

**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Group Work****)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Dumac**  **Naufal**  **Muhammad** | | **Student ID Number**  **2101699653**    **2101714843**  **2101718476** | |
| 1.    2.    3. | | **Revano Chen**  **Muhammad Zavier**  **Andi Yusuf** |  | |
|  |  |
| **Course Code** | **: COMP6510** |  |  | | **Course Name** | | **: Programming Language** | |
| **Class** | **: L2BC - LEC** |  |  | | **Name of Lecturer(s)** | | **:** 1. Minaldi Loeis | |
|  |  |  |  | |  | |  | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment**  (if any) | : Parking Simulation | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 5-05-2018** |  |  | | **Submission Date** | | **: 5-05-2018** | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(Name of Student)

Dumac Revano Chen, Muhammad Andi Yusuf, Naufal Muhammad Zavier

**Parking Slot Simulation**

This program is a simulation that simulates the daily actions that occurs in a parking lot. This means that the simulation simulates and record vehicles going in, staying and leaving for a set amount of time. It also tracks whether or not there is capacity left for vehicles to enter or not, if not the vehicles do not enter as there is no space left. The simulation will run for 48 seconds, with each 2a second representing 1 hour as to represent a whole full day.

There will be various types of vehicle entering and each different type of vehicles have their own pay rate. At the end of the simulation the report is printed stating how many vehicles were parked and gives the total revenue of that day.

**Explanation Of The Code**

**Phasecontrol.java**

This is the first and only class to be executed in the main Java file of ParkingFinalProject.java. Phasecontrol runs the main loop of the simulation, which is based on a timer that lasts for 48 seconds.

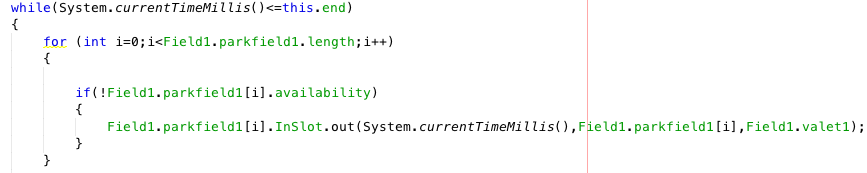


Figure 1: Phasecontrol: while loop

During this loop the vehicles, car and bus are generated into the queue of varr. The amount of cars and bus generated is randomized.

Users also have the ability what day they want to simulate, whether it is work day or a holiday. It is set by using random range where work day has a higher range as more cars typically use more parking due to people working.

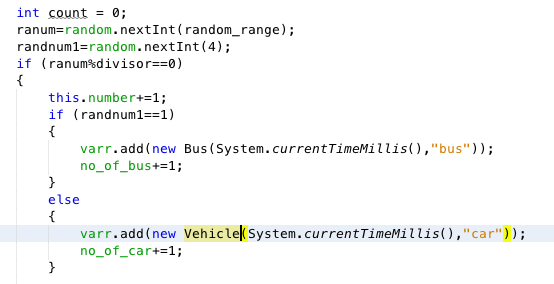


Figure 2: Phasecontrol: random generation

**Vehicle.java**

This class contains the constructor for the class Vehicle, which contains the blueprint for the object ‘car’. The class vehicle has CurrentTimeMillis as its parameter so that it can resume the timer from the while loop found in Phasecontrol.java. Another function called int get\_stay returns how long a car has stayed in the parking lot. Once the time limit is over, the data of the car that is inside the queue data container is emptied, and the total amount of fare that car generates is collected. Queue is chosen as it has a FIFO structure, hence it’s easy to insert and remove data in order. This is shown in Figure 3 below.

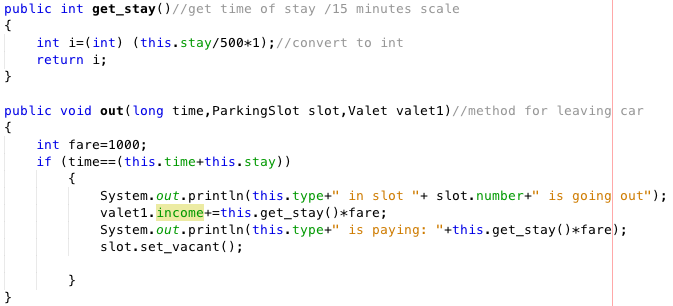
****

Figure 3: Vehicle get\_in and out

The following function, void set\_stay() sets how long a vehicle stays in the parking lot before it’s data is set to empty. Figure 4 shows the range in which a vehicle can stay. The range is determined by a random integer, hence different vehicles stay for different intervals of time.



Figure 4: Set stay

**Bus.java**

This class contains the vehicles of the type bus. This inherits from the parent class Vehicle which was explained previously. Unlike the car in class vehicle, the bus object has its own version function void out. The void out in class vehicle is overridden with the function below. The main difference is the fare on the bus is higher than the car.

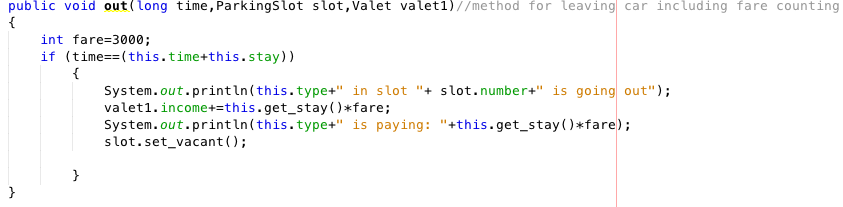


Figure 5: void out for class Bus

**ParkingField.java**

ParkingField initializes the Parkingslot, ParkingSlotb and the Valet. This class constructs those classes alongside the array Parkfield which will contain the vehicles that is popped from the queue. An array is chosen as it has definitive capacity.

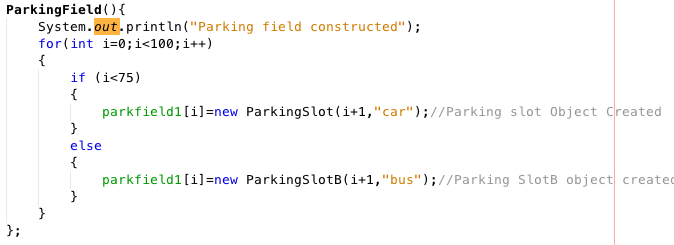
****

Figure 6: Constructor for ParkingField

The way it is setup is that if for the first 75 iteration of the for loop, it will create a ParkingSlot for Vehicle car, afterwards it creates ParkingSlotb for class Bus.

**ParkingSlot.java**

This class can manipulate the ParkingSlot for the car and mainly consists of setters and getters. The important function in this class is void Car\_In() which outputs where is a car is currently parked. As shown in Figure 7 below, it sets the vacancy available or not and can also get the current availability for checking.

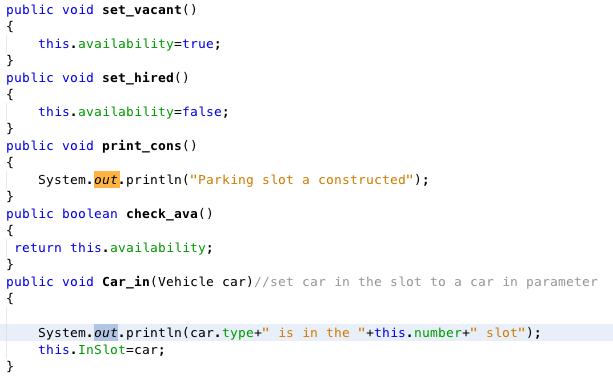


Figure 7: ParkingSlot

**ParkingSlotB.java**

ParkingSlotB inherits from its parent class ParkingSlot, and it’s only solely for vehicle bus. All the implementation is identical to its parent class.

**Valet.java and Operator.java**

This class uses an interface from operator, and its main purpose is to check whether there is any free space left for vehicles to enter the parking slot and to register the vehicles inside the parking slot.

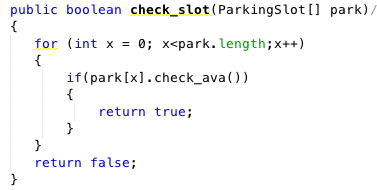


Figure 8: Checks for free space

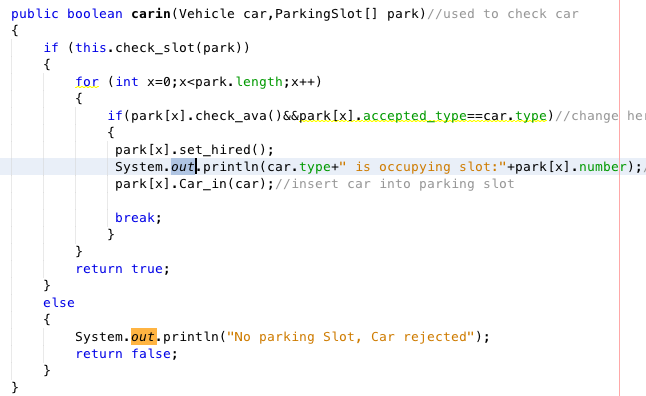


Figure 9: boolean carin

Figure 9 above shows how the car is registered through the valet. If there is enough space and the vehicle type is car, it will set the space as occupied through the function set\_hired (from class ParkingSlot) and it will insert the car into the parking slot for a set amount of time.

**Results Of the Simulation**

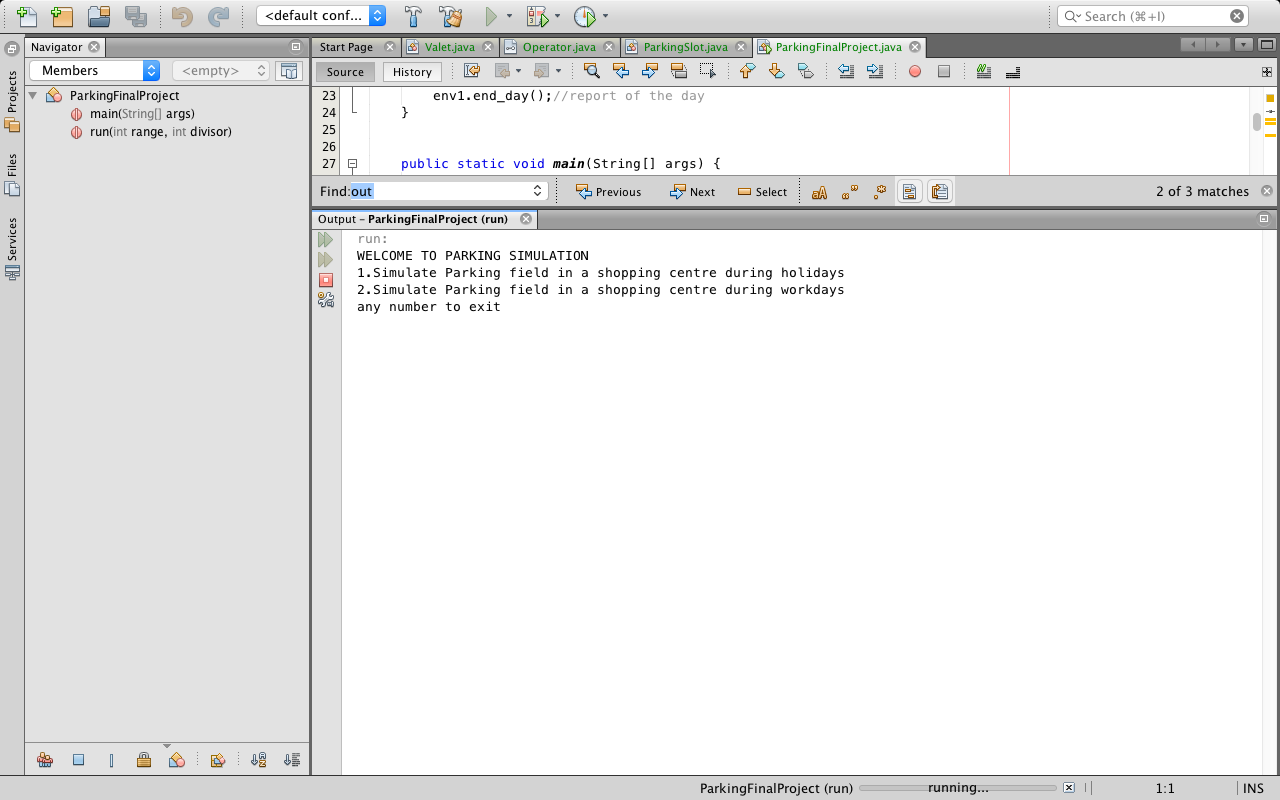


Figure 10: Menuscreen

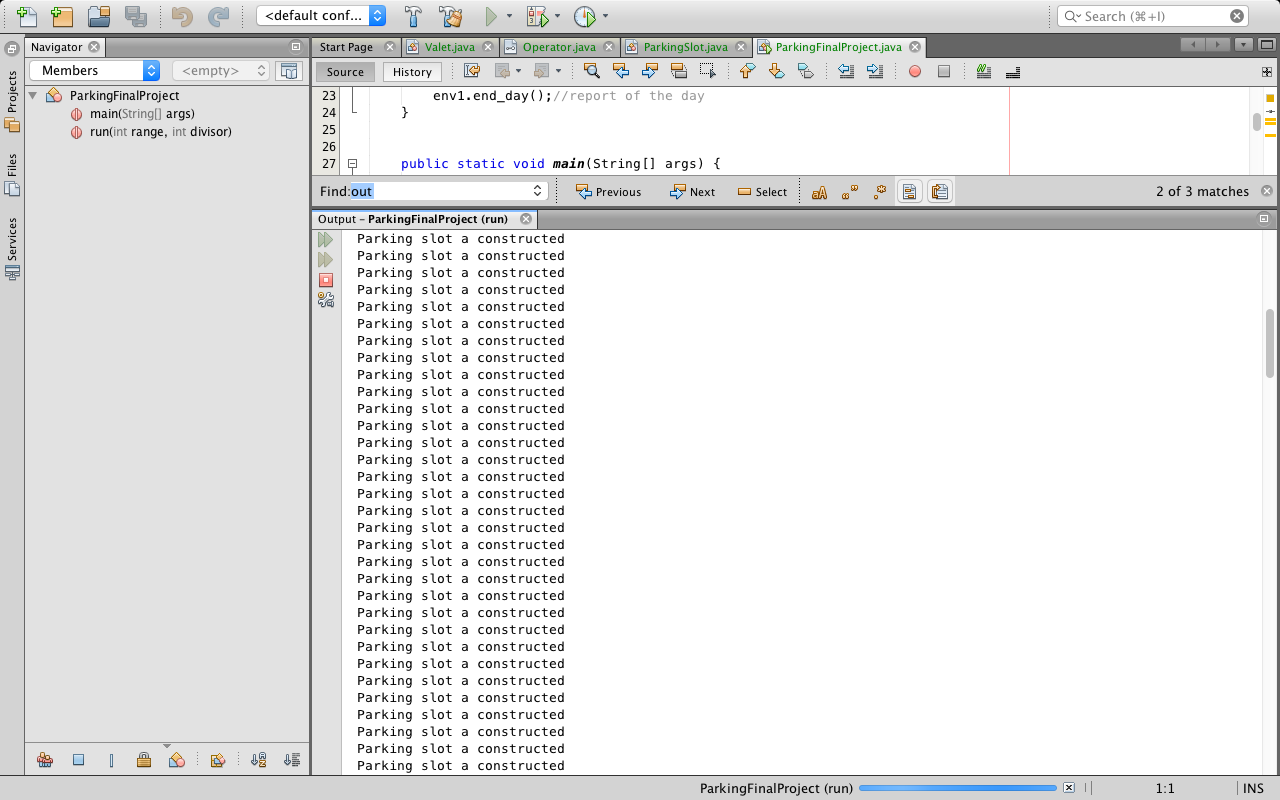


Figure 11: Construction Of Parking Slot

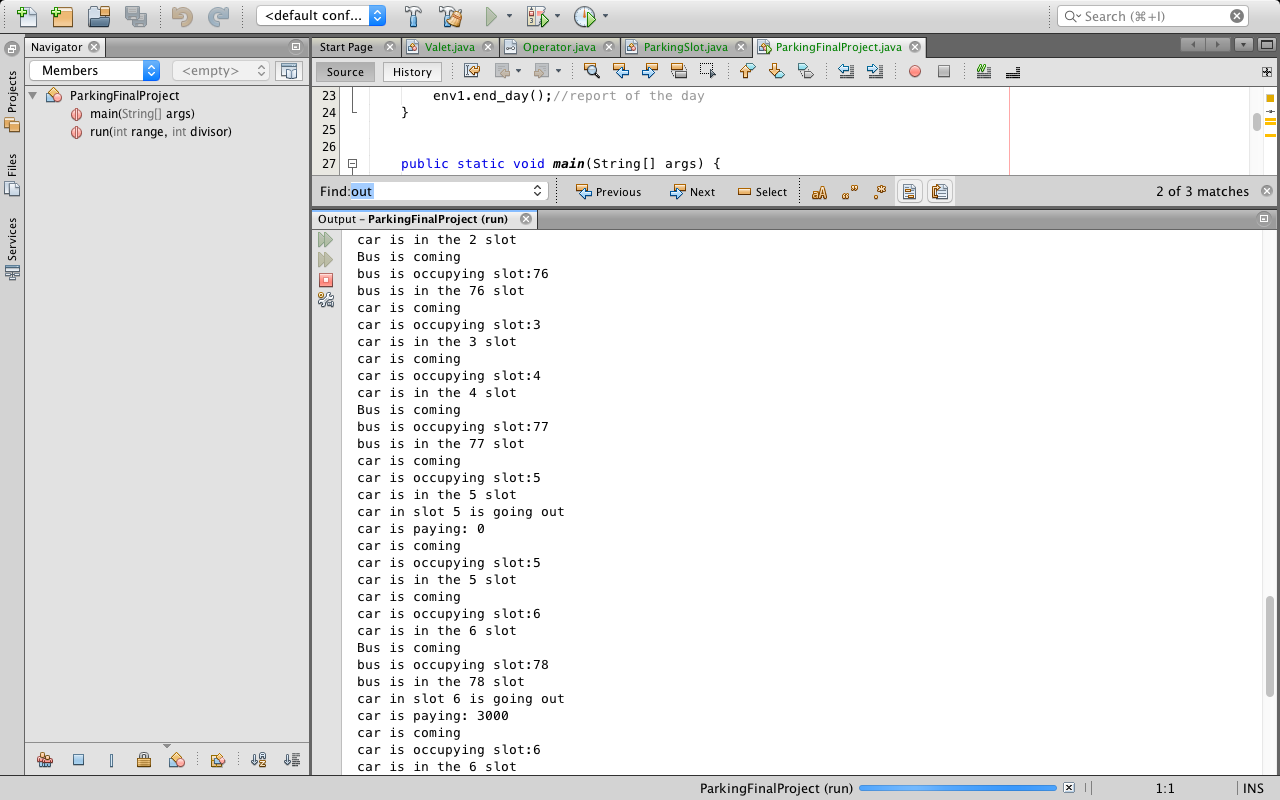


Figure 12: Vehicles Entering, occupying and leaving

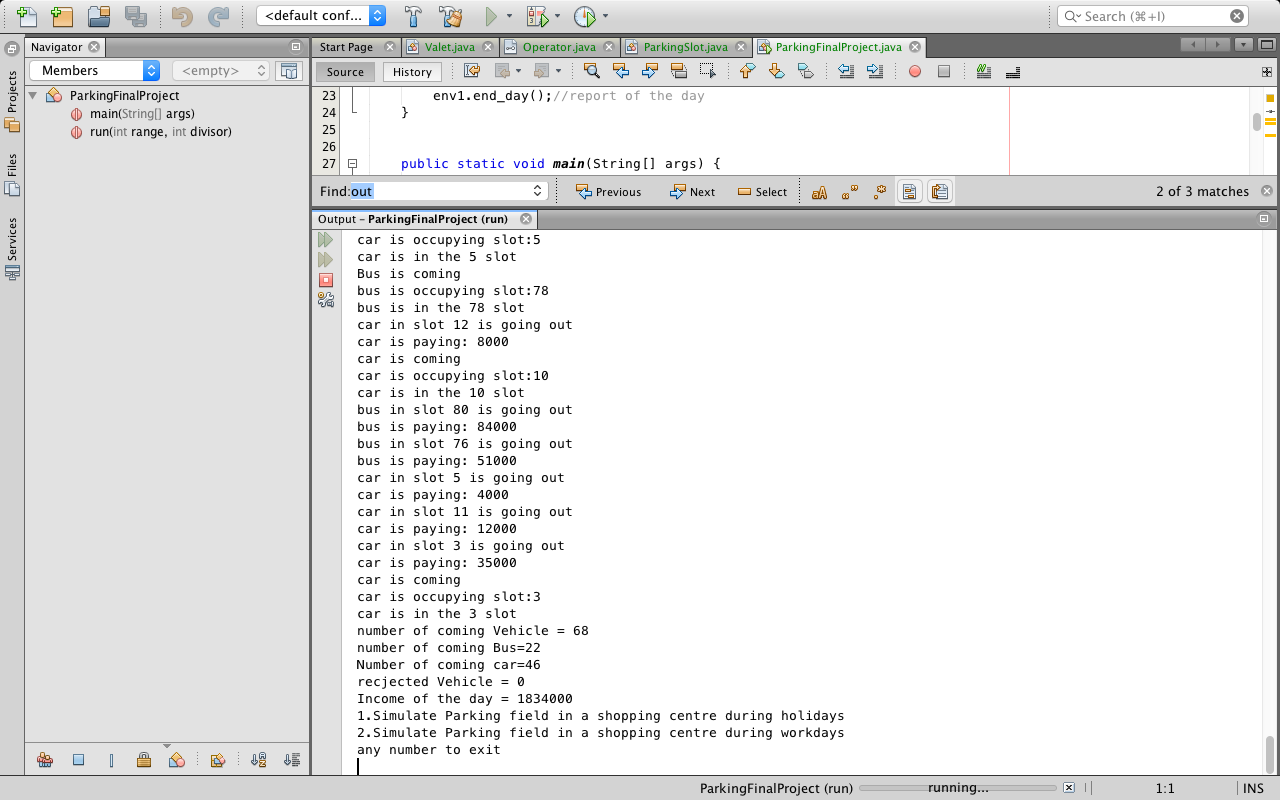


Figure 13: End of simulation

Project Demo: <https://drive.google.com/open?id=1EeFhd3LOOHISh4W9ViHQkpHuaPMSDmab>

GitHub: <https://github.com/nafuza1299/Programming-Language-Final-Project>