# CIST2362 Software Design Document

Parking Ticket Sim Program/Class object

Kimberly Jackson

# Introduction

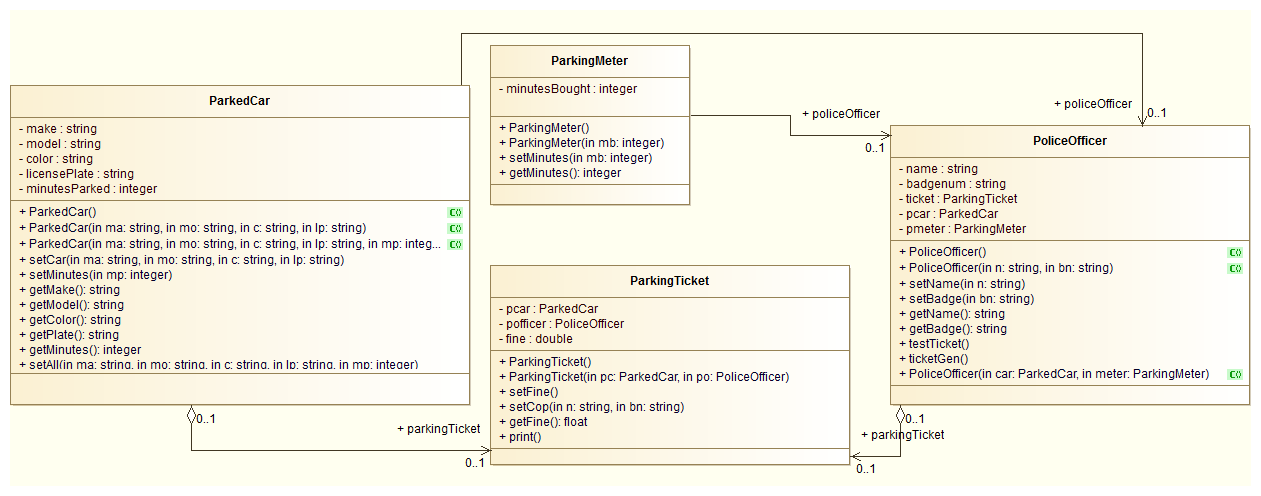
A program has been requested to assist in the automation of the parking meters. There are 4 classes being requested - Parked cars, Parking meters, Parking tickets for expired meters, Police Officer for the issuer of the ticket. 10 meters will have 10 cars at them. Randomize the number of minutes paid for each car. Then cause time to advance and either remove cars, decrease the number of minutes on the meter, or issue a ticket as necessary. The simulation will run at 30minute intervals.

# Use Cases

*Use Case: Automated - No user input*

* 1. *Program starts. 10 cars and meters are generated. One police officer is generated.*
  2. *Every car and meter pair is set to a random number of minutes for length of time parked and length of time paid for.*
  3. *Each car and their meter is displayed at start.*
  4. *Thirty minutes is added to the time.*
  5. *The cars are checked by the PoliceOfficer to determine if any need tickets.*
  6. *Tickets are issued for the cars that have expired ParkingMeters.*
  7. *Cycle continues until all cars are checked.*

# Design Overview



# System Tasks Description

### Function main

*constant int SIZE = 10*

*ParkedCar cars[SIZE]*

*ParkingMeters meters[SIZE]*

### function greeting

*[display greeting]*

*for counter = 0 to SIZE-1*

*if counter/2*

*car[counter].setMinutes = 90*

*else if counter/3*

*car[counter].setMinutes = 180*

*else if counter/5*

*car[counter].setMinutes = 300*

*else*

*car[counter].setMinutes = 30*

*endif*

*end for*

*displayallcars(car[counter])*

### Function displayallcars

*for counter = 0 to SIZE-1*

*[Display car[counter]]*

*end for*

### function step30

*for timecounter = 30 to 300, step 30*

*for counter=0 to SIZE-1*

*[police officer checks]*

*end for*

*end for*

*[Exit greeting]*

*end main*

### ParkedCar Constructors

*default = blank strings*

*alternate = accept 4 strings for the make, model, color, and license plate*

*alternate = accepts 4 strings and one int to change the number of minutes parked*

### ParkedCar Mutators

*setCar = accepts 4 strings*

*setMinutes = accepts 1 int*

*setAll = accepts 4 strings and 1 int*

### ParkedCar Accessors

*getMake = returns the make string*

*getModel = returns the model string*

*getColor = returns the color string*

*getPlate = returns the license plate*

*getMinutes = returns the minutes the car has been parked*

### ParkingMeter Constructors

*default = set 0*

*alternate = accepts 1 int to initialize the minutes bought*

### Mutators

*setMinutes = accepts 1 int to set the number of minutes paid for*

### Accessors

*getMinutes = returns int of minutes*

### PoliceOfficer Constructors

*default = set everything to blank*

*alternate = accepts 2 strings for name of officer and badge number*

*alternate = accepts Parkedcar object and ParkingMeter object*

*alternate = accepts 2 strings for name and badge number, ParkedCar, and ParkingMeter objects*

### Mutators

*setName = accept 1 string for the name*

*setBadge = accept string for badge number*

*testTicket?*

*ticketGen*

### Accessors

*getName = returns string containing name*

*getBadge = returns string containing the badge number*

### Parkingticket Constructors

*default = ParkedCar and PoliceOfficer - contains a link to a function that will calculate the fine*

### Mutators

*setFine = set of if statements that return a value based on the difference between the time on the meter and the time the car has been sitting - would have to accept an integer value from parkingmeter*

*setCop = accepts 2 strings to set the cop’s name and badge number*

### Accessors

*getFine = returns a float containing the fine*

*print = a cout of all the information passed in, including the value of the fine*