Schulich School of Engineering
University of Calgary

ENGG 233 –Fall 2017

Department of Electrical and Computer Engineering

Term Project: Parking Lot Simulator

Acknowledgment: Thanks to Bailey Duncan one of our third-year Software Engineering Students who helped us on developing this project as part his summer project collaborating with our Software Engineering program.

Project Objectives

1. To gain experience with the design and development of a relatively larger software project
2. To gain practical experience in using programming constructs learned in ENGG 233 such as classes, arrays, functions, etc.
3. To gain experience with iterative development process
4. To develop teamwork and presentation skills as a programmer

Submissions and Group Work

- Students have the option of working on the project in groups of two or working alone. Groups of larger than 2 people will not be allowed.
- If students choose to work in groups, **they must choose a partner from within their TA lab groups.** For example, if a student is in B03-Group Blue, they must choose a partner from that group.

Instructions for Groups

Students who work in groups must follow the following guidelines and regulations:
1. The workload **must be divided equally** between the two students. During the demos, the TAs will specifically ask about what each student has done in order to ensure both students have contributed.
2. Both students must be present at each demo. If a student is absent, that student will not be awarded the grades for the demo.
3. Only one of you needs to submit each milestone to D2L. However, please make sure both of your names are included in your submission.

Important Note:
You can work with other groups and students, however, be very careful not to simply use the code from other students. Plagiarism is a serious academic offense. Please consult the University of Calgary Calendar for more information:
http://www.ucalgary.ca/pubs/calendar/current/k-5-1.html

Introduction:

You have been hired by the municipal government of the city of XYZ to design and develop a Parking Lot Simulator. This city intends to use this simulator to determine if a given piece of land is appropriate to be converted to city parking.

This document contains instructions to design and develop a Processing program that implements a Parking Lot Simulator. The target simulator is supposed to demonstrate cars randomly entering the parking lot and staying there for a period of time, and then leaving the lot randomly. The exact time that a car enters the lot, and the time that it leaves the lot are recorded. Based on these two times the simulator should display the payable fee for each car at the exit gate.

Summary of Basic Requirements:

- Draw a colorful parking lot with the capacity of 60 cars consisting of 6 rows with the capacity of 10 cars per row. There should be one entrance and one exit gate for the parking lot and they must be depicted in the drawing.
- Lot should have a main street (vertical, north-south) in the middle and several avenues (horizontal, east-west). Streets divide the parking lot into several lot sections.
- Each parking stall is green when available and red when occupied.
- The program must show cars advancing the entrance gate and turning into the lot randomly if the lot's entrance sign shows stalls available. Otherwise, cars pass the gate and not enter.
- If the number of cars reaches the maximum (60) the entrance sign shows "Full". Otherwise, it should display the number of available stalls.
- Each car is charged according to certain hourly rates explained later.

Program Modules:

Your program must have several modules that will be organized as separate tabs in Processing IDE.

Here is the list of some of the modules or classes that your program should have (Note: you may decide to have more classes. You are also free to make changes to the classes as you see fit, as long as the functionality of the program is not compromised):

- **ParkingLotSimulator:**
  This is a global file that is also your program starting point. This module should contain several global objects as follows:
  - An object of class ParkingLot
  - Two objects of class Streets (north street and south street)
  - An object of a class called PriceCalculator that is responsible to calculate the cost of parking in a stall (cost of parking = stay time * cost per hour)
  - More variables, as needed

  This module must also have several functions as follows:
  - A setup function that creates the static parts of the program such as setting the size of the street, instantiating the global objects, etc.
  - A draw function that draws the dynamic part of the program. For example, cars moving into the screen and moving out of the screen.
  - MouseClick or MousePressed functions as needed

- **Class ParkingLot:**
  - Member variables (attributes):
    - lots dimensions
    - an array of objects of ParkingStallSection
    - two objects of type Gate; one for the north gate and one for the south gate.
  - Member functions (behaviors):
    - Needs to have member function drawLot that draws a Parking lot on the screen (Figure 2).

- **Class ParkingStallSection**:
  - This class keeps track of a section of parking stalls. Figure 2 shows that the parking lot, has 6 parking stall sections, with each having 10 stalls.
  - Member variables (attributes): as needed
    - Has an array of parking stalls
  - Member functions (behaviors): as needed

- **Class ParkingStall:**
  The definition of this class was part of exercise E in lab 7 and you can use the given code. Or, you can modify it and use it as you need.

- **Class Date:**
  The definition of this class was part of exercise E in lab 7 and you can use the given code. Or, you can modify it and use it as you need.

Class Gate:

- Member variables (attributes):
    - status of the gate (closed/open),
    - position on the screen (x and y coordinate).
    - This class also can have a member variable called fee, which should be displayed at the exit gate.
- Member functions (behaviors):
    - Member functions openGate, and closeGate. The entrance gate should be closed when parking is full.

- **Class Street:**
    - Member variables (attributes):
        - Street name or number (north street, south street)
    - Member functions (behaviors):
        - A function called drawStreet that displays a street and displays its name on the screen.

- **Class Car:**
    - Member variables (attributes):
        - speed, position, or any other attribute as needed
    - Member functions (behaviors):
        - A function called move that moves a car forward on a street, and also makes right or left turns.
        - Function drawCar that draws a car on the screen.

- **Class Control Panel**
    - That shows time, date, the parking rates, etc.
      Note: You don't have to implement the parking rates in the detail that is explained in figure 1. The following rates are adequate:
        - Monday to Saturday $3 per hour.
        - Sundays $1.5 per hour.

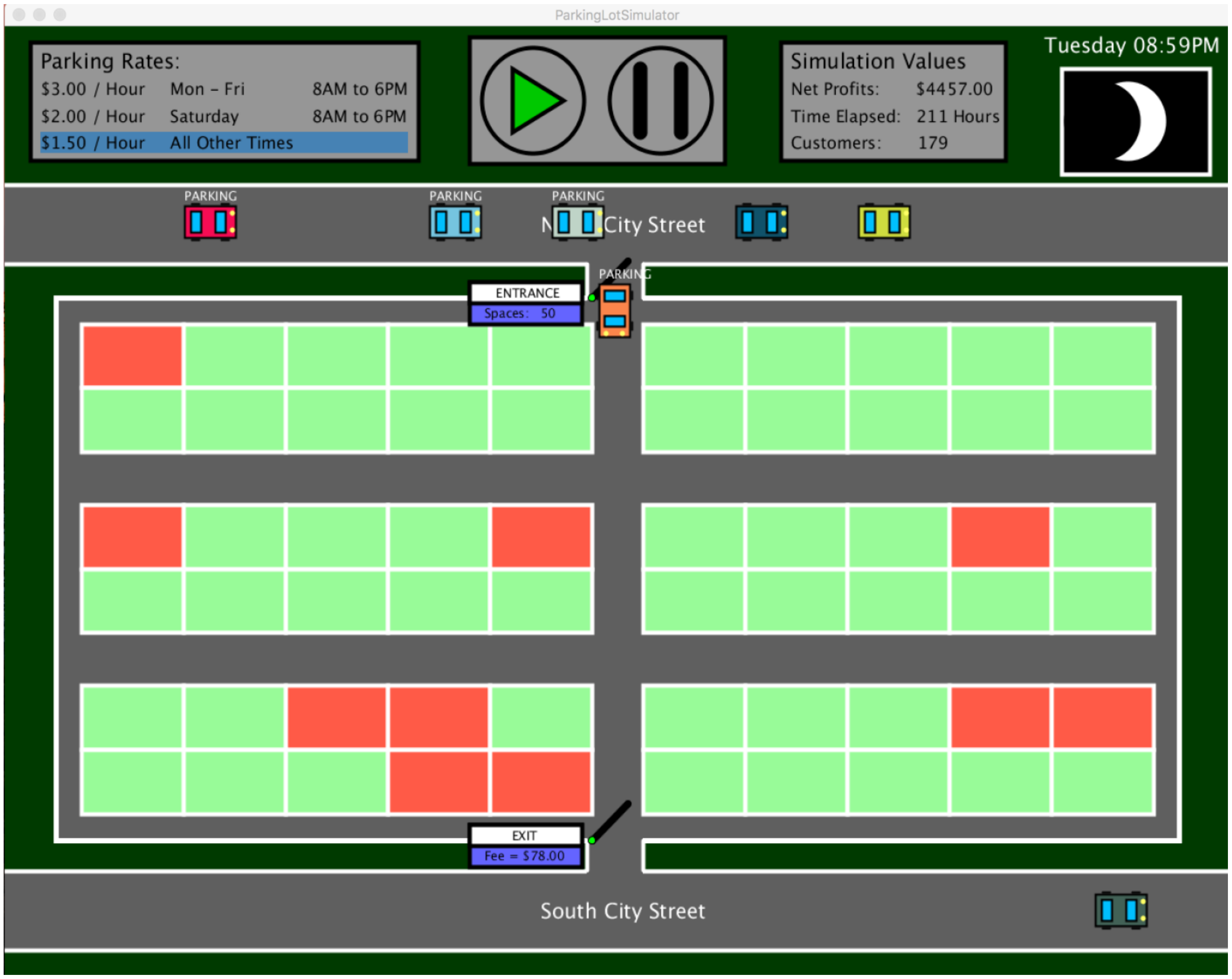
      Also, the time and date can be simply shown by text. You don't have to have a picture of the moon or sun for the day and night.
    - Behaviors: drawPanel that draws component shown in figure 1.

Important Notes:

1. You can add more classes if needed.

2. In addition to the member functions listed, all classes should have constructors as needed.

Figure 1 below illustrates a completed version of this program. As it can be seen, the user interface shows the parking lot, its parking sections, parking stalls, gates, etc.


ParkingLotSimulator

Parking Rates:

$3.00 / Hour	Mon – Fri	8AM to 6PM
$2.00 / Hour	Saturday	8AM to 6PM
$1.50 / Hour	All Other Times	

Simulation Values

Net Profits:	$4457.00
Time Elapsed:	211 Hours
Customers:	179

Tuesday 08:59PM

Figure 1 – Completed user interface of the program

This project is designed to be implemented in three phases (milestones) as outlined in the rest of this document.

Project Requirements in Iteration

Software projects can be very complex, therefore the process used to manage this complexity is very important. One effective process widely used in software development is referred to as iterative development.

Simply put, iterative development is the processes of breaking down a larger project into smaller steps (i.e. iterations or milestones). To ensure you are on track for the project, and to make sure you will get marks for all your efforts, not just for the completed project, this project is divided into 3 different milestones. For each milestone, you should fully test your code and ensure you satisfy all the requirements for that milestone correctly. This is very important as you don't want to keep building on a faulty project.

Important Advice: You should save a version of your project after each milestone. This is the basic idea of "*version control*" in real-life software development. This is to keep track of the changes made to your program and to ensure that you have a working version of your software which you can go back to if needed.

Milestone 1 (70 points):

This milestone is due on the week of Nov 20th. Students must demo their milestone in their scheduled lab during this week, and submit their code on D2L before their scheduled labs:

- **Lab Section B01:** Wednesday, Nov 22, before 8:00 AM.
- **Lab Section B02:** Wednesday, Nov 22, before 3:00 PM.
- **Lab Section B03:** Tuesday, November 21, before 8:00 AM.
- **Lab Section B04:** Thursday, November 23, before 12:00 PM.

In this milestone you are expected to implement the following classes/modules:

- Classes ParkingStall and Date that are already available to you.
- Class ParkingStallSection
- Class ParkingLot
- Enough of ParkingLotSimulator file so that your program is able to run

A program output, as demonstrated in the following figure, is the minimum expectation; but you can demonstrate additional feature if you like:

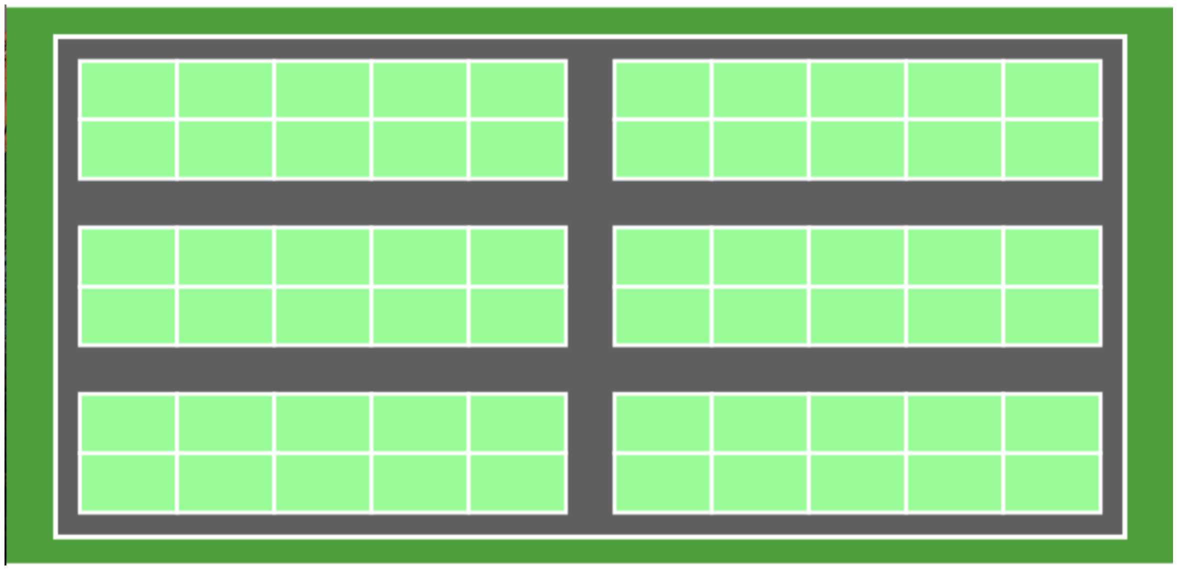


Figure 2 – Program output for milestone 1

What to Submit for Milestone 1:

Submit on the D2L (Dropbox) the copy of your entire program as a pdf file and a zip file containing all your `.pde` files: `ParkingStallSection.pde`, `ParkingLot.pde`, `ParkingStall.pde`, `ParkingSimulator.pde`, and any other `.pde` files that you have created in addition to the above files.

Milestone 2 (70 points):

This milestone is due on the week of Nov 27th. Students must demo their milestone in their scheduled lab during this week, and submit their code on D2L before their scheduled labs:

- **Lab Section B01:** Wednesday, Nov 29, before 8 AM.
- **Lab Section B02:** Wednesday, Nov 29, before 3 PM.
- **Lab Section B03:** Tuesday, November 28, before 8 AM.
- **Lab Section B04:** Thursday, November 30, before 12 PM.

In this milestone you are expected to implement the following classes:

- Class Street
- Class Gate
- Class PriceCalculator
- More code to be added to ParkingLotSimulator file as needed

A program output, as demonstrated in the following figure, is the minimum expectation; but you can demonstrate additional feature if you like:

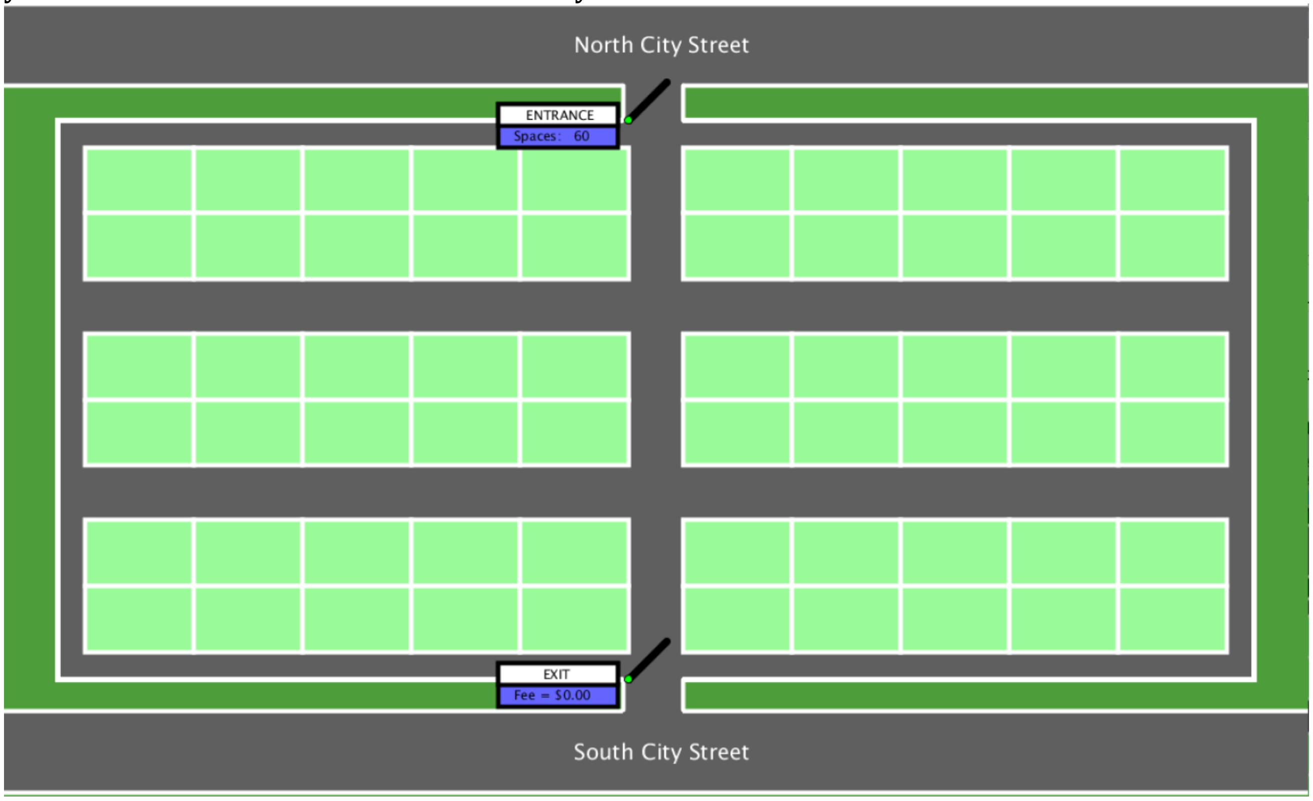


Figure 3 - Program output for milestone 1

What to Submit for Milestone 2:

Submit on the D2L (Dropbox) the copy of your entire program as a `pdf` file and a zip file containing all your `.pde` files: `ParkingStallSection.pde`, `ParkingLot.pde`, `ParkingStall.pde`, `ParkingSimulator.pde`, `Street.pde`, `Gate.pde`, `PriceCalculator.pde`, `Date.pde`, and any other `.pde` files that you have created in addition to the above files.

Milestone 3 (60 points):

This milestone is due on the week of December 4th. Students must demo their milestone in their scheduled lab during this week, and submit their code on D2L before their scheduled labs:

- **Lab Section B01:** Wednesday, Dec 6, before 8 AM.
- **Lab Section B02:** Wednesday, Dec 6, before 3 PM.
- **Lab Section B03:** Tuesday, Dec 5, before 8 AM.
- **Lab Section B04:** Thursday, Dec 7, before 12 PM.

In this milestone you are expected to implement the following classes:

- Car
- Class ControlPanell
- Other possible classes as needed
- Complete ParkingLotSimulator file so that your program is fully functional

In this version, your program you should demonstrate cars moving in the street, randomly entering the parking lot. When a car enters one of the parking stalls randomly, the stall should turn from green to red. Also, cars should leave the parking lot randomly. When this happens, one of the red stalls must change from red to green. At the exit, the parking fee should be displayed (see figure 1)

What to Submit for Milestone 3:

Submit on the D2L (Dropbox) the copy of your entire program as a `pdf` file and a zip file containing all your `.pde` files: `ParkingStallSection.pde`, `ParkingLot.pde`, `ParkingStall.pde`, `ParkingSimulator.pde`, `Street.pde`, `Gate.pde`, `PriceCalculator.pde`, `Car.pde`, `ControlPanel.pde`, `Date.pde`, and any other `.pde` files that you have created in addition to the above files.