Parking Ticket Simulator

Hissamuddin Shaikh

CSCI 201

30th of June 2018

St. Cloud State University

**Parking Ticket Simulator**

**Problem Statement**:

Create a program that utilizes Four Classes to simulate a Police Officer issuing a Parking Ticket for a Vehicle that has been parked without payment, or for more than the paid amount of time.

**Input/Output Description:**

Input: The Make of Car

The Model of the Car

The Color of the Car

The License Number of the Car

Number of Minutes the Car has been parked

Number of Minutes Purchased by the Car Owner

The First Name of the Officer issuing the Ticket

The Last Name of the Officer issuing the Ticket

The Badge Number of the Officer issuing the Ticket

Output: The Make of Car

The Model of the Car

The Color of the Car

The License Number of the Car

The First Name of the Officer issuing the Ticket

The Last Name of the Officer issuing the Ticket

The Badge Number of the Officer issuing the Ticket

The Amount of the Fine issued (if any).

**Variables:**

* ***Variables for the Parked Car (ParkedCar) class:***
  + *carMake*
    - A private variable constructed to store the Make of the Parked Car.
    - Type: string
  + *carModel*
    - A private variable constructed to store the Model of Parked Car.
    - Type: string
  + *carColor*
    - A private variable constructed to store the Color of the Parked Car.
    - Type: string
  + *carLicenseNum*
    - A private variable constructed to store the License Number of the Car.
    - Type: string
  + *numMinutesParked*
    - A private variable constructed to store the Number of Minutes that the Car has been Parked for.
    - Type: int
  + cMake
    - A variable used in the parametrical constructor for the Class, for the information about the Make of the Parked Car.
    - Type: string
  + cModel
    - A variable used in the parametrical constructor for the Class, for the information about the Model of the Parked Car.
    - Type: string
  + cColor
    - A variable used in the parametrical constructor for the Class, for the information about the Color of the Parked Car.
    - Type: string
  + cLicenseNum
    - A variable used in the parametrical constructor for the Class, for the information about the License Number of the Parked Car.
    - Type: string
  + cNumMinParked
    - A variable used in the parametrical constructor for the Class, for the information about the Number of Minutes that the Car has been Parked for.
    - Type: int
* ***Variables for the Parking Meter (ParkingMeter) class:***
  + *purchasedParkingMins*
    - A private variable constructed to store the Number of Minutes that the Car Owner has paid for.
    - Type: int
  + *purchasedMinutes*
    - A variable used in the parametrical constructor for the Class, for the information about the Number of Minutes that the Car Owner has paid for.
    - Type: int
* ***Variables for the Parking Ticket (ParkingTicket) class:***
  + *Car*
    - Object to access the class *ParkedCar*.
  + *Meter*
    - Object to access the class *ParkingMeter*.
  + *Officer*
    - Object to access the class *PoliceOfficer*.
  + *fineAmount*
    - A private variable constructed to store the Amount of Fine issued to the Car Owner.
    - Type: int
  + *carT*
    - An object used in the parametrical constructor for the Class, for the information about the class *ParkedCar*.
  + *meterT*
    - An object used in the parametrical constructor for the Class, for the information about the class *ParkingMeter*.
  + *officerT*
    - An object used in the parametrical constructor for the Class, for the information about the class *PoliceOfficer*.
* ***Variables for the Police Officer (PoliceOfficer) class:***
  + *lastName*
    - A private variable constructed to store the Police Officers Last Name.
    - Type: string
  + *firstName*
    - A private variable constructed to store the Police Officers First Name.
    - Type: string
  + *badgeNum*
    - A private variable constructed to store the Police Officers Badge Number.
    - Type: string
  + *lName*
    - A variable used in the parametrical constructor for the Class, for the information about the Last Name of the Police Officer.
    - Type: string
  + *fName*
    - A variable used in the parametrical constructor for the Class, for the information about the First Name of the Police Officer.
    - Type: string
  + *bNum*
    - A variable used in the parametrical constructor for the Class, for the information about the Badge Number of the Police Officer.
    - Type: string
  + *C*
    - An object used in the function *isTicketNecessary*, which calculates if a ticket needs to be issued, for the information about the class *ParkedCar*.
  + *M*
    - An object used in the function *isTicketNecessary*, which calculates if a ticket needs to be issued, for the information about the class *ParkingMeter*.
* ***Variables for the Main Program (Function):***
  + *carMake*
    - A variable to store the Make of the Parked Car, being inspected by the Police Officer.
    - Type: string
  + *carModel*
    - A variable to store the Model of the Parked Car, being inspected by the Police Officer.
    - Type: string
  + *carColor*
    - A variable to store the Color of the Parked Car, being inspected by the Police Officer.
    - Type: string
  + *carLicenseNum*
    - A variable to store the License Number of the Parked Car, being inspected by the Police Officer.
    - Type: string
  + *lastName*
    - A variable to store the Last Name of the Police Officer, inspecting the Parked Car.
    - Type: string
  + *firstName*
    - A variable to store the First Name of the Police Officer, inspecting the Parked Car.
    - Type: string
  + *badgeNum*
    - A variable to store the Last Name of the Police Officer, inspecting the Parked Car.
    - Type: string
  + *numMinutesParked*
    - A variable to store the Number of Minutes that a Car has been parked for, in a Spot.
    - Type: int
  + *purchasedParkingMins*
    - A variable to store the Number of Minutes that the Owner has paid for, to park at the spot.
    - Type: int
  + *Car*
    - Object to access the class *ParkedCar*
  + Meter
    - Object to access the class *Parkingmeter*
  + Officer
    - Object to access the class *PoliceOfficer*

**Program Design:**

1. Prompt the Police Officer to insert the Car Make of the Parked Car.
2. Prompt the Police Officer to insert the Car Model of the Parked Car.
3. Prompt the Police Officer to insert the Car Color of the Parked Car.
4. Prompt the Police Officer to insert the Car License Number of the Parked Car.
5. Prompt the Police Officer to Insert the Number of Minutes that the Car has been parked.
6. Prompt the Police Officer to Insert the Number of Minutes that the Car Owner has paid for.
7. Prompt the Police Officer to Insert his/her First Name.
8. Prompt the Police Officer to Insert his/her Last Name.
9. Prompt the Police Officer to Insert his/her Badge Number.
10. Calculate if a Ticket needs to be issued or not.
11. If a Ticket needs to be issued, Print the Ticket. Otherwise, don’t issue a ticket and end the Program.

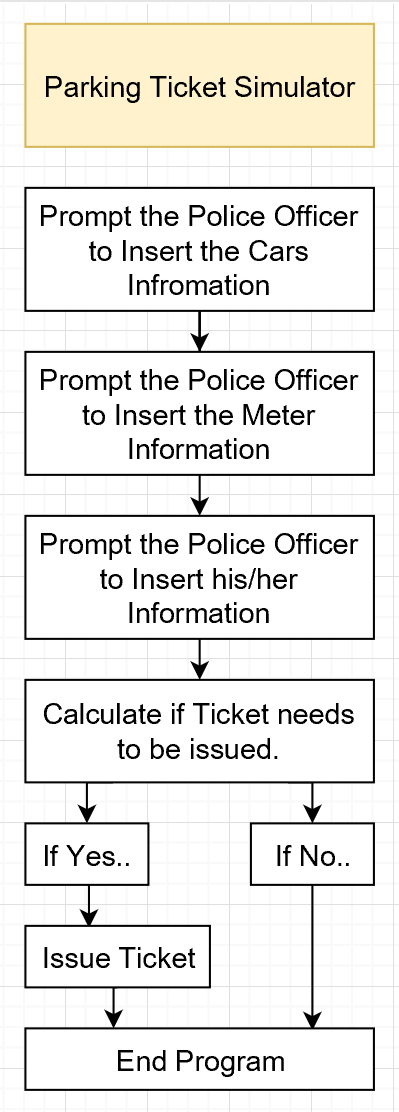
 **Flowchart:**

Figure. A

**Algorithm Development (Detailed Pseudocode):**

* Prompt the Police Officer to inset the Make, Model, Color and License Number of the Parked Car through a Menu, that looks like the following:
  + Parking Ticket Simulator
  + -------------------------------------
  + Car Information
  + -------------------------------------
  + Car Make:
    - Store this Information in the variable *carMake*.
  + Car Model:
    - Store this Information in the variable *carModel*.
  + Car Color:
    - Store this Information in the variable *carColor*.
  + Car License Number:
    - Store this Information in the variable *carLicenseNum*.
* Setup a do-While loop that includes the following:
  + Prompt the Police Officer to insert the Number of Minutes that the Car has been Parked at the Spot, through the statement, “Number of Minutes the Car has been Parked: “. Store this information in *numMinutesParked*.
* End the do-While loop when *numMinutesParked* is greater than 0. (This do-While loop is a form of Error Control)
* Set *Car* as the object to access the class *ParkedCar*.
* Pass the values in the variables *carMake*, *carModel*, *carColor*, *carLicenseNumber* and *numMinutesParked* into *Car* (Parametrical Constructor).
* Prompt the Police Officer to insert the Number of Minutes that the Car Owner has paid for, (if any) through a Menu that looks like the following:
  + -------------------------------------
  + Meter Information
  + -------------------------------------
    - Setup a do-While loop that includes the following:
      * Displays “Number of Minutes Purchased:”, stores this information in the variable *purchasedParkingMins*.
    - End the do-While loop when the value stored in *purchasedParkingMins* is greater than 0. (This do-While loop is a form of Error Control).
* Set *Meter* as the object the access the class *ParkingMeter*.
* Pass the value in the variable *purchasedParkingMins* into *Meter* (Parametrical Constructor).
* Prompt the Police Officer to insert his/her First and Last Name, as well as his/her Badge Number through a Menu, that looks like the following:
  + -------------------------------------
  + Police Officer Information
  + -------------------------------------
  + First Name:
    - Store this Information in the variable *firstName*.
  + Last Name:
    - Store this Information in the variable *lastName*.
  + Badge Number:
    - Store this Information in the variable *badgeNum*.
* Set *Officer* as the object to access the class *PoliceOfficer*.
* Pass the values in the variables *firstName lastName* and *badgeNum* into *Officer* (Parametrical Constructor).
* Run the information in the classes (object) *Car* and *Meter*, through the function *Officer.isTicketNeccessary(Car, Meter)* in an if-statement to see if a Ticket needs to be issued.
  + Set *Ticket* as the object to access the class *ParkingTicket*
  + If a ticket needs to be issued, run the information in the classes (object) *Car*, *Meter*, and *Officer*, through the function ParkingTicket Ticket(Car, Meter, Officer).
  + Display the result of the Ticket through the function *Ticket.print().*
  + Otherwise, issue no Ticket and display the following:
    - -------------------------------------
    - No Ticket Issued
    - -------------------------------------

**Program Listing:-**

***ParkedCar.h:***

1. #ifndef PARKEDCAR\_H
2. #define PARKEDCAR\_H
3. #include <string>
4. using namespace std;
5. class ParkedCar
6. {
7. public:
8. /\*Default Constructor\*/
9. ParkedCar();
10. /\*Constructor with Parameters\*/
11. ParkedCar(string cMake, string cModel, string cColor, string cLicenseNum, int cNumMinParked);
12. /\*Function to retrieve the information in numMinutesParked\*/
13. int getNumParkedMinutes() const;
14. /\*Function to display the values in all Private Variables\*/
15. string print();
16. private:
17. /\*Stores the Make of the Car\*/
18. string carMake;
19. /\*Stores the Model of the Car\*/
20. string carModel;
21. /\*Stores the Color of the Car\*/
22. string carColor;
23. /\*Stores the License Number of the Car\*/
24. string carLicenseNum;
25. /\*Stores the Number of Minutes that the Car has been parked for\*/
26. int numMinutesParked;
27. };
28. #endif // PARKEDCAR\_H

***ParkedCar.cpp:***

1. #include "ParkedCar.h"
2. using namespace std;
3. ParkedCar::ParkedCar()
4. {
5. carMake = "";
6. carModel = "";
7. carColor = "";
8. carLicenseNum = "";
9. numMinutesParked = 0;
10. }
11. ParkedCar::ParkedCar(string cMake, string cModel, string cColor, string cLicenseNum, int cNumMinParked)
12. {
13. carMake = cMake;
14. carModel = cModel;
15. carColor = cColor;
16. carLicenseNum = cLicenseNum;
17. numMinutesParked = cNumMinParked;
18. }
19. int ParkedCar::getNumParkedMinutes() const
20. {
21. return numMinutesParked;
22. }
23. string ParkedCar:: print()
24. {
25. string carDescription = " - Car - \n";
26. carDescription += " Make: " + carMake;
27. carDescription += "\n Model: " + carModel;
28. carDescription += "\n Color: " + carColor;
29. carDescription += "\n License Number: " + carLicenseNum;
30. return carDescription;
31. }

***ParkingMeter.h:***

1. #ifndef PARKINGMETER\_H
2. #define PARKINGMETER\_H
3. #include <string>
4. using namespace std;
5. class ParkingMeter
6. {
7. public:
8. /\*Default Constructors\*/
9. ParkingMeter();
10. /\*Constructor with Parameters\*/
11. ParkingMeter(int purchasedMinutes);
12. /\*Function to retrieve the value stored in purchasedParkingMins\*/
13. int getPurchasedParkingMins() const;
14. /\*Function to display the values in all Private Variables\*/
15. string print();
16. private:
17. /\*Stores the Number of Minutes the Car Owner has paid for\*/
18. int purchasedParkingMins;
19. };
20. #endif // PARKINGMETER\_H

***ParkingMeter.cpp:***

1. #include "ParkingMeter.h"
2. #include <string>
3. using namespace std;
4. ParkingMeter::ParkingMeter()
5. {
6. purchasedParkingMins = 0;
7. }
8. ParkingMeter::ParkingMeter(int purchasedMinutes)
9. {
10. purchasedParkingMins = purchasedMinutes;
11. }
12. int ParkingMeter::getPurchasedParkingMins() const
13. {
14. return purchasedParkingMins;
15. }
16. string ParkingMeter::print()
17. {
18. return "ParkingMeter print()";
19. }

***ParkingTicket.h:***

1. #ifndef PARKINGTICKET\_H
2. #define PARKINGTICKET\_H
3. #include "ParkedCar.h"
4. #include "ParkingMeter.h"
5. #include "PoliceOfficer.h"
6. #include <string>
7. using namespace std;
8. class ParkingTicket
9. {
10. public:
11. /\*Constructor with Parameters\*/
12. ParkingTicket(ParkedCar &carT, ParkingMeter &meterT, PoliceOfficer &officerT);
13. /\*Function to retrieve the calculated value of the Fine issued \*/
14. int calcFineAmount();
15. /\*Function to display the values in all Private Variables\*/
16. string print();
17. private:
18. /\*Object to access the class ParkedCar\*/
19. ParkedCar Car;
20. /\*Object to access the class ParkingMeter\*/
21. ParkingMeter Meter;
22. /\*Object to access the class PoliceOfficer\*/
23. PoliceOfficer Officer;
24. /\*Stores The Amount of Fine issued to the Car Owner\*/
25. int fineAmount;
26. };
27. #endif // PARKINGTICKET\_H

***ParkingTicket.cpp:***

1. #include "ParkingTicket.h"
2. #include "ParkedCar.h"
3. #include "ParkingMeter.h"
4. #include "PoliceOfficer.h"
5. #include <string>
6. #include <cmath>
7. #include <cstdio>
8. #include <sstream>
9. using namespace std;
10. ParkingTicket::ParkingTicket(ParkedCar &carT, ParkingMeter &meterT, PoliceOfficer &officerT)
11. {
12. Car = carT;
13. Meter = meterT;
14. Officer = officerT;
15. fineAmount = calcFineAmount();
16. }
17. int ParkingTicket::calcFineAmount()
18. {
19. return (25 + 10 \* (ceil((Car.getNumParkedMinutes()- Meter.getPurchasedParkingMins())/60.0) - 1));
20. }
21. string ParkingTicket::print()
22. {
23. ostringstream fineAmountStr;
24. fineAmountStr << " Amount: $" << fineAmount;
25. string ticketDescription = "";
26. ticketDescription += "\n -------------------------------------\n";
27. ticketDescription += " \*\*\*\*\*\*\*\*\*\*\* PARKING TICKET \*\*\*\*\*\*\*\*\*\*\n";
28. ticketDescription += " -------------------------------------\n";
29. ticketDescription += Car.print();
30. ticketDescription += "\n -------------------------------------\n";
31. ticketDescription += Officer.print();
32. ticketDescription += "\n -------------------------------------\n";
33. ticketDescription += " - Fine - \n\n";
34. ticketDescription += fineAmountStr.str();
35. ticketDescription += "\n -------------------------------------\n";
36. return ticketDescription;
37. }

***PoliceOfficer.h:***

1. #ifndef POLICEOFFICER\_H
2. #define POLICEOFFICER\_H
3. #include "ParkedCar.h"
4. #include "ParkingMeter.h"
5. #include <string>
6. using namespace std;
7. class PoliceOfficer
8. {
9. public:
10. /\*Default Constructor\*/
11. PoliceOfficer();
12. /\*Constructor with Parameters\*/
13. PoliceOfficer(string lName, string fName, string bNum);
14. /\*Function to calculate whether or not a Ticket should be issued\*/
15. bool isTicketNeccessary(ParkedCar& C, ParkingMeter& M);
16. /\*Function to display the values in all Private Variables\*/
17. string print();
18. private:
19. /\*Stores the Police Officers Last Name\*/
20. string lastName;
21. /\*Stores the Police Officers First Name\*/
22. string firstName;
23. /\*Stores the Police Officers Badge Name\*/
24. string badgeNum;
25. };
26. #endif // POLICEOFFICER\_H

***PoliceOfficer.cpp:***

1. #include "PoliceOfficer.h"
2. using namespace std;
3. PoliceOfficer::PoliceOfficer()
4. {
5. lastName = "";
6. firstName = "";
7. badgeNum = "";
8. }
9. PoliceOfficer::PoliceOfficer(string lName, string fName, string bNum)
10. {
11. lastName = lName;
12. firstName = fName;
13. badgeNum = bNum;
14. }
15. bool PoliceOfficer::isTicketNeccessary(ParkedCar& C, ParkingMeter& M)
16. {
17. if ((M.getPurchasedParkingMins() - C.getNumParkedMinutes()) < 0)
18. {
19. return true;
20. }
21. else
22. {
23. return false;
24. }
25. }
26. string PoliceOfficer::print()
27. {
28. string officerDescription = " - Police Officer - \n";
29. officerDescription += "\n First Name: " + firstName;
30. officerDescription += "\n Last Name: " + lastName;
31. officerDescription += "\n Badge Number: " + badgeNum;
32. return officerDescription;
33. }

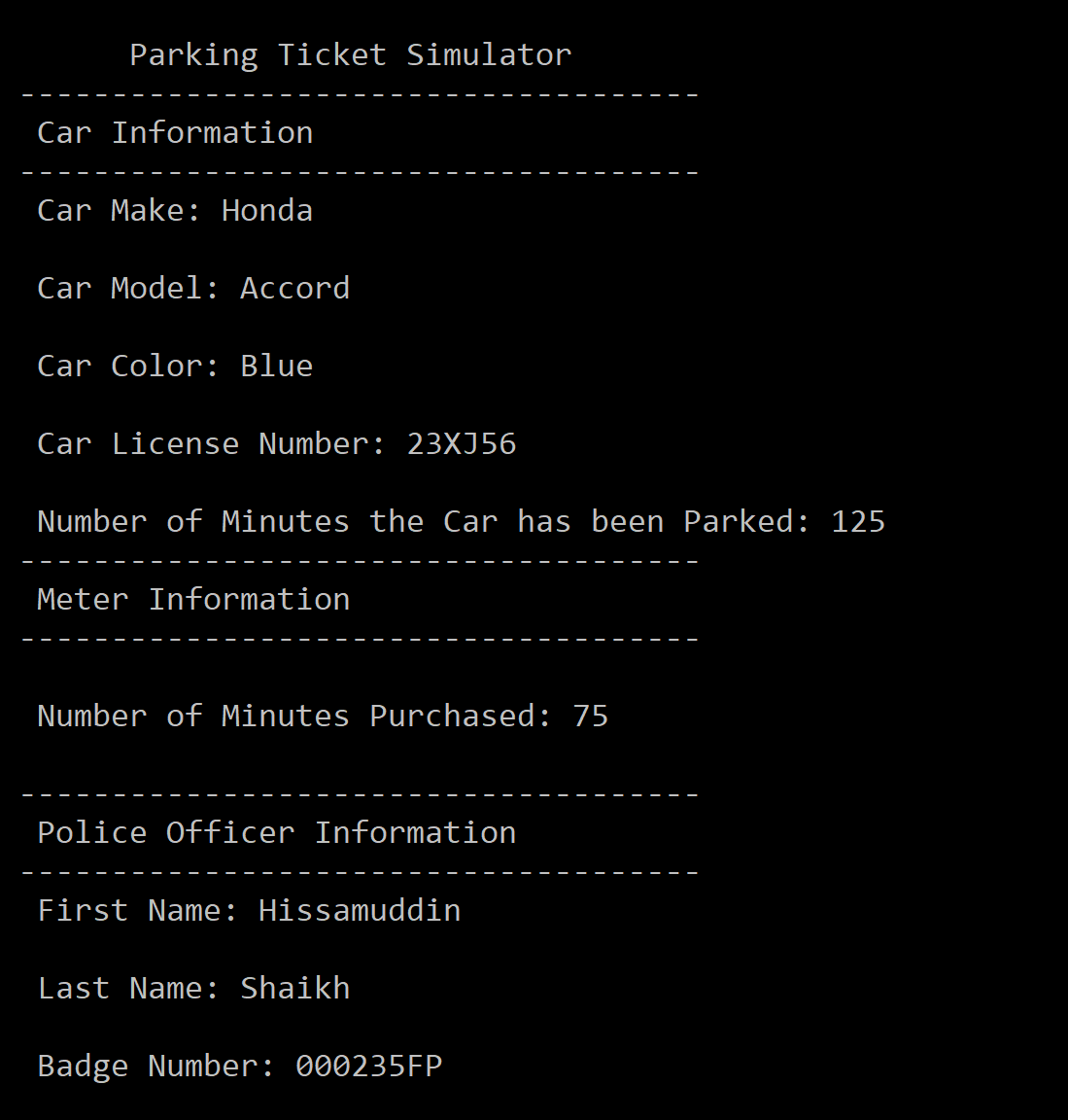
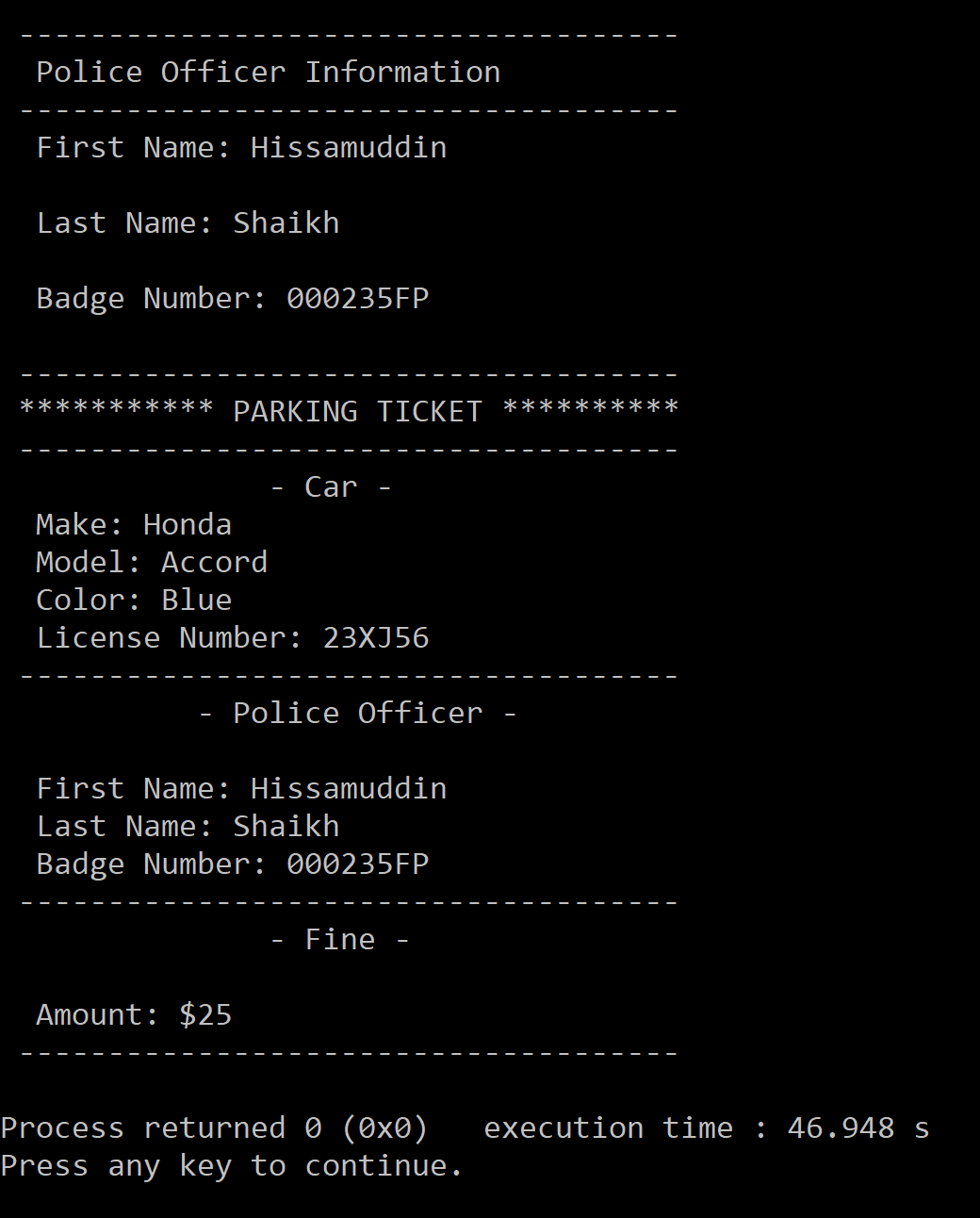
***Main Program:***

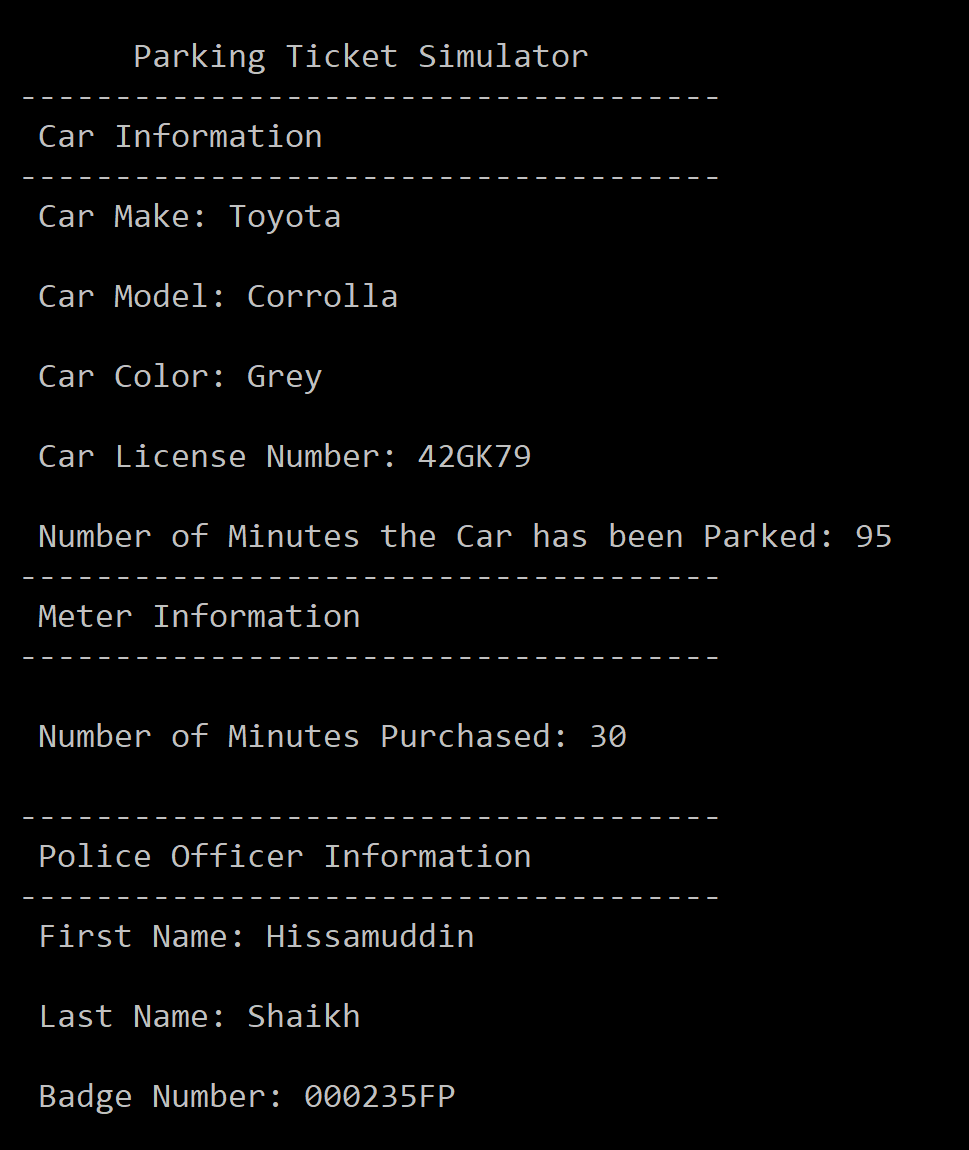
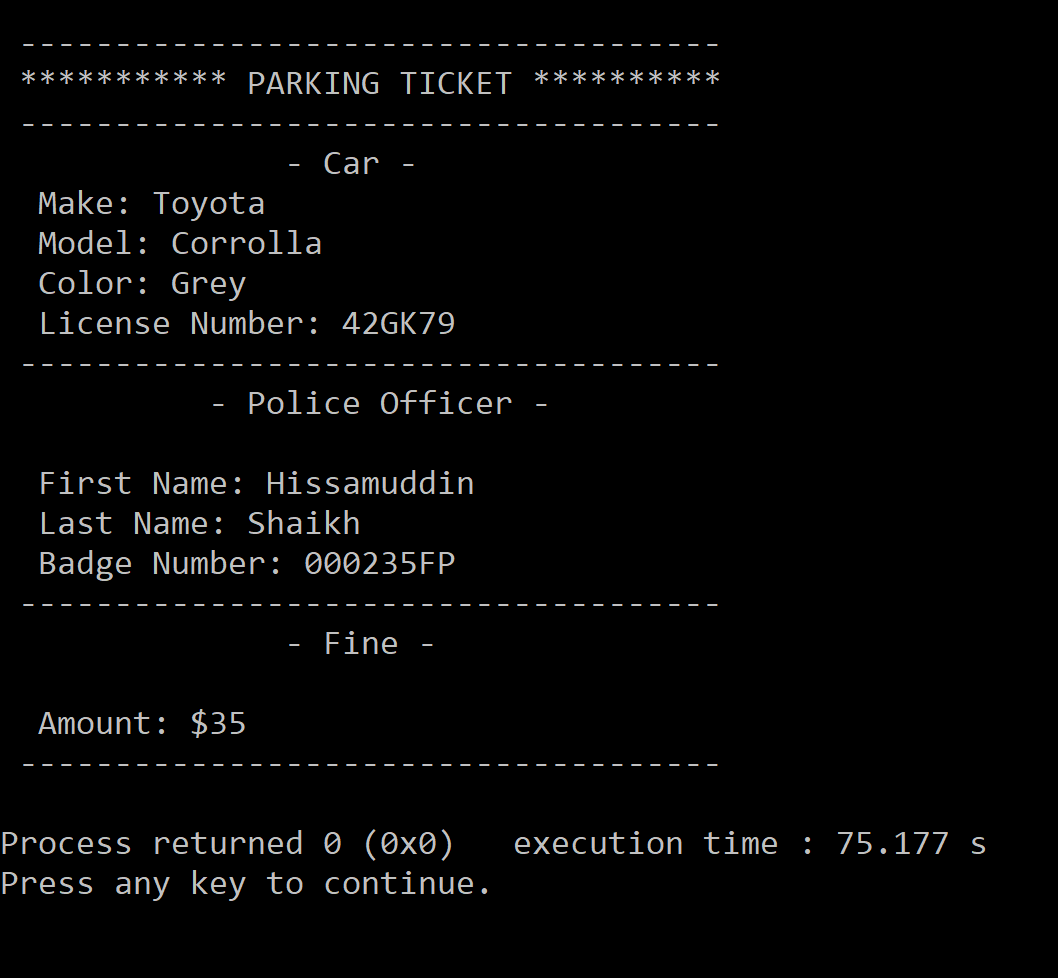
1. /\*ParkingTicketSimulator by Hissamuddin Shaikh\*/
2. #include <iostream>
3. #include <cmath>
4. #include <iomanip>
5. #include <string>
6. #include "ParkedCar.h"
7. #include "ParkingMeter.h"
8. #include "ParkingTicket.h"
9. #include "PoliceOfficer.h"
10. using namespace std;
11. int main()
12. {
13. /\*Variable Setup\*/
14. string carMake;
15. string carModel;
16. string carColor;
17. string carLicenseNum;
18. string lastName;
19. string firstName;
20. string badgeNum;
21. int numMinutesParked;
22. int purchasedParkingMins;
23. /\*Setup for Car Information Input\*/
24. cout << endl << setw(31) << "Parking Ticket Simulator" << endl;
25. cout << " -------------------------------------" << endl;
26. cout << setw(17) << "Car Information" << endl;
27. cout << " -------------------------------------" << endl;
28. cout << setw(12) << "Car Make: ";
29. cin >> carMake;
30. cout << endl << setw(13) << "Car Model: ";
31. cin >> carModel;
32. cout << endl << setw(13) << "Car Color: ";
33. cin >> carColor;
34. cout << endl << setw(22) << "Car License Number: ";
35. cin >> carLicenseNum;
36. /\*Setup for Input of Minutes Car has been Parked\*/
37. do
38. {
39. cout << endl << setw(45) << "Number of Minutes the Car has been Parked: ";
40. cin >> numMinutesParked;
41. } while (numMinutesParked < 0);
42. /\*Setup for passing information to the class ParkedCar\*/
43. ParkedCar Car(carMake, carModel, carColor, carLicenseNum, numMinutesParked);
44. /\*Setup for Meter Information Retrieval\*/
45. cout << " -------------------------------------" << endl;
46. cout << setw(19) << "Meter Information" << endl; // Get meter's information ...
47. cout << " -------------------------------------" << endl << endl;
48. do
49. {
50. cout << setw(31) << "Number of Minutes Purchased: " ;
51. cin >> purchasedParkingMins;
52. }while (purchasedParkingMins < 0);
53. ParkingMeter Meter(purchasedParkingMins);
54. /\*Setup for Police Officer Information\*/
55. cout << endl << " -------------------------------------" << endl;
56. cout << setw(28) <<"Police Officer Information" << endl;
57. cout << " -------------------------------------" << endl;
58. cout << setw(14) << "First Name: ";
59. cin >> firstName;
60. cout << endl << setw(13) << "Last Name: ";
61. cin >> lastName;
62. cout << endl << setw(16) << "Badge Number: ";
63. cin >> badgeNum;
64. /\*Setup for calculating if a Parking Ticket needs to be issued or not\*/
65. PoliceOfficer Officer(lastName, firstName, badgeNum);
66. if (Officer.isTicketNeccessary(Car, Meter))
67. {
68. ParkingTicket Ticket(Car, Meter, Officer);
69. cout << Ticket.print();
70. }
71. else
72. {
73. cout << endl << " -------------------------------------" << endl;
74. cout << setw(28) <<"\* No Ticket Issued \*" << endl;
75. cout << " -------------------------------------" << endl;
76. }
77. return 0;
78. }

**Note:**

Screenshots of the original code are attached at the end of the document (in the notes section), incase this is unclear, or something didn’t copy appropriately.

**Sample test run of the program:**

 1st Sample:

2nd Sample:

**Observations, error handling and general comments:**

The program runs correctly as the result in the 1st Sample and 2nd Sample, correspond to manual computations.

**1st Sample (Manual Computation) =**

Car Make = Honda

Car Model = Accord

Car Color = Blue

Car License Number = 23XJ56

Number of Minutes Car has been Parked = 125

Number of Minutes purchased = 75

Ticket to be issued = Yes

Amount to be charged = (125 - 75) = 50 (less than 1 hour) = $25

Police Officer First Name = Hissamuddin

Police Officer Last Name = Shaikh

Police Officer Bade Number = 000235FP

Appropriate Information displayed on Ticket = Yes

This is the same as the one computed by the computer.

**2nd Sample (Manual Computation) =**

Car Make = Toyota

Car Model = Corolla

Car Color = Grey

Car License Number = 42GK79

Number of Minutes Car has been Parked = 95

Number of Minutes purchased = 30

Ticket to be issued = Yes

Amount to be charged = (95-30) = 65 (more than 1 Hour) => $25 + $10 = $35

Police Officer First Name = Hissamuddin

Police Officer Last Name = Shaikh

Police Officer Bade Number = 000235FP

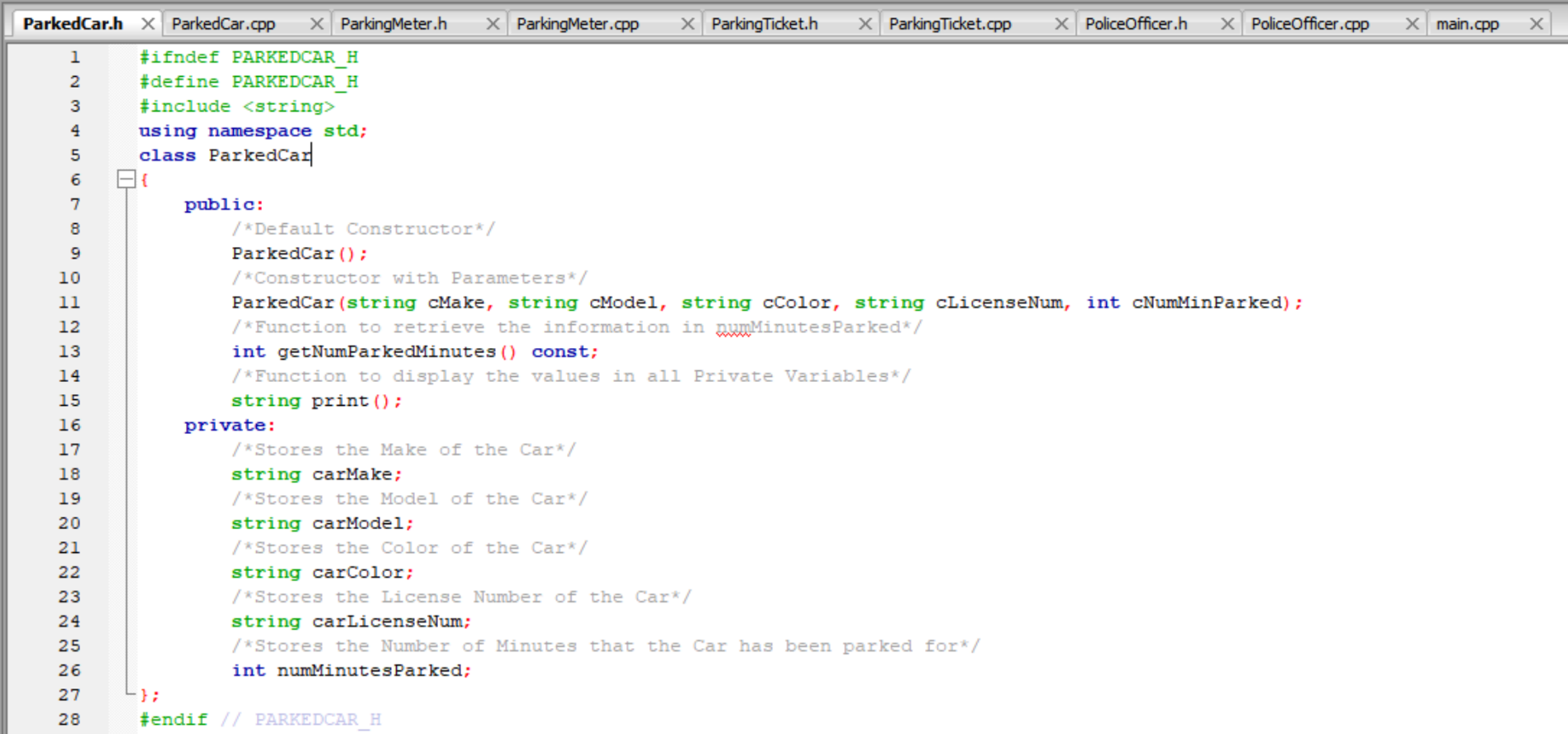
Appropriate Information displayed on Ticket = Yes

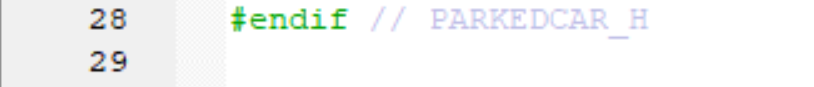
This is the same as the one computed by the computer.

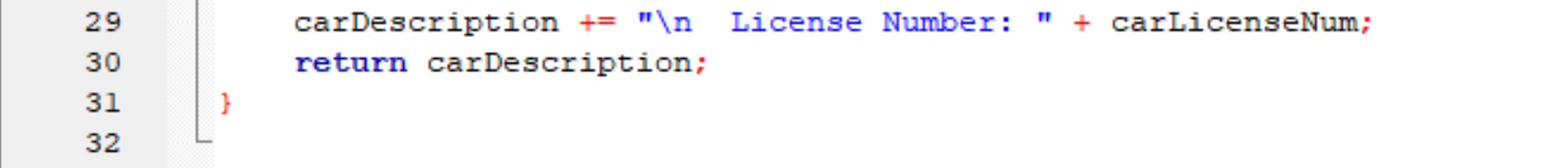
No errors should be encountered, if the user follows the prompts, and enters the appropriate data. Moreover, various measures have been taken to limit the input of error causing data, such as allowing the user to enter alphanumerical data for all inputs, except for where only numbers can be accepted, such as the Minutes. Additionally, an if-statement and do-while loops have been setup to ensure that the appropriate data is conceived by the program. Therefore, a lot of error control was employed in this program, and the presentation of the Program was made as neat and comprehensible as possible.

**Conclusions**

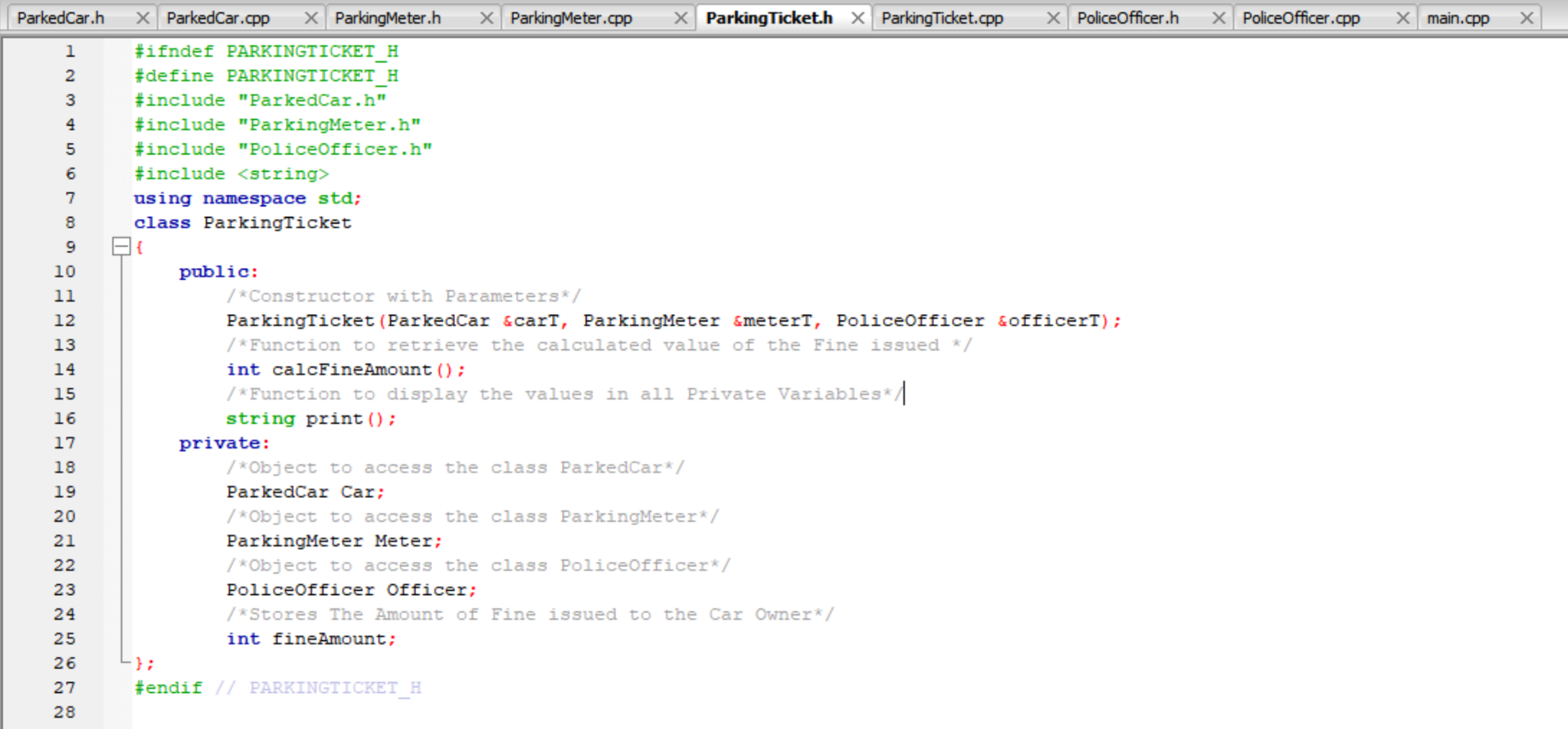
The Parking Ticket Simulator makes great use of the four classes, ParkedCar, ParkingMeter, ParkingTicket and PoliceOfficer, as it constructs a very realistic scenario of a Police Officer administrating a Parking Ticket. I do acknowledge that some of the member functions in these classes were not utilized, or not shown in the sample runs, but they do function correctly. It’s just that the implementation/showcasing of all member functions in a single program, is far too difficult and unlikely. Therefore, the Parking Ticket Simulator is a well-rounded program, capable of performing its task effectively.

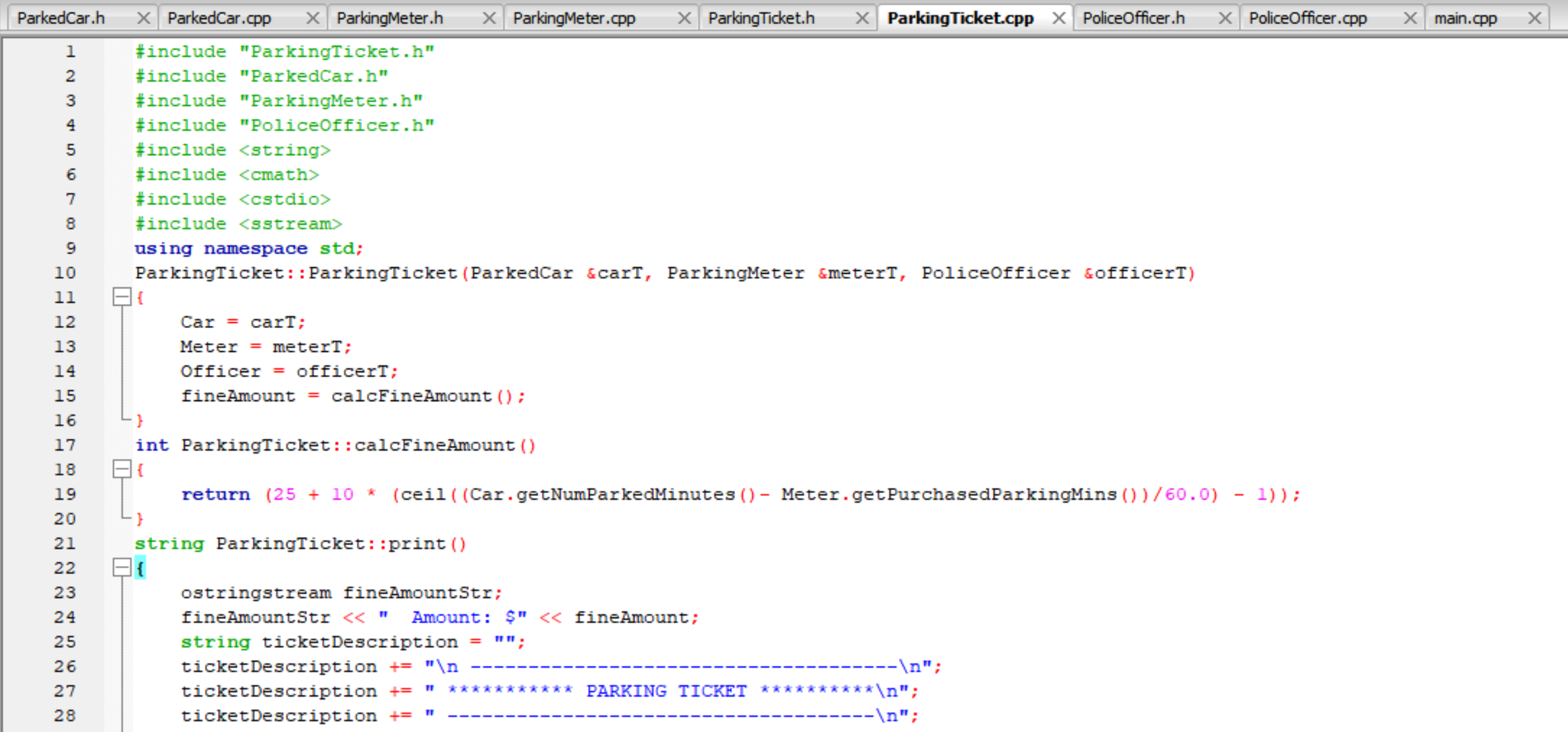
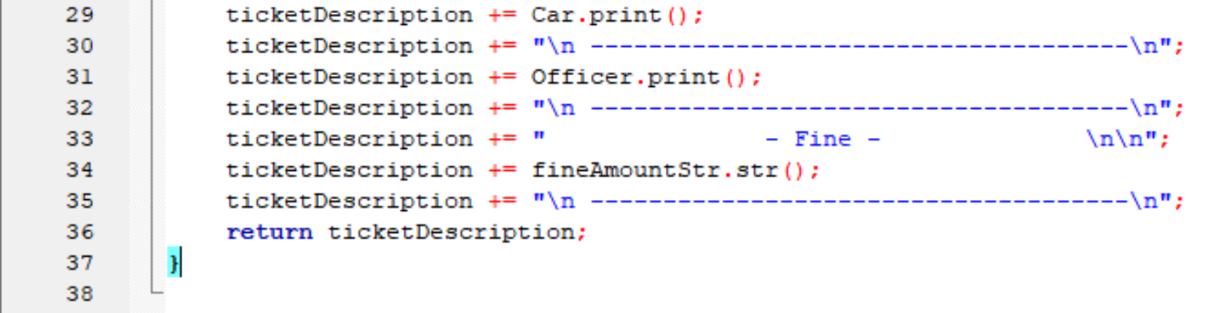
**Notes:**

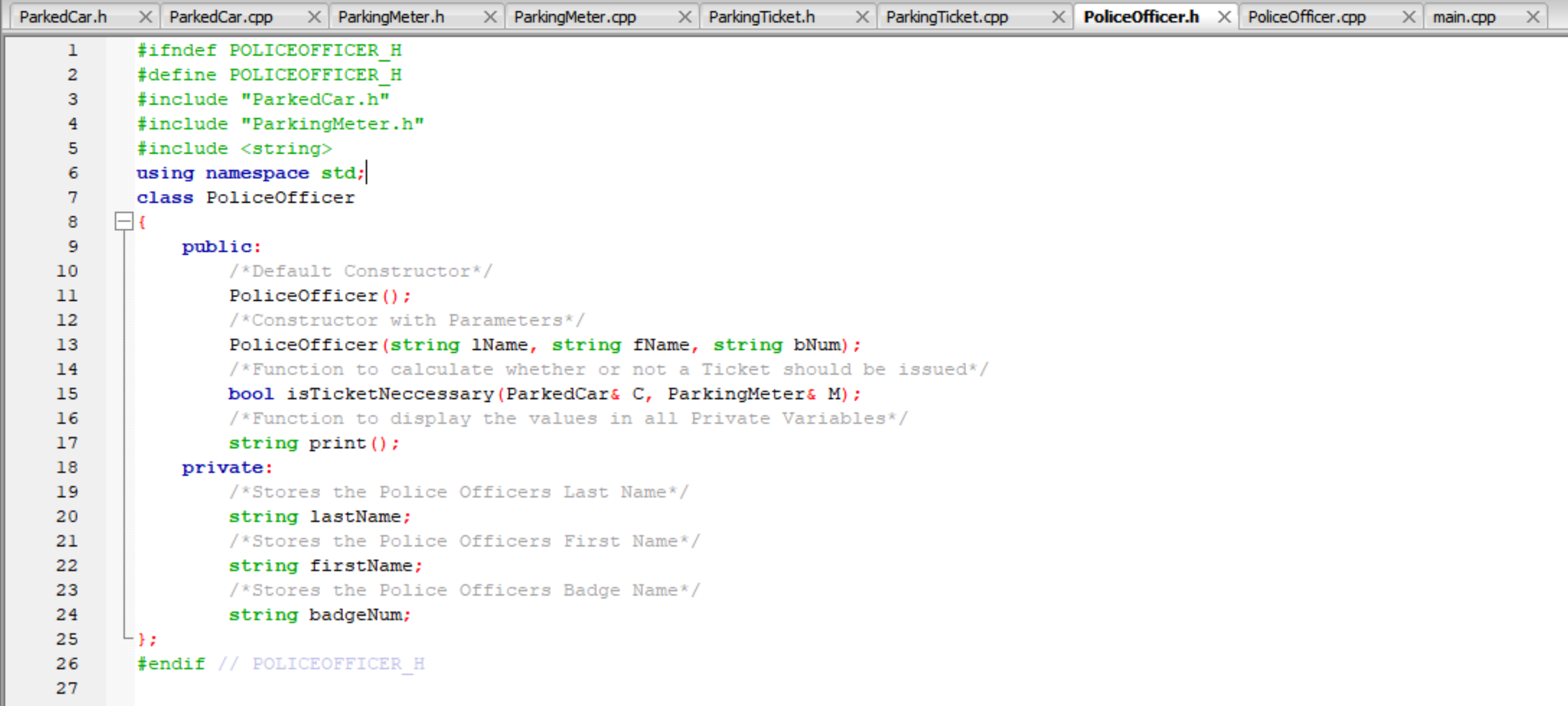












A screenshot of a cell phone

Description generated with high confidence



