COS70006: Object Oriented Programming
Project 1 - Semester 2, 2021

20% of your final mark

Due by at the end of Week 6 (11:59pm Sunday 12 Sept 2021)

NOTE: Extensions must be applied for before the due date by emailing the unit convener. A request for extension after the due time or without a valid medical certificate will **NOT** be responded to.

Project 1 – Parking Spot System - Requirements

A small system is required that will help manage cars at a parking site for a company. You are to develop a system that has exactly following four classes:

- Application class
- CarPark class
- ParkingSlot class
- Car class

Application is the Console (Text Based) Interface class including the main() method and handling all inputs and outputs.

CarPark is responsible for maintaining a list of available parking slots. You should be able to find a slot, add a slot, delete a slot, and provide a list of all slots included in the car park.

There are two types of parking slots: slots only for visitors and slots only for staff members. A parking slot must have an identifier, which starts with a capital letter, followed by a two-digit number e.g. “D01”, “E27”. A parking slot also should know if a car and what car is parked in the slot. You must be able to add a car to the slot and remove a car from the slot.

A car will be identified by its registration number. A registration number always starts with a capital letter, followed by a four-digit number e.g. “T2345”. A car should have an owner and knows if the owner is a staff member.

For this assessment, you do NOT need to maintain a list of parked cars in any of your classes.

The office manager requires a simple Console (Text Based) Interface that will have the following menu items.

1. Add a parking slot, all information provided by users
2. Delete a parking slot by slot ID (only if not occupied)
3. List all slots in a well-defined format with information including slot ID, slot type, whether occupied, and if occupied, show the car registration and the owner.
4. Park a car into a slot (provide slot ID and car information)
5. Find a car by registration number and show the slot and the owner if the car is in
6. Remove a car by registration number
7. Exit

Required conditions to be checked for user inputs:

1. User inputs for menu options, car information, and parking slot information should not crash the program
2. Parking slot number must be an uppercase letter followed by 2 digits
3. Car registration number must be an uppercase letter followed by 4 digits
4. Each slot should have a unique slot number
5. A parking slot cannot be deleted if there is a car being parked there
6. Visitor car can only be parked in a visitor slot and staff car can only be parked in a staff slot
7. A car can only be parked in an unoccupied slot
8. A car can only be parked in one slot

Code:

- The solution must be a BlueJ Project.

Some Expectations.

1. All classes and methods include Javadoc
2. Code is well structured and object oriented.
3. User interface class (Application) is broken down into single purposed methods
4. User interface class (Application) is separated from business logic classes
5. The user input is safe and will not crash the program
6. The user should be well informed about what he/she is expected to enter and the feedback of their action.
7. Pre-condition checking is included in the class methods.

Plagiarism

THIS IS AN INDIVDUAL PROJECT.

- The submitted work must be your own work.
- You must keep your own work from other students.
- You may NOT view the code of other students.
- You may discuss the work with teaching staff.
- You may discuss the big picture with peers but the final design should be yours.
- You must name and code attributes and operations on your own.
- There will be absolutely no tolerance of plagiarism.
- Any person that presents any work that is not their own or is not properly referenced will be awarded 0 marks for the project.

Submission

Zip you BlueJ project into a file and submit it to ESP by the due time

Marking Scheme

Total marks: 80

Items	Max Marks
Code readability: • Javadoc for all class headers and methods headers • Proper comments for variables and blocks • Proper indentations and use of blank lines • Proper Java naming conventions and meaningful names	12
Appropriate implementation of functionality using well-structured OO classes • User Interface class separated from business logic classes • Broken down into single purposed methods • Popper user messages for user inputs and outputs • The user input is safe and will not crash the program • Each class is defined as required • Logic of code can be followed	18
Each function in the menu works as required and required conditions checked • Add a parking slot *(8 marks)* • Delete a parking slot (only if not occupied) *(8 marks)* • List all slots in a well-defined format *(8 marks)* • Park a car to a specified slot *(8 marks)* • Find a car and show the slot if the car is in *(8 marks)* • Remove a car *(8 marks)* • Exit *(2 marks)*	50
Total	80