Mahavir Patel
OOP-1
Assignment 2

MyParkingGarage.cpp code:

```
//
//  main.cpp
//  Class 3 Practice
//
//  Created by Mahavir Patel on 9/27/19.
//  Copyright © 2019 Mahavir Patel. All rights reserved.
//


#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <algorithm>
#include <ctime>
#include <random>
#include <numeric>
#include <cstring>
#include "Car.h"
#include "Ticket.h"

using namespace std;

int timeMinHolder (int minute) {

   if (minute >= 60) {
      minute = minute - 60;
   }

   return minute;

}

int timeHrsHolder (int minute, int hour) {

   if (minute >= 60) {
      hour++;
   }
```

```
    return hour;

}

int getDurationHrs(double ran) {

    if (ran < 0) {
        ran = -1*ran;
    }

    int total_sec = ran*60*60;

    int durationHrs = total_sec/(60*60);
    int durationMin = (total_sec - (durationHrs*60*60))/60;

    return durationHrs;
}

int getDurationMin(double ran) {

    if (ran < 0) {
        ran = -1*ran;
    }

    int total_sec = ran*60*60;

    int durationHrs = total_sec/(60*60);
    int durationMin = (total_sec - (durationHrs*60*60))/60;

    return durationMin;
}

int updatedHrs (int hours, int min, int hrs_inc, int min_inc) {

    if (hrs_inc != 0 || min_inc != 0) {
        return hrs_inc;
    }

    else {
        if(hours >= 24) {
            hours = hours - 24;
        }
        return hours;
    }
```

```
}

int updatedMin (int hours, int min, int hrs_inc, int min_inc) {

   if (hrs_inc != 0 || min_inc != 0) {
        return min_inc;
     }

   else {
      if (min >= 60) {
         min = min - 60;
      }
      return min;
   }
}

vector<Car> removeCar(vector<Car> x, vector<Ticket> y, int totalMin) {

   if(x.empty() == false) {
      for(int i = x.size()-1; i >= 0; i--) {

         int totalMinuteExit = x.at(i).getTotalExitHours()*60 + x.at(i).getTotalExitMin();

         if(totalMin > totalMinuteExit) {
            x.erase(x.begin()+i);
         }
      }
   }

   return x;

}

/*void searchQ_oneInput(vector<Car> x) {

   string mak;
   string mod;
   string col;
   string num;
   int option;

   if(x.empty() == true) {
      cout << "There are no cars in the system. ";
```

```
    }

    cout<< "Search the car. Select one of the options." << endl;
    cout << "1. Search the car by maker:\n2. Search the car by model:\n3. Search the car by
color:\n4. Search the car by number: " << endl;
    cout << "Please enter your option: ";
    cin >> option;

    if (option == 1) {
        cout << "Please enter the maker: ";
        cin >> mak;
        cout << "\n";
        for(int i = x.size()-1; i >= 0; i--) {

            if (x.at(i).getCarMaker() == mak) {
                cout << x.at(i).getCarMaker() << " " << x.at(i).getCarModel() << " " <<
x.at(i).getCarColor() << " " << x.at(i).getCarNum() << endl;
            }

        }

    }

    else if (option == 2) {

        cout << "Please enter the model: ";
        cin >> mod;
        cout << "\n";
        for(int i = x.size()-1; i >= 0; i--) {

            if (x.at(i).getCarModel() == mod) {
                cout << x.at(i).getCarMaker() << " " << x.at(i).getCarModel() << " " <<
x.at(i).getCarColor() << " " << x.at(i).getCarNum() << endl;
            }

        }

    }

    else if (option == 3) {

        cout << "Please enter the color: ";
        cin >> col;
        cout << "\n";
```

```
      for(int i = x.size()-1; i >= 0; i--) {

         if (x.at(i).getCarColor() == col) {
            cout << x.at(i).getCarMaker() << " " << x.at(i).getCarModel() << " " <<
x.at(i).getCarColor() << " " << x.at(i).getCarNum() << endl;
         }

      }

   }

   else {

      cout << "Please enter the number: ";
      cin >> col;
      cout << "\n";
      for(int i = x.size()-1; i >= 0; i--) {

         if (x.at(i).getCarColor() == col) {
            cout << x.at(i).getCarMaker() << " "
            << x.at(i).getCarModel() << " "
            << x.at(i).getCarColor() << " " << x.at(i).getCarNum() << endl;
         }

      }

   }

   cout << "\n\n";

} */

void searchQ_multipleInputs(vector<Car> x) {

   string options;
   string maker{""}, model{""}, color{""}, number{""};

   cout<< "You can select one or multiple option(s)." << endl;
   cout << "1. Search the car by maker.\n2. Search the car by model.\n3. Search the car by
color.\n4. Search the car by number. " << endl;
   cout << "Enter your option(s) without any space: ";
   cin >> options;
```

```
   size_t first = options.find("1"), second = options.find("2"), third = options.find("3"), fourth =
options.find("4");

   if(first != string::npos) {
      cout << "1. Enter the car maker: ";
      cin >> maker;
   }

   if(second != string::npos) {
      cout << "2. Enter the car model: ";
      cin >> model;
   }

   if(third != string::npos) {
      cout << "3. Enter the car color: ";
      cin >> color;
   }

   if(fourth != string::npos) {
      cout << "4. Enter the car number: ";
      cin >> number;
   }

   for(Car theCar: x) {

      if((maker == theCar.getCarMaker() || maker == "") && (model == theCar.getCarModel() ||
model == "") &&
         (color == theCar.getCarColor() || color == "") && (number == theCar.getCarNum() ||
number == "")) {

          cout << theCar.getCarMaker() << " "
          << theCar.getCarModel() << " "
          << theCar.getCarColor() << " " << theCar.getCarNum() << endl;
       }

   }

}

int main() {


   default_random_engine engine{static_cast<unsigned int>(time(0))};
   normal_distribution<float> destribution(5.0,2.0);
```

```
vector<Car> myCar;
vector<Ticket> myTicket;
vector<int> ticketTotal;
Car carptr;
Ticket ticketPtr;
bool done{false};
int choice;
double ran;
int total;
string query;
string maker{""}, model{""}, color{""}, number{""};
time_t timeEnter = time(0);
tm *ltm = localtime(&timeEnter);

int totalStay{0};

int tm_hrs;
int tm_min;

int time_hrs_inc = ltm->tm_hour;
int time_min_inc = ltm-> tm_min;

int time_min_holder;

int tm_hrs_inc_ptr;
int tm_min_inc_ptr;

int total_min_after_inc;

int pricePtr;

while(!done) {

   cout << "1. Print-out the car information in the garage." << endl;
   cout << "2. Add a car." << endl;
   cout << "3. Increment time by 30 minutes." << endl;
   cout << "4. Search a particualr car in the garrage." << endl;
   cout << "5. Exit" << endl;

   cout << "Selecet you option: ";
   cin >> choice;

   switch (choice) {
```

```
        case 1:

            cout << "\n\nCars in the system: " << endl;
            for (Car theCar: myCar) {
            cout<< "Enter Time: " << theCar.getEnterHour() << ":" << theCar.getEnterMin()
               << "   " << theCar.getCarMaker()
               << "   " << theCar.getCarModel()
               << "   " << theCar.getCarColor()
               << "   " << theCar.getCarNum()
               << "   " << "ExtiTimetobe: " << theCar.getExitHour() << ":" << theCar.getExitMin() << "
" << "Total-Charge: " << theCar.getTicket() << endl;
            }


            cout << "\n\n";
            break;

        case 2:
            timeEnter = time(0);
            ltm = localtime(&timeEnter);
            tm_hrs = updatedHrs(ltm->tm_hour, ltm->tm_min, tm_hrs_inc_ptr, tm_min_inc_ptr);
            tm_min = updatedMin(ltm->tm_hour, ltm->tm_min, tm_hrs_inc_ptr,
tm_min_inc_ptr);
            carptr.setEnterHour(tm_hrs);
            carptr.setEnterMin(tm_min);
            cout << "Enter the maker: ";
            cin >> maker;
            carptr.setCarMaker(maker);
            cout << "Enter the model: ";
            cin >> model;
            carptr.setCarModel(model);
            cout << "Enter the color: ";
            cin >> color;
            carptr.setCarColor(color);
            cout << "Enter the number: ";
            cin >> number;
            carptr.setCarNum(number);
            ran = destribution(engine);
            carptr.setDurationHour(ran);
            carptr.setDurationMin(ran);
            carptr.setExitHour(ran, tm_hrs, tm_min);
            carptr.setExitMin(ran, tm_hrs, tm_min);
            ticketPtr.set_ticket(carptr.getDurationHour(), carptr.getDurationMin());
```

```
            carptr.setTicket(ticketPtr.get_ticket());

            cout << "\n\n";
            myCar.push_back(carptr);
            //myTicket.push_back(ticketPtr);
            ticketTotal.push_back(carptr.getTicket());
            time_hrs_inc = updatedHrs(ltm->tm_hour, ltm->tm_min, tm_hrs_inc_ptr,
tm_min_inc_ptr);
            time_min_inc = updatedMin(ltm->tm_hour, ltm->tm_min, tm_hrs_inc_ptr,
tm_min_inc_ptr);
            break;

        case 3:
            carptr.setTimeInc(time_hrs_inc, time_min_inc);
            cout << "Your time is increamented. The new time is: " << carptr.getTimeInc() << endl;
            time_min_inc = time_min_inc + 30;
            time_min_holder = time_min_inc;
            time_min_inc = timeMinHolder(time_min_inc);
            time_hrs_inc = timeHrsHolder(time_min_holder, time_hrs_inc);
            tm_hrs_inc_ptr = time_hrs_inc;
            tm_min_inc_ptr = time_min_inc;
            total_min_after_inc = tm_hrs_inc_ptr*60 + tm_min_inc_ptr;
            myCar = removeCar(myCar, myTicket, total_min_after_inc);
            cout << "\n\n";
            break;

        case 4:

            cout << "\n\nSearch the car.";
            //searchQ_oneInput(myCar);
            searchQ_multipleInputs(myCar);
            cout << "\n\n";
            break;

        case 5:

            done = true;
            for (Car theCar: myCar) {
               cout<< "Enter Time: " << theCar.getEnterHour() << ":" << theCar.getEnterMin()
                  << "  " << theCar.getCarMaker()
                  << "  " << theCar.getCarModel()
                  << "  " << theCar.getCarColor()
                  << "  " << theCar.getCarNum()
```

```
                << "  " << "Exti Time to be: " << theCar.getExitHour() << ":" << theCar.getExitMin()
<< " " << "Total-Charge: " << theCar.getTicket() << endl;
            }

            cout << "The total is: " << accumulate(ticketTotal.begin(), ticketTotal.end(), 0) << endl;

        }
    }

}
```

Car.h code:

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <algorithm>
#include <ctime>
#include <random>


using namespace std;

class Car {

private:
   // for the car
   string maker;
   string num;
   string model;
   string color;

   //for the time
   string enteringTime;
   string exitTime;
   string duration;
   string timeInc;

   //for the enter time
   int h;
   int m;

   //for time increament
   int time_hrs_inc;
   int time_min_inc;
   int time_sec_inc;

   //for the exit time
   double total_seconds;
   int hour;
   int minute;
   int second;
   int totalHrs;
   int totalmin;
```

```
    int totalsec;
    int totalHrsTemp;
    int totalMinTemp;

    //for the duration
    double total_sec;
    int durationHrs;
    int durationMin;
    int durationSec;

    //for the tickets
    int charge;

public:

    void setCarMaker(string carMaker) {
        maker = carMaker;
    }

    string getCarMaker() const {
        return maker;
    }

    void setCarNum(string carNum) {
        num = carNum;
    }

    string getCarNum() const {
        return num;
    }

    void setCarModel(string carModel) {
        model = carModel;
    }

    string getCarModel() const {
        return model;
    }

    void setCarColor(string carColor) {
        color = carColor;
    }

    string getCarColor() const {
```

```
   return color;
}

void setEnterTime(int hour, int minute) {
   enteringTime = to_string(hour) + ":" + to_string(minute);
}

string getEnterTime() const {
   return enteringTime;
}

void setDurationHour(double ran) {

   if (ran < 0) {
      ran = -1*ran;
   }

   total_sec = ran*60*60;

   durationHrs = total_sec/(60*60);
   durationMin = (total_sec - (durationHrs*60*60))/60;

}

int getDurationHour() const {
   return durationHrs;
}

void setDurationMin(double ran) {

   if (ran < 0) {
      ran = -1*ran;
   }

   total_sec = ran*60*60;

   durationHrs = total_sec/(60*60);
   durationMin = (total_sec - (durationHrs*60*60))/60;

}

int getDurationMin() const {
   return durationMin;
}
```

```
string getDuration() const {
   return duration;
}

void setTimeInc(int tm_hrs, int tm_min) {

   tm_min = tm_min + 30;

   if (tm_min >= 60) {
      tm_min = tm_min - 60;
      tm_hrs++;
   }

   if (tm_hrs >= 24) {
      tm_hrs = tm_hrs - 24;
   }

   timeInc = to_string(tm_hrs) + ":" + to_string(tm_min);

}

string getTimeInc() const {
   return timeInc;
}

void setEnterHour(int hour) {
   h = hour;
}

int getEnterHour() const {
   return h;
}

void setEnterMin(int minute) {
   m = minute;
}

int getEnterMin() const {
   return m;
}

void setExitHour(double random, int tm_hrs, int tm_min) {
```

```
   if (random < 0) {
      random = -1*random;
   }

   total_seconds = random*60*60;

   hour = total_seconds/(60*60);
   minute = (total_seconds - (hour*60*60))/60;

   totalHrs = hour + tm_hrs;
   totalmin = minute + tm_min;
   totalHrsTemp = totalHrs;

   if(totalmin >= 60) {
      totalmin = totalmin - 60;
      totalHrs++;
   }

   if(totalHrs >= 24) {
      totalHrs = totalHrs - 24;
   }

}

int getExitHour() const {
   return totalHrs;
}


void setExitMin(double random, int tm_hrs, int tm_min) {

if (random < 0) {
   random = -1*random;
}

total_seconds = random*60*60;

hour = total_seconds/(60*60);
minute = (total_seconds - (hour*60*60))/60;

totalHrs = hour + tm_hrs;
totalmin = minute + tm_min;
totalMinTemp = totalmin;
```

```
    if(totalmin >= 60) {
       totalmin = totalmin - 60;
       totalHrs++;
    }

    if(totalHrs >= 24) {
       totalHrs = totalHrs - 24;
    }

    }

    int getExitMin() const {
       return totalmin;
    }

    void setTicket(int price) {
       charge = price;
    }

    int getTicket() const {
       return charge;
    }

    int getTotalExitHours() const {
       return totalHrsTemp;
    }

    int getTotalExitMin() const {
       return totalMinTemp;
    }
};
```

Ticket.h Code:

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <string>
#include <algorithm>
#include <ctime>
#include <random>


using namespace std;

class Ticket {

public:

   void set_ticket(int durationHrs, int durationMin) {

      if(durationMin != 0) {
         durationHrs++;
      }

      if(durationHrs <= 3) {
         price = 4;
      }

      if(durationHrs > 3 && durationHrs <= 9) {
         price = 4 + (durationHrs - 3);
      }

      if(durationHrs > 9) {
         price = 10;
      }

   }

   int get_ticket() const {
      return price;
   }

private:

   int price;
```

};

Output:

```
1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 2
Enter the maker: honda
Enter the model: accord
Enter the color: purple
Enter the number: 1234


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 2
Enter the maker: lexus
Enter the model: iota
Enter the color: green
Enter the number: 5647


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 1


Cars in the system:
Enter Time: 16:2   honda   accord   purple   1234   ExtiTimetobe: 18:28   Total-Charge: 4
Enter Time: 16:2   lexus   iota   green   5647   ExtiTimetobe: 17:20   Total-Charge: 4


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 3
Your time is increamented. The new time is: 16:32


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 1
```

```
Cars in the system:
Enter Time: 16:2   honda   accord   purple   1234   ExtiTimetobe: 18:28   Total-Charge: 4
Enter Time: 16:2   lexus   iota   green   5647   ExtiTimetobe: 17:20   Total-Charge: 4


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 3
Your time is increamented. The new time is: 17:2


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 2
Enter the maker: lexus
Enter the model: imagica
Enter the color: silver
Enter the number: 4590


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 1


Cars in the system:
Enter Time: 16:2   honda   accord   purple   1234   ExtiTimetobe: 18:28   Total-Charge: 4
Enter Time: 16:2   lexus   iota   green   5647   ExtiTimetobe: 17:20   Total-Charge: 4
Enter Time: 17:2   lexus   imagica   silver   4590   ExtiTimetobe: 18:3   Total-Charge: 4
```

```
1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 3
Your time is increamented. The new time is: 17:32


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 1


Cars in the system:
Enter Time: 16:2   honda   accord   purple   1234   ExtiTimetobe: 18:28   Total-Charge: 4
Enter Time: 17:2   lexus   imagica   silver   4590   ExtiTimetobe: 18:3   Total-Charge: 4
```

```
1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 4


Search the car.You can select one or multiple option(s).
1. Search the car by maker.
2. Search the car by model.
3. Search the car by color.
4. Search the car by number.
Enter your option(s) without any space: 13
1. Enter the car maker: honda
3. Enter the car color: purple
honda accord purple 1234


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 4


Search the car.You can select one or multiple option(s).
1. Search the car by maker.
2. Search the car by model.
3. Search the car by color.
4. Search the car by number.
Enter your option(s) without any space: 24
2. Enter the car model: imagica
4. Enter the car number: 4590
lexus imagica silver 4590


1. Print-out the car information in the garage.
2. Add a car.
3. Increment time by 30 minutes.
4. Search a particualr car in the garrage.
5. Exit
Selecet you option: 5
Enter Time: 16:2  honda  accord  purple  1234  Exti Time to be: 18:28 Total-Charge: 4
Enter Time: 17:2  lexus  imagica  silver  4590  Exti Time to be: 18:3 Total-Charge: 4
The total is: 12
Program ended with exit code: 0
```