**Bachelor of Science with honors in Computing Science/ Bachelor of Science with honors in Cyber Security**

**Course No 1**

Algorithms and Data Structures

**Module Code : 502IT**

**Submission Date : 01/June/2024**

**SUBMITTED BY:**

**Name :**

**CU Index Number :**

**Word Count :**

Contents

[**Introduction & Problem Analysis 2**](#_Toc167856179)

[**Introduction: 2**](#_Toc167856180)

[**Problem Analysis: 3**](#_Toc167856181)

[**Use Case Diagram. 3**](#_Toc167856182)

[**Use Case Diagram Description: 4**](#_Toc167856183)

[**UML Class Diagram 5**](#_Toc167856184)

[**UML Class Diagram Description: 5**](#_Toc167856185)

[**Code to test the program with appropriate test cases 7**](#_Toc167856186)

[**Data Structures: 10**](#_Toc167856187)

[**Coherence: 10**](#_Toc167856188)

[**Data Structure Consistency: 10**](#_Toc167856189)

[**Harmony: 11**](#_Toc167856190)

[**Summary Report 11**](#_Toc167856191)

[**1. Unified Modeling Language: 11**](#_Toc167856192)

[**2. Solution design through diagrams 11**](#_Toc167856193)

[**3. Object Oriented Programming and Python: 11**](#_Toc167856194)

[**References 12**](#_Toc167856195)

# Introduction & Problem Analysis

## Introduction:

**The task is to create a technical solution for a real world problem. The main focus is not the solution itself, but how we approach the problem, plan for the design of its solution and finally output a technical solution of that problem. We are going to use object oriented modeling tool, UML (Unified Modeling Language) 2.0 to design the solution of this real world scenario. UML is a graphical way of describing computer systems. There are 2 ways to approach a problem through UML; one is UML blueprint, in which we plan in detail and this approach usually out puts a design which is directly translatable to writing a code. The other one is UML sketch; we create a rough sketch of design of the solution for the problem at hand. The output through this is usually not translatable to a code directly. We will make a UML blueprint that will contain all details of static and dynamic part of the system that will be making. It is to be noted that UML architect that we will make will be platform independent, so it can be used to solve the problem on another platform as well. The platform dependent architects are specific for a particular platform and therefore, cannot be deployed on another platform.**

**The UML has 2 types of development processes.**

* **Waterfall**
* **Iterative**

**The waterfall method incorporates all aspects in a sequence like its name suggests. It should be used when the processes and end goal of the project is well known in advance. Its stages are Analysis, Design, Coding and Testing. The iterative method divides the task in smaller sub tasks and applies waterfall method on each part separately. The iterative method is easy to deploy and recommended when there is not much clarity about the process and it has to evolve during development process. The real world scenario at hand is very detailed, so we will use waterfall development process.**

**Similar to development processes, UML has 2 types of planning processes.**

* **Predictive Planning**
* **Adaptive Planning**

**We are going to use predictive planning process which should be used when we can list all the requirements beforehand. The real world problem at hand is quite explanatory, which is to be solved using Object Oriented Programming. We can analyze all the requirements and use cases in analyses, use case diagram and class diagram.**

**After making a firm ground for solving the task at hand, we will analyze the problem, look at it through use case diagram and UML diagram e.g. class diagram. We will support our written text through rational reasoning and code snippets.**

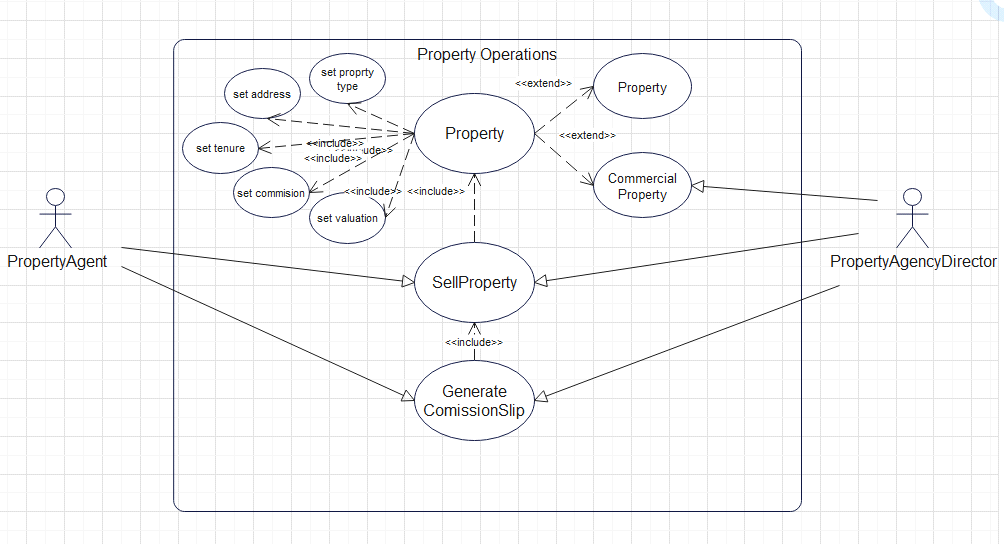
## Problem Analysis:

**In order to solve the problem at hand we break down the procedure in steps. In the first step we will explore UML and object oriented concepts. In the second step we will see the use cases and potential solutions through UML diagram. In the third step we will write up the code for the solution of the particular problem at hand. We will also test the resulting software with different inputs to check if the solution works correctly.**

**In order to execute the first step, we have gone through a few articles on UML and object oriented concepts. This is the most crucial step as it lays the foundation of the knowledge needed to solve the assignment. In the second step, we will create the use case diagram and UML diagram. The use case diagram will have actors and use cases. The actors and use cases will be correlated rationally. After that out goal will be to incorporate all the use cases and actors in UML class diagram in such a way that they will be directly translatable to a coding language. In the third step we will write up the code for the particular problem. We will use the python programming language and we will use object oriented programming to enable object oriented application into our solution. After that we will run the program and test it using different input values.**

**We will adopt a general approach, so we can apply these steps to any problem.**

# Use Case Diagram.

****

**Figure 1: Use Case Diagram showing Actors, use cases and their relation**

## Use Case Diagram Description:

**The use case diagram communicates different functions and classes a system needs to have in a very compact graphical method. The main components of a use case diagram are actors, System container, use cases and connections. The actors are the persons who are going to interact with the system; they are the real world user of this system. They are usually shown as sketched human. There is a large rectangle in the middle of the use case diagram, it is called system container. The actors are placed outside of the system container and the use cases are placed inside the system container. The use cases of the system are represented by ellipse. An ellipse can be extending ellipse and including ellipse shown by relevant connections. The connecting lines show the connections within the system and outside the system container as well. A dotted line with text <<extend>> means this use case provides an additional optional use case. A dotted line with text <<include>> means this use case is connected with another use case and it takes some input from that use case. The solid lines with arrow head show how actors can interact with the system.**

**The main two functions of this system are shown on the top which are setting different attributes of the property and selling of the property. The property has 2 extensions which are property and commercial property. At the bottom, there is a use case of GenerateSlip. By using GenerateSlip, property agent and director can find about their total earning. The details are as follow:**

* **Actors**

**a) Property Agent: The property agent can be an employee of the Property agency. It will have 2 lists of properties. One list will contain details of the sold properties and the other list will contain details of the properties which are to be sold. The property agent's share is 70 percent by default.**

**b) Property Agency Director: The property agency director is a higher level employee of the property agency. It extends from the Property Agent class. In addition to the attributes and methods of the property agent class, the property agency director will include an additional data member which is commission sharing rate from agents working under him. The default value for this will be 5% reaching to a max of 15%. It is also known as overriding commission.**

* **Use Cases**

**a) Sell Property: Both the actors have the use case to sell a property. Both actors have 2 lists of properties. When a property is sold, it is removed from the list of hold properties and is added to the list of sold properties. The use case "Sell Property" verify the property from the list of properties which have the following further use cases:**

**i) Set property type: This use case provide us the facility to set the type of property, whether it is a house, flat, kitchen apartment or another kind of property.**

**ii) Set address: This use case provide us the facility to set the address of the property which is helpful in selling the property to the clients to whom it suits the most. The address also has the impact on the valuation of the property.**

**iii) Set tenure: In the context of property dealing, the term tenure means whether the property is on lease, rent or any other condition. It directly influences valuation of the property.**

**iv) Set commission: The commission is the amount that the property agent will get upon sale. By default, its value is 1 percent and it can be changed for specific property.**

**v) Set valuation: Valuation corresponds to the value of the property.**

**The use case diagram is very helpful to understand all the application that you want your system to have. In this system a property agency can use this software to manage their properties, agents and directors. They can also very conveniently get and set the attributes of properties, agents and directors. And finally, the calculations of their salary will be made easy by the use case of get commission slip.**

# UML Class Diagram

## UML Class Diagram Description:

**The UML class diagram is the graphical way to represent different classes in your system and how they interact with each other. In our system diagram we will have 5 classes which are PropertyAgent, PropertyAgencyDirector, CommissionSlip, Property and CommercialProperty. Any class can be represented by a box with three sections. The 1st section contains the name of the class. The second section contains the name of different attributes of the class. The attributes are usually public, so they are represented by "+". Each attribute starts with a prefix +, - or # showing if it is public, private or restricted. After that there is the name of the attribute followed by its type. In the PropertyAgent class the first attribute is name and its type is string because of save the names as string in programming language. The data type of registration number is integer because it is expected that the registration number will contain numeric values.**

**The last section contains the functions or methods of the class. The functions/methods are usually not public, they are set as private and they are represented by "-". If a function is neither public not private, but restricted, it is represented by "#". The functions are represented by their name and then they are followed by "()", which specifies that they are functions or methods. Then they are followed by what the function returns. If the function simply brings change to the system without out putting something, the return is void for that function. If the function outputs something like a slip, the return will not be void.**

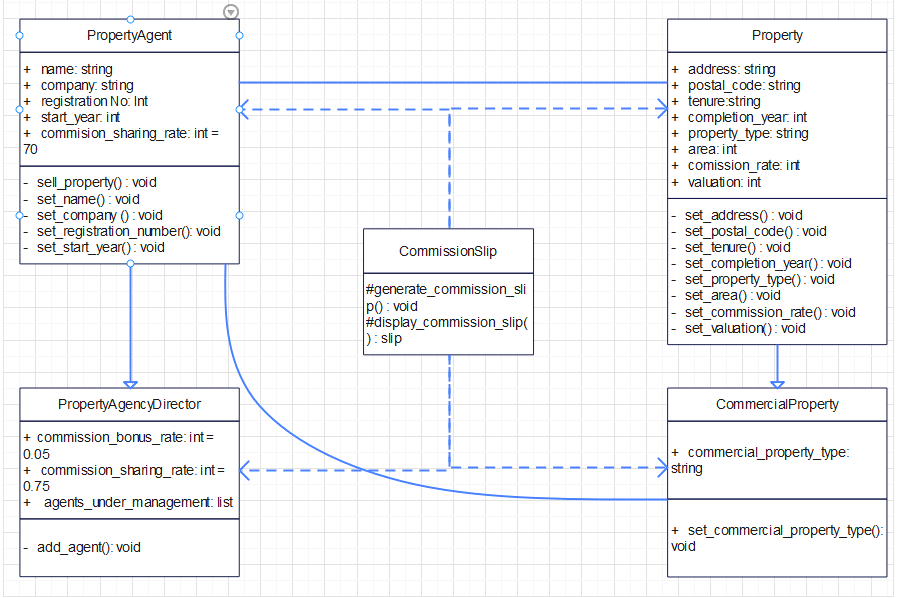
**The classes in this UML diagram are connected to each other through different types of connections and arrow heads. These connections and arrow heads are not chosen randomly. Each line type and arrow head contains a meaning and these connections and arrow heads are then directly translatable to a programming language.**

**The solid line without any arrows shows association between different classes. This association does not make a class dependent on another class, they simply shows that there is some connection between these 2 classes. For example, the property agent and property may not be dependent on each other but we know that there is some connection between property agent and the property, so we have connected these two classes with association connector. Similarly, property agent is also associated with the commercial property as well. The property director is also associated with these two classes but we have not shown the connection to keep a room for showing other connections, moreover, property agency director is inherited from property agent class, so even if we don’t show the connection it should be understood that property agency director is associated with all the associated of the property agent class. The dotted line with simple arrow head "- - - - >" shows dependency. The CommissionSlip class depends on property agent, agency director, property and commercial property because it takes values from these classes in other to calculate commission of property agent and agency director. Lastly, the connection line with plain line and blank arrowhead shows a special kind of dependency, which is inheritance. The property agency director is inherited from property agent class and commercial property is inherited from property class, so there is this line and arrowhead between property agency director and property agent, and between property and commercial property. The methods of commission slip class are restricted because only a few actors should be able to access the commission slip for a person due to ethical reasons.**

**The property class serves as the central component in this system. It does not take any dependency from another class. All of its attributes and methods are its own. The commercial property class is inherited from property class. The property agent and agency director class take attributes from property and commercial property classes. Lastly, the commission slip class is dependent on all the classes to calculate a detailed commission showing the commission per agency directory, per agent, and also the detailed distribution of commission from different properties. We can say that commercial property, property agent, agency director and commission slip are sub classes or child classes of the property class. The property class can be called as parent class or super class.**

**By showing the overall attributes of the classes, method and their relationships, we can now work on to create the solution of this real world problem.**

**UML Diagram:**

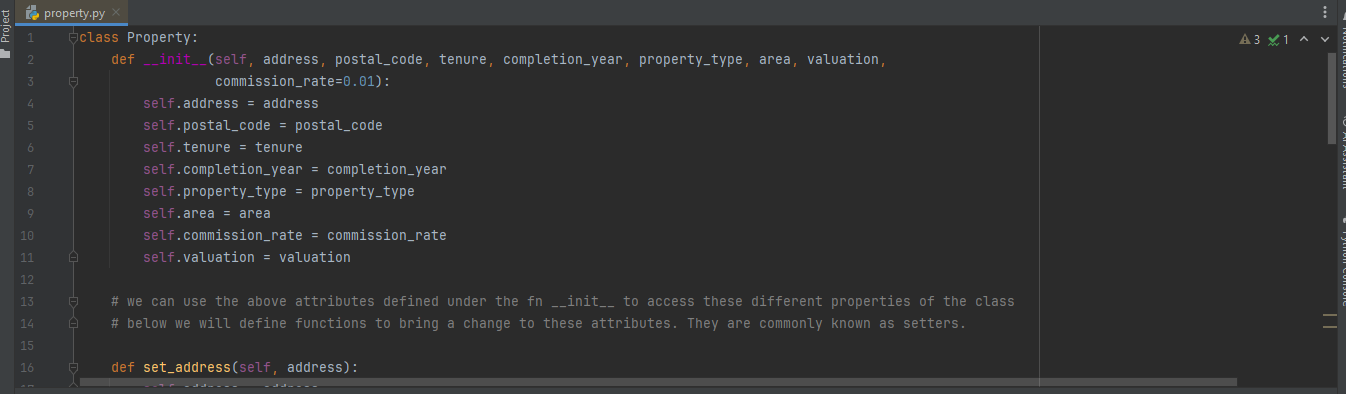
****

**Figure 2: UML Class Diagram showing classes, methods, attributes and their interrelation**

# Code to test the program with appropriate test cases

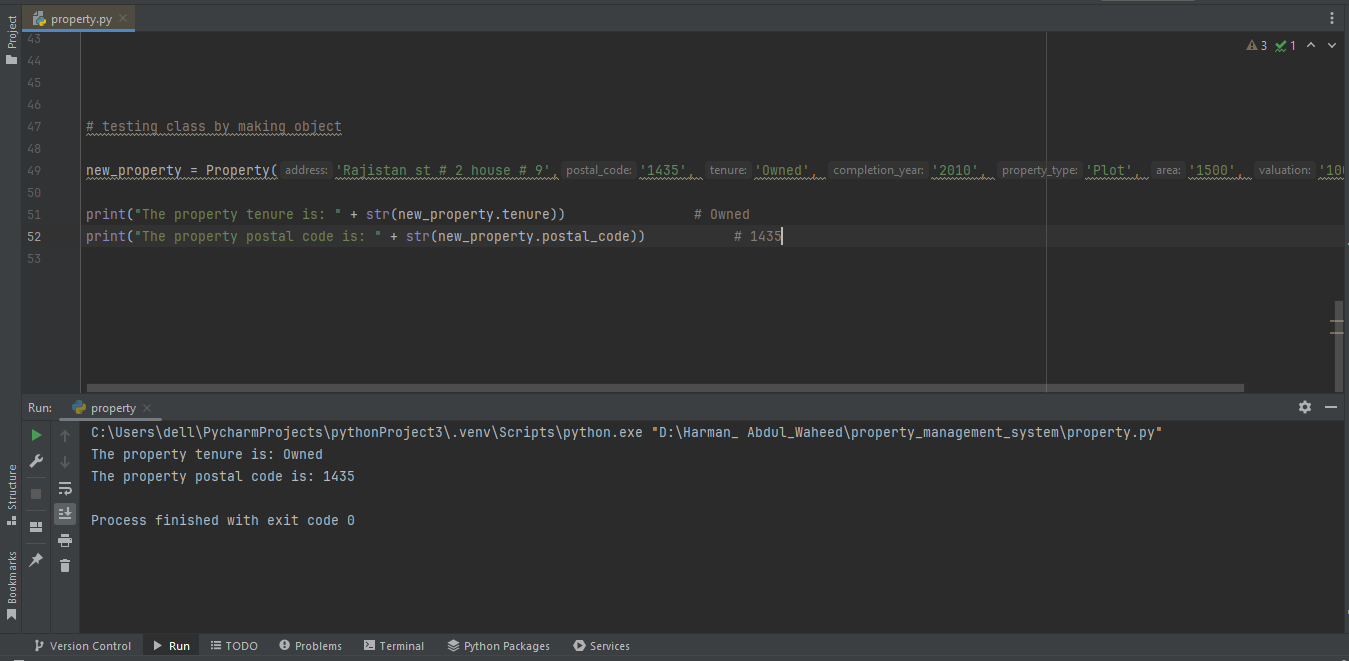
**The python code was written using the use cases and UML class diagram. We used the predictive planning using the waterfall development process as stated in the beginning. The use cases of the use case diagram and interconnectivity of classes in UML class diagram was very helpful as they were directly translatable into python programming. We used object oriented programming and made different classes of all the classes shown in UML class diagram. Inheritance, dependence was established using UML class diagram as well. The .py file has been attached with the assignment showing the code.**

**Furthermore the code was tested to see if the desired output is given or not. Below I have pasted a few screen shots to show some test cases.**

****

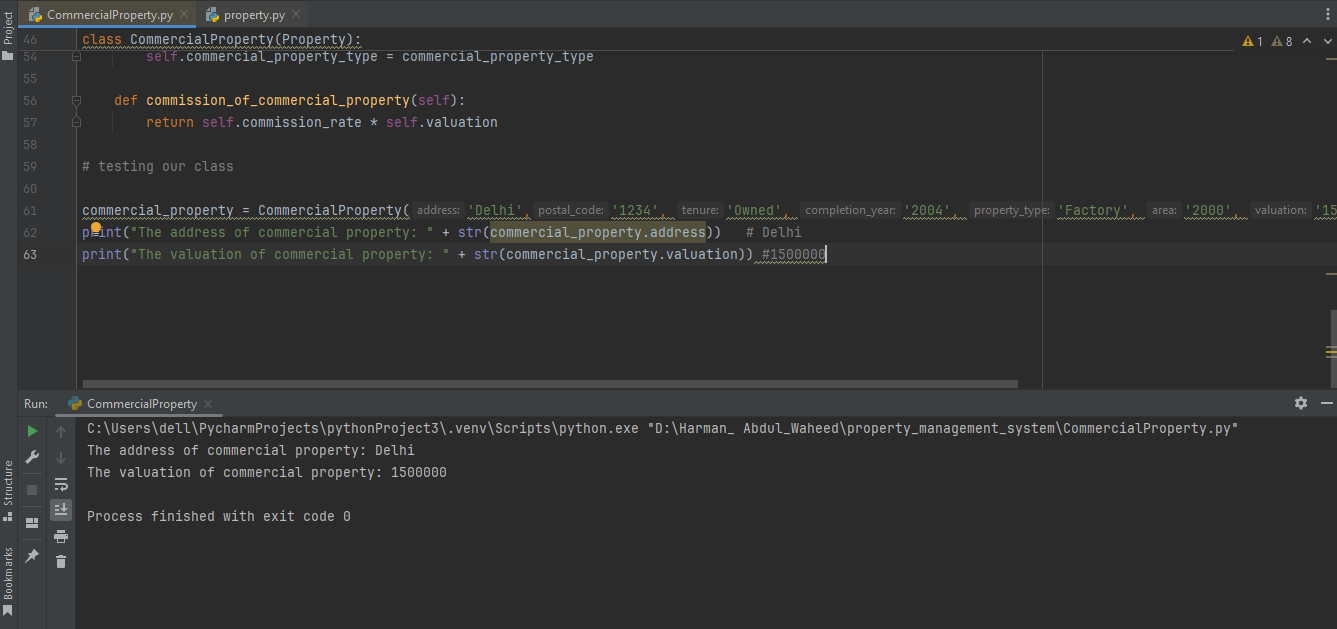
**Figure 3.1: Property class is being initiated with different attributes and methods**

**In this screenshot we have shown how to create the class. The syntax can be used to create any kind of class in Python. The def \_\_init\_\_ is known as constructor and it is used to make attributes of the class. Moreover, the functions defined in the class work as method of the class. The attributes are called with the syntax class\_name.attribute and the methods are called with the syntax class\_name.method(). The parenthesis at the end of method name suggests that it is a method/function of the class.**

****

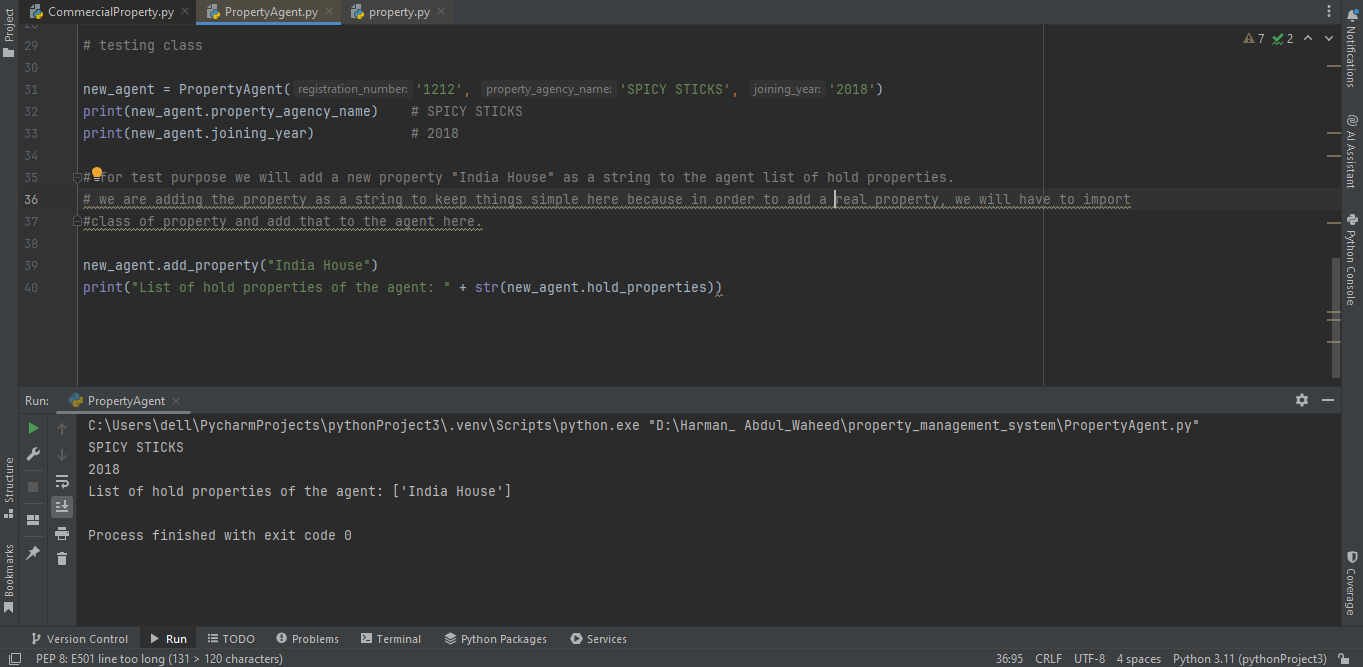
**Figure 3.2: 2 attributes of property class’ object are accessed and printed**

**In the figures above an object of property class has been created and tested. We created an object of property class and printed 2 of its properties. It ran without an error which shows that it has passed the test case.**

****

**Figure 4: Test case of Commercial Property class**

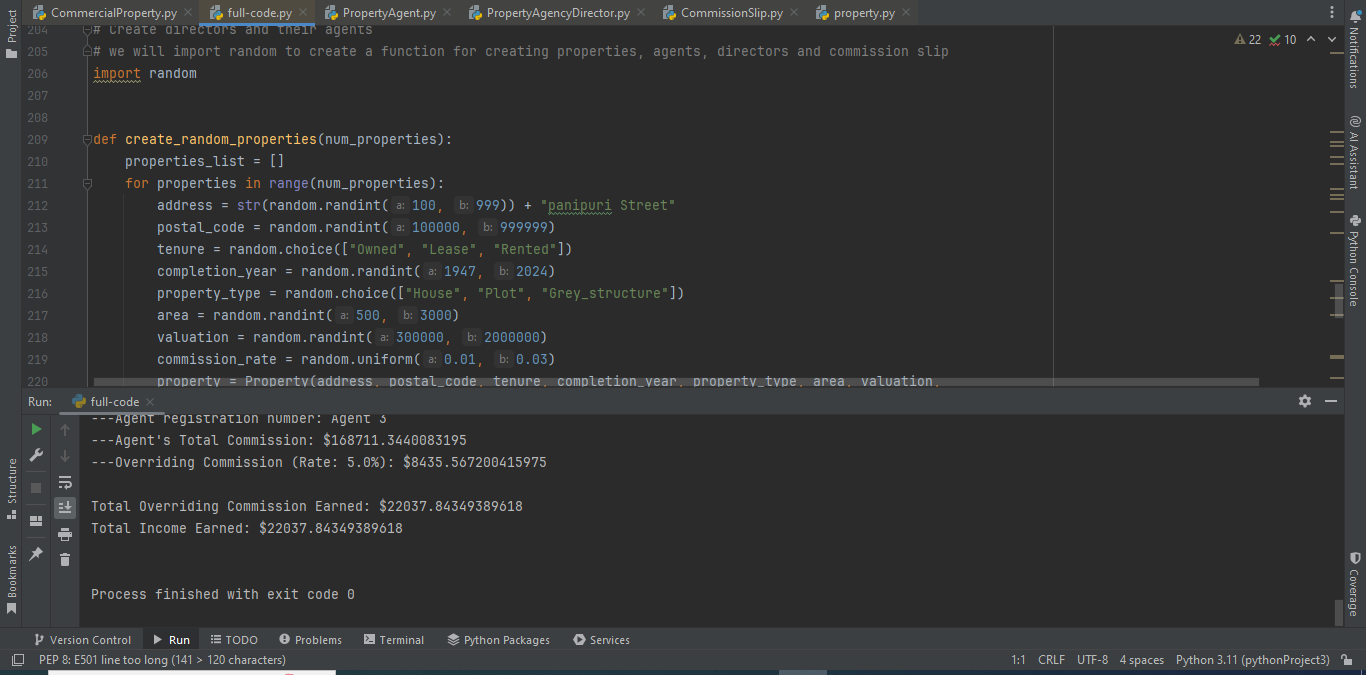
**In the figure above, the commercial property class has been created. In order to create the commercial property class, we had to create the property class first because it would throw an error if we create a child class without the parent class.**

****

**Figure 5: Test case of PropertyAgent class**

**In the screenshot above, we created a property agent with name new\_agent. The new\_agent is an object of PropertyAgent class which has a few attributes and methods. We have printed a few attributes, added a property to the list of hold properties of this agent in order to test the code.**

**We used random function to create objects/instances of the classes of property, commercial property, and property agent and agency director. For address a random value between 100 and 999 was chosen along with a street name which is string. Postal code was chosen between 100000 and 999999 to make it a 6 digit post code. Tenure was chosen between freehold and leesehold randomly. Completion year was chosen between 2000 and 2023. Property type was chosen from the choices between House, Plot or grey structure randomly. The area was a random choice between 500 to 3000 square feet. Valuation was also chosen randomly between 300000 to 2000000. The commission rate is set to 1 percent as default but was chosen between 1 to 3 percent randomly.**

****

**Figure 6: Running the entire code to create objects of all classes and printing out commission slip of agent and director**

**We created functions to create random properties, commercial properties, directors and their agents. We tested is numerous times and the result being out putted were correct all the times.**

# Data Structures:

**It is important to have a familiarity with data and data structures in order to handle complex problems. We took care to use the relevant data structures in our assignment. There are various data structures in in python language that we can use to solve a problem while using object oriented programming. However, care must be taken of according to the below mentioned key points:**

## Coherence:

**The data structure that we chose to use for our data must be coherent with the type of the system that we are going to build. For example list data type should be used to store the data type which does not have properties associated to it which needs to be stored along with the list objects. If we need to save associated properties along with the objects, we will need to use the data type array which is more suitable for storing rows and columns**

## Data Structure Consistency:

**It is crucial to stay consistent in the approach of using data structures. For example, if we have used a specific data structure for a particular case, care must be taken to use that same data structure for the next instances as well. If the consistency of the data structure is compromised, it can lead to confusion, as well as error while writing the code. Consistency of the data structure helps to ensure smooth running of operations.**

## Harmony:

**Harmony should be observed in selecting the data structure. The data being stored must be compatible with the data structure. In the context of Python, one dimensional data can be stored in array or lists. Arrays should be used if the data type is consistent. Lists should be used if the data type is not consistent. Array can be used to save higher dimensional data as well. For example 2D or 3D data can be stored in arrays. To save the data in higher dimensions, tensors are recommended.**

# Summary Report

**This assignment provided an exciting way to add new skills to the inventory of our problem solving skills. We were able to grasp a firm grip on the following topics:**

## 1. Unified Modeling Language:

**Designing a technical solution using UML has improved our understanding of using algorithms to solve a problem. Before going through this assignment, we used to solve the problem at hand without a sequential approach. The problems were being solved by that method as well but those methods could not be applied to solve other problem. After completing this assignment, we have developed the skills to design a technical solution for any problem at hand. The steps that we used to solve this particular problem can be applied to any other real world problem as well to reach to a solution. This means, the solution designed for this assignment is a general design which can be reused to solve another problem at hand.**

**Familiarization with the concepts like water fall approach and iterative approach, predictive and adaptive planning has been a great help and helped us to see the solution of the problem in a structured way before even starting to write the code. By employing such techniques we can list all the requirements and functions of software in advance.**

## 2. Solution design through diagrams

**Diagrams such as use case diagrams and UML class diagrams provides a very all rounded approach to analyze the problem, designing a solution and planning different functionalities of the system.**

**We were able to learn how a use case diagram provides the interaction between actors and use cases of the system. We also learnt how different use cases can interact with each other as well. In addition to this, we learnt how to plan a system in a comprehensive manner. We learnt how different arrow heads depicts different behavior in UML class diagrams. If a UML class diagram is created carefully, it can be directly translated to code. The use case diagram is like a pseudo code in a diagram form providing a very structured approach to solve the problems.**

## 3. Object Oriented Programming and Python:

**We were able to familiarize ourselves with the object oriented programming concepts. By creating different classes, useful attributes and required functions we were able to set relations between classes. The concepts like inheritance and dependency of classes on one another added new skills to our inventory of problem solving tools. We were also familiarized with the concepts of attributes and methods.**

**We appreciate that object oriented programming can be used to solve almost any task. We can make different classes with relevant attributes and methods to solve the tasks at hand. Moreover, it aligns with the structured way of solving problems.**

**Moreover, we got familiarized with python language which is being used the most for programming these days. While writing the code for case study, we learn syntax of the language, classes and objects in python.**

**Conclusion:**

**This assignment has provided a comprehensive knowledge base regarding UML, object oriented programming and python. We learnt how to design a solution for any problem related to creating a system or software. The use case diagram and UML class diagram provided us with the knowledge of creating a structured planning for a solution. The object oriented programming in python is also a new tool in our inventory of problem solving tools.**

**This assignment has laid a foundation for further knowledge and work. Creating a software system that can be used in real world companies has given the confidence to have an approach to solve real world problems with my programming and problem solving skills.**

# References

**OMG. (n.d.) What is UML? Available at: https://www.uml.org/what-is-uml.htm (Accessed: 28 May 2024)**

**Python Software Foundation. (n.d.) The Python Tutorial: 9. Classes. Available at: https://docs.python.org/3/tutorial/classes.html (Accessed: 28 May 2024)**

**Booch, G., Rumbaugh, J. and Jacobson, I. (2005) The Unified Modeling Language User Guide. 2nd edn. Boston: Addison-Wesley**