# Project: Instructions

**Assessment Resources:**

|  |
| --- |
| Marking key available for lecturer via Blackboard.  Students may refer to the lecture material (GitHub and Blackboard) in formulating their answers. |

**Assessment Instructions:**

|  |
| --- |
| Students must complete every section. Answer succinctly using full sentences. At most three paragraphs are expected per answer.  All answers must be at the student’s own words – copying generated code or answers from ChatGPT or other AI tools is **strictly** prohibited.  Please ensure that all instructions are followed **carefully**, and submissions are well-organized, clearly labelled, and accompanied by any necessary explanations or justifications. |

## Objectives

By completing this task, you will demonstrate the following competencies as outlined in ICTPRG430:

* **Modularity**: Implementing the logic for one object operation using a modular approach.
* **Data Structures**: Utilizing arrays of primitive data types within a class.
* **File Operations**: Reading from and writing to a text file.
* **Class Design:** Developing two classes with four instance variables each.
* **Object Construction**: Creating a class that offers two options for object construction.
* **Object Aggregation**: Employing user-defined object aggregation within a class.
* **Polymorphism**: Implementing polymorphism to enhance code extensibility.
* **Debugging**: Utilizing a debugging tool to troubleshoot your code.
* **Code and Documentation Conventions**: Applying specified coding and documentation standards.
* **Unit Testing:** Conducting and documenting two unit test cases.

As part of this assessment, you will demonstrate competencies in using a version control system, as outlined in ICTICT449. You will plan, install, create, and manage a repository to control versions of your code for the Carpark system.

## Scenario

The City of Moondalup is progressively embracing smart city initiatives to enhance urban living, improve efficiency in city services, and promote sustainable practices. As part of this initiative, the city council is eager to transition to a smart parking solution to optimize carpark usage, reduce traffic congestion, and enhance the overall parking experience for residents and visitors.

The City of Moondalup is progressively embracing smart city initiatives to enhance urban living, improve efficiency in city services, and promote sustainable practices. As part of this initiative, the city council is eager to transition to a smart parking solution to optimize carpark usage, reduce traffic congestion, and enhance the overall parking experience for residents and visitors.

You have been contracted to create a prototype solution that uses sensors and displays to provide timely information about available parking bays as well as relevant information about weather and other community messages.

The city’s Chief Technology Officer (CTO), has outlined the following requirements:

* The system must accurately track the status of each parking bay in real-time.
* The display must be updated promptly as cars enter or exit.
* The system should be robust, easy to maintain, and scalable for future enhancements.
* The application must follow best coding practices and include unit testing.
* You must use Git and Github for version management.

## Coding requirements

To meet the specifications of the project, you must do the following:

* Create at least three classes.
* At least one class must include three or more parameters.
* At least one class must *aggregate* another class.
* You should demonstrate an example of polymorphism
* Include at least two unit cases
* Create a main.py demonstrating the core interaction between instances of your classes
* Use PEP8 throughout your code and docstrings for major functions within your code

## Version control requirements

* Create a new repository and configure it with a README, .gitignore, and other essential setup files.
* Initialize your local repository and link it to a remote repository on GitHub.
* Make initial commits with the basic structure of your Carpark system.
* As you develop the system, commit your changes each time you reach a significant milestone or complete a task.
* Make at least three commits to demonstrate the evolution of your project.
* Manage any changes or improvements by committing to the repository with clear, descriptive commit messages.

## Questions

1. Have you included the carpark-prj-guided.md document, with all evidencing sections addressed?

Yes, I included mark down document with all the evidencing sections addressed.

(if you answered no, see below. If you answered yes, you can skip the remaining questions)

**You must complete all evidencing requirements in the project guide and include with your submissions.**

**—OR—**

**Include the following answers with your submission:**

1. What is an example of polymorphism in your code? Justify your answer

The example of polymorphism in my code are “Sensor” class and its subclasses “EntrySensor and ExitSensor”. To be specific, “detect\_vehicle” involves polymorhism. Both “EntrySensor and ExitSensor” classes inherit from the abstract base class “Sensor”. They both implement the “update\_car\_park” method which is an abstract method implemented in the “Sensor class”.

1. What is an example of aggregation in your code? Justify your answer

The relationship between the “Carpark”, “Sensor” and “Display” classes are the example of aggregation. “Carpark” class has a list of “sensor” and “display” objects. “Sensor” and “display” objects exists independently from “carpark” object. That means they can exist without being tied to “carpark” instance.

Outline at least one *specific* instance where you performed debugging.

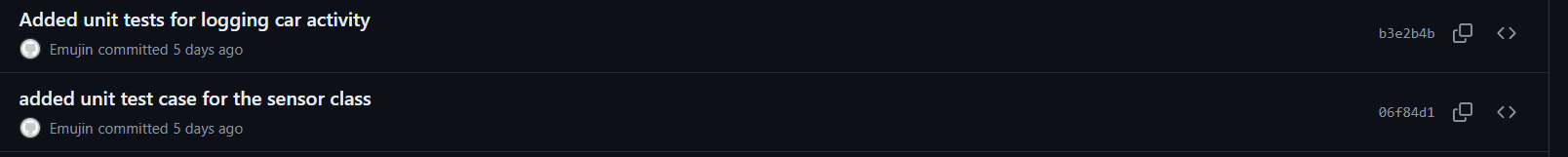
What was the issue? Issue was when unit testing, specifically when testing car logging when entering, It was raising asserion error.

How did you identify the problem? I ran the test method to see the exact nature of the assertion error to confirm the problem exists.Then I checked the log file to see it writes the car activity when running the test. After that I examine the implementation of the “add\_car and remove\_car” methods along with the “log\_car” method to ensure the methods are correctly set up.

What steps did you take to resolve the problem? To resolve this problem, I fixed the “new \_carpark” instances in the test method. After that, I ensured “new\_carpark” instance is used consistently in the test method. Then I checked the log file if its logging correctly the expected entries and exits.

Attach three screenshots of your GitHub repo demonstrating discrete steps in your development process (hint: if you didn’t do it in real time, you can use your revision history!





## Submission

Your final submission should include:

* Your local git repo in a zip file.
* Your .git/ **must** be included with your zip file.
* Your .gitignore file should exclude any files that are not required for marking
* Do **not** include ~~venv/~~ in your submission.
* This worksheet, completed with your documentation and optionally the completed carpark project guide.

## Assessment Criteria

You will be assessed on:

* The correct implementation of OOP concepts.
* Code functionality and adherence to the provided specifications.
* Quality and clarity of code documentation.
* Successful execution and documentation of unit tests.
* Appropriate commit history and steps as per the core instructions.
* Answers to questions and/or evidencing criteria