMinutes**, **get_minutesForMaintenance** and **get_endOfLastWorkUnitInShift** without parameters returning the values of the respective attributes.
*a public method with name **set_endOfLastWorkUnitInShift** with a clock time of class **Time** as parameter and without return values assigning the clock time to the respective attribute.
*a public method with name **addDrivingMinutes** with a number of minutes as parameter and without return values adding the value to the respective attribute as well as checking whether the driving minutes have exceeded the number of hours to next maintenance - if yes the Boolean attribute shall get value **true**.
*a public method with name **maintenanceDone** without parameter and without return values being called after a maintenance and resetting the value of the Boolean attribute to **false** and the driving minutes to value 0.
*a public method with name **print_statistics** without parameter and without return values after calling member function **print** outputting in round brackets the model description, **"driving time: "** and the driving time since last maintenance in hours and minutes as well as a following **"(*)"** in case of a maintenance is needed (see examples below).
*a public virtual method with name **print** without parameter and without return value writing the number of the vehicle onto the standard character output stream.
*Define a class **Bus** inherited publicly from class **Vehicle**. This class shall have a public constructor with four parameters for the number of the vehicle, the model, the number of hours in-between two maintenances as well as the duration of a maintenance and only call the constructor of the upper class.
*a public method with name **print** without parameter and without return value writing **"bus "** onto the standard character output stream and calling same name method of its upper class.
*Define a class **Tram** inherited publicly from class **Vehicle**. This class shall have a public constructor with four parameters for the number of the vehicle, the model, the number of hours in-between two maintenances as well as the duration of a maintenance and only call the constructor of the upper class.
*a public method with name **print** without parameter and without return value writing **"tram "** onto the standard character output stream and calling same name method of its upper class.
*Define a class **LocalBus** inherited publicly from class **Bus**. This class shall have a public constructor with four parameters for the number of the vehicle, the model, the number of hours in-between two maintenances as well as the duration of a maintenance and only call the constructor of the upper class.
*a public method with name **print** without parameter and without return value writing **"local "** onto the standard character output stream and calling same name method of its upper class.
*Define a class **LongDistanceBus** inherited publicly from class **Bus**. This class shall have a public constructor with four parameters for the number of the vehicle, the model, the number of hours in-between two maintenances as well as the duration of a maintenance and only call the constructor of the upper class.

*a public method with name **print** without parameter and without return value writing **"local "** onto the standard character output stream and calling same name method of its upper class.

Subtask3:

*Define a small class hierarchy for the staff.
First define a base class with name **Staff** with following members:

*a private unsigned integer attribute with name **no** for the staff number.a private C++ string attribute with name **name** storing the name.
*two private Boolean attributes with names **busLicense** and **tramLicense** storing whether the person of staff has a respective driving license.
*a private unsigned integer attribute with name **minutesWorkingShift** for the count of already done working minutes within a shift work.
*a private unsigned integer attribute with name **minutesWorkingWeek** for the count of already done working minutes within a working week.
*a private unsigned integer attribute with name **maxMinutesWorkingWeek** for the count of the minutes to work within a whole working week (e.g. in case of a 40 hours week 40 * 60 = 2400 Minuten).
*a private attribute with name **endOfLastWorkUnitInShift** of type **Time** to store the time until when the last work unit given in a shift work will last.
*a public constructor with three parameters for the staff number, the name and the number of working hours in a week with default value 40. The respective three attributes shall be initialised appropriately by an initialisation list, additionally no driving licenses, 0 working minutes in the shift and the week and the clock time for the end of the last work unit to 00:00.
*four public methods with names **get_no**, **get_minutesInWorkingShift**, **get_minutesInWorkingWeek**, **get_maxMinutesInWorkingWeek** and **get_endOfLastWorkUnitInShift** without parameters returning the values of the respective attributes.
*a public method with name **set_endOfLastWorkUnitInShift** with a clock time of class **Time** as parameter and without return values assigning the clock time to the respective attribute.
*a public method with name **addWorkingMinutes** with a number of minutes as parameter and without return value adding the parameter value to the attributes **minutesWorkingShift** and **minutesWorkingWeek**.
*a public method with name **newWorkingShift** without parameter and without return values setting value 0 to **minutesWorkingShift** and clock time 00:00 to **endOfLastWorkUnitInShift**.
*two public methods with names **hasBusLicense** and **hasTramLicense** without parameters returning the values of the two respective attributes.
*two public methods with names **setBusLicense** and **setTramLicense** without parameters setting the values of the respective attributes to **true** in case of the respective driving license is passed.

*two public methods with names **unsetBusLicense** and **unsetTramLicense** without parameters setting the values of the respective attributes to **false** in case of the driving license expires or gets withdrawn.
*a public method with name **print_statistics** without parameter and without return value after calling member function **print** outputting string **" working time: shift "** and the already done work time in a work shift as well as **", week "** and the work time of the whole week (see examples below).
*a public virtual method with name **print** without parameter and without return value writing the staff number only onto the standard character output stream.
*Define a class **Driver** inherited publicly from class **Staff**. This class shall have a public constructor with two parameters for the staff number and the name and only call the constructor of the upper class.
*a public method with name **print** without parameter and without return value writing **"driver "** onto the standard character output stream and calling same name method of its upper class.
*Define a class **Mechanic** inherited publicly from class **Staff**. This class shall have a public constructor with two parameters for the staff number and the name and only call the constructor of the upper class with a 35 hour working week.
*a public method with name **print** without parameter and without return value writing **"mechanic "** onto the standard character output stream and calling same name method of its upper class.

Subtask 3:

Define a class hierarchy for the working units being typical for a public person transport company.
*First define a base class with name **WorkUnit** with following members:

*two protected attributes with names **startOfWorkUnit** and **endOfWorkUnit** of type **Time** to store the time a planned/done work unit lasts.
*a protected reference attribute with name **vehicle** of type **Vehicle** to a vehicle which will be driven or maintained during a work unit.
*a protected reference attribute with name **staff** of type **Staff** to a person of staff who will drive or maintain a vehicle in a work unit.
*a public constructor with four parameters for the start of the work unit of type **Time** and its duration in minutes, a reference to a vehicle of type **Vehicle** and a reference to a person of staff of type **Staff**. The two attributes of type Time shall be initialised with start time by the first parameter and the calculated end time by its duration, the two reference attributes by the respective reference parameters.
*a public pure virtual method with name **print** without parameter and without return value.
*Define a class **Tour** inherited publicly from class **WorkUnit** modeling a circulation/return drive with a vehicle and a driver. This class shall have a public constructor with four initialisation parameters for the start of the work unit of type **Time** and its duration in minutes, the reference to a vehicle of type **Vehicle** and the reference to a driver of type **Staff**. For the driver as well as the vehicle the duration shall be added as working or driving time and for both the end time for this work unit as last one in shift work.
*a public method with name **print** without parameter and without return value

writing **"tour at "** and the start time of the work unit, then **": "** and sending message **print** to the vehicle, following writing **"; "** and sending message **print** to the driver, and last in round brackets **"until "** and the end time of the work unit (see examples below).
*Define a class **Maintenace** inherited publicly from class **WorkUnit** modeling a service for a vehicle by a mechanic as work unit. This class shall have
a public constructor with four initialisation parameters for the start of the work unit of type **Time** and its duration in minutes, the reference to a vehicle of type **Vehicle** and the reference to a driver of type **Staff**. For the mechanic the duration shall be added as working time, for the mechanic as well as for the vehicle the end time for this work unit as last one in shift work, and the vehicle shall be send a message **maintenanceDone**.
*a public method with name **print** without parameter and without return value
writing **"maintenance for "**, sending message **print** to the vehicle, writing **" by "** and sending message **print** to the mechanic, following writing **" from "** and the start time of the work unit last in round brackets **"until "** and the end time of the work unit and last in round brackets **" until "** and the end time (see examples below).

Subtask 5:

Define a class with name **LineSchedule** for a time schedule of vehicles and staff for a line with following members:

*four constant private natural number attributes with names **maxVehicles** and value 10, **maxStaff** and value 20, **maxMinutesInWorkingShift** and value 8 * 60 = 480 as well as **maxWorkUnits** and value 24 * 4 * **maxStaff** = 1920 (a work unit will take at least 15 minutes).
*three private natural number attributes with names v**ehicles**, **persons** and **workUnits**.
*a private pointer to an object of type **Line**.
*three private arrays **vehicles** with 10 pointers to vehicles, **staff** with 20 pointers to staff and **workUnit** with 1920 pointers to work units.
*a public constructor with a pointer parameter of type **Line** to initialise the line pointer.
Also **maxVehicles-2** local busses shall be defined on heap with vehicle numbers 100, 101, 102, ..., model **"MAN Lion's City"**, maintenance every 100 driving hours with each 60 minutes duration; their addresses shall be assigned to array **vehicle**.
A long distance bus of type **LongDistanceBus**, model **"MAN Lion's Coach"**, maintenance every 1500 driving hours with 90 minutes duration and a tram model **"Siemens Avenio"**, maintenance every 6000 driving hours with 120 minutes duration shall also be defined on heap and assigned to the last two pointers in array **vehicle**.
Furthermore mechanic "Jack Fix" with staff number 5001 and "Jo Repair" with staff number 5002 shall be defined on heap and their addresses assigned to the first two pointers of array **staff**. As drivers newly defined on heap addresses of objects with consecutive staff numbers 1001, 1002, ... shall also be assigned to array **staff**; names: "Tina Quick", "Jim Street", "Max Hurry", "Maria Slow", "Mia Wheel", "Lea Rocket", "Michael Run", "Sara Rapid", "Alan Speed", "Ben Unhasty", "Nicole Fast", "Robert Rush", "Lucy Power", "John Drive", "George Beam", "Sam Jumper", "Tony Metro" and "Lara Hasty". For all these a bus driving license shall be set in a loop.
A first maintenance shall be done by "Jack Fix" for all busses, for the tram by "Jo Repair";

addresses of new objects on heap representing these working units shall be assigned to array **`workUnit`** starting with index 0 increased one by one.
Finally by calling below member functions as example or (optionally getting additional points) for the week from Monday to Saturday working units for tours of line 933 shall be planned/scheduled and at the end of the constructor all working units as well as a statistic for the week shall be outputted (see example).
*a public method with name **`print_workUnits`** without parameter and without return value outputting a list of all working units line by line as shown in example.
*a public method with name **`print_statistics`** without parameter and without return value which outputs for all vehicles and the complete staff a week statistic as shown in example.

Subtask 6

Add in main directly before return 0; the definition of an object of type LineSchedule for line 933.

Example Program Run

```
cout << line933 << endl;
line:     933
from:     Duisburg Rheindeich
to:       Uni-Nord
duration: 32 min

route of line 933
   Duisburg Rheindeich
 1 Lilienthalstrasse
 3 Javastrasse
 4 Katholische Kirche
 5 Am Schluetershof
 6 Tierheim
 8 Sperrschleuse
10 Landesarchiv NRW
12 Schwanentor
15 Friedrich-Wilhelm-Platz
17 Lehmbruck-Museum
18 Tonhallenstrasse
20 Duisburg Hbf
22 Duisburg Hbf Osteingang
23 Blumenstrasse
25 Bismarckstrasse
27 Kammerstrasse
28 Lenaustrasse
29 Nettelbeckstrasse
30 Buergerstrasse
31 Universitaet
32 Uni-Nord
```

and so on, outputs from task H4, then

```
WORK UNITS
==========

maintenance for local bus 100 by mechanic 5001 Jack Fix from 07:30 (until 08:30)
maintenance for local bus 101 by mechanic 5001 Jack Fix from 08:30 (until 09:30)
maintenance for local bus 102 by mechanic 5001 Jack Fix from 09:30 (until 10:30)
maintenance for local bus 103 by mechanic 5001 Jack Fix from 10:30 (until 11:30)
maintenance for local bus 104 by mechanic 5001 Jack Fix from 11:30 (until 12:30)
maintenance for local bus 105 by mechanic 5001 Jack Fix from 12:30 (until 13:30)
maintenance for local bus 106 by mechanic 5001 Jack Fix from 13:30 (until 14:30)
maintenance for local bus 107 by mechanic 5001 Jack Fix from 14:30 (until 15:30)
maintenance for long distance bus 108 by mechanic 5001 Jack Fix from 07:30 (until 09:00)
maintenance for tram 109 by mechanic 5002 Jo Repair from 07:30 (until 09:30)
tour at 04:18: local bus 100; driver 1001 Tina Quick (until 05:32)
tour at 04:48: local bus 101; driver 1002 Jim Street (until 06:02)
tour at 05:18: local bus 102; driver 1003 Max Hurry (until 06:32)
tour at 05:48: local bus 100; driver 1001 Tina Quick (until 07:02)
tour at 06:03: local bus 101; driver 1002 Jim Street (until 07:17)
tour at 06:18: local bus 103; driver 1004 Maria Slow (until 07:32)
tour at 06:33: local bus 102; driver 1003 Max Hurry (until 07:47)
tour at 06:48: local bus 104; driver 1005 Mia Wheel (until 08:02)
tour at 07:03: local bus 100; driver 1001 Tina Quick (until 08:17)
tour at 07:18: local bus 101; driver 1002 Jim Street (until 08:32)
tour at 07:33: local bus 103; driver 1004 Maria Slow (until 08:47)
tour at 07:48: local bus 102; driver 1003 Max Hurry (until 09:02)
tour at 08:03: local bus 104; driver 1005 Mia Wheel (until 09:17)
tour at 08:18: local bus 100; driver 1001 Tina Quick (until 09:32)
tour at 08:33: local bus 101; driver 1002 Jim Street (until 09:47)
tour at 08:48: local bus 103; driver 1004 Maria Slow (until 10:02)
tour at 09:03: local bus 102; driver 1003 Max Hurry (until 10:17)
tour at 09:18: local bus 104; driver 1005 Mia Wheel (until 10:32)
tour at 09:33: local bus 100; driver 1001 Tina Quick (until 10:47)
```

and so on, finally as example (your output may be different depending on your more or less complicated schedule generation)

```
WEEK STATISTICS
===============
Vehicles
========
local bus 100 (MAN Lion's City) driving time: 108:32(*)
local bus 101 (MAN Lion's City) driving time: 108:32(*)
local bus 102 (MAN Lion's City) driving time: 102:22(*)
local bus 103 (MAN Lion's City) driving time: 67:50
local bus 104 (MAN Lion's City) driving time: 67:50
local bus 105 (MAN Lion's City) driving time: 00:00
local bus 106 (MAN Lion's City) driving time: 00:00
local bus 107 (MAN Lion's City) driving time: 00:00
long distance bus 108 (MAN Lion's Coach) driving time: 00:00
tram 109 (Siemens Avenio) driving time: 00:00
Staff
=====
mechanic 5001 Jack Fix   working time: shift 00:00, week 09:30
mechanic 5002 Jo Repair  working time: shift 00:00, week 02:00
driver 1001 Tina Quick   working time: shift 00:00, week 39:28
driver 1002 Jim Street   working time: shift 00:00, week 39:28
driver 1003 Max Hurry    working time: shift 00:00, week 39:28
driver 1004 Maria Slow   working time: shift 00:00, week 39:28
driver 1005 Mia Wheel    working time: shift 00:00, week 39:28
driver 1006 Lea Rocket   working time: shift 00:00, week 39:28
driver 1007 Michael Run  working time: shift 00:00, week 39:28
driver 1008 Sara Rapid   working time: shift 00:00, week 39:28
driver 1009 Alan Speed   working time: shift 00:00, week 39:28
driver 1010 Ben Unhasty  working time: shift 00:00, week 39:28
driver 1011 Nicole Fast  working time: shift 00:00, week 25:54
driver 1012 Robert Rush  working time: shift 00:00, week 19:44
driver 1013 Lucy Power   working time: shift 00:00, week 12:20
driver 1014 John Drive   working time: shift 00:00, week 01:14
driver 1015 George Beam  working time: shift 00:00, week 01:14
driver 1016 Sam Jumper   working time: shift 00:00, week 00:00
driver 1017 Tony Metro   working time: shift 00:00, week 00:00
driver 1018 Lara Hasty   working time: shift 00:00, week 00:00
```

********** final date for submit the the task 7th february2017**********