



Higher Nationals in Computing

**Unit 20: Advanced Programming  
ASSIGNMENT 1**

Learner’s name: Nguyen Phu Quy

ID: GCS190822

Class: GCS0901

Subject code: 1651

Assessor name: **PHAN MINH TAM**

Assignment due: February 20, 2022Assignment submitted: February 20, 2022

**ASSIGNMENT 1 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | **Unit 20: Advanced Programming** | | |
| **Submission date** | February 20, 2022 | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Phu Quy | **Student ID** | GCS190822 |
| **Class** | GCS0901 | **Assessor name** | Phan Minh Tam |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** |  |

**Grading grid**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | M1 | M2 | D1 | D2 |
|  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Signature & Date:** | | |

**ASSIGNMENT 1 BRIEF**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Business** | | |
| **Unit number** | Unit 20: Advanced Programming | | |
| **Assignment title** | Examine and design solutions with OOP and Design Patterns | | |
| **Academic Year** | 2021 - 2022 | | |
| **Unit Tutor** | Phan Minh Tam | | |
| **Issue date** | 28 September 2021 | **Submission date** | February 20, 2022 |
| **IV name and date** |  | | |

|  |
| --- |
| **Submission Format:** |
| *Format:* The submission is in the form of a **group written report**. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system.  *Submission* Students are compulsory to submit the assignment in due date and in a way requested by the Tutors. The form of submission will be a soft copy in PDF posted on corresponding course of <http://cms.greenwich.edu.vn/>  *Note:* The Assignment *must* be your own work, and not copied by or from another student or from  books etc. If you use ideas, quotes or data (such as diagrams) from books, journals or other sources, you must reference your sources, using the Harvard style. Make sure that you know how to reference properly, and that understand the guidelines on plagiarism. *If you do not, you definitely get fail* |
| **Assignment Brief and Guidance:** |
| **Scenario**: You have recently joined a software development company to help improve their documentation of their in-houses software libraries which were developed with very poor documentation. As a result, it has been very difficult for the company to utilise their code in multiple projects due to poor documentation. Your role is to alleviate this situation by showing the efficient of UML diagrams in OOAD and Design Patterns in usages.  **Tasks**  You and your team need to explain characteristics of Object-oriented programming paradigm by applying Object-oriented analysis and design on a given (assumed) scenario. The scenario can be small but should be able to presents various characteristics of OOP (such as: encapsulation, inheritance, polymorphism, override, overload, etc.).  The second task is to introduce some design patterns (including 3 types: creational, structural and behavioral) to audience by giving real case scenarios, corresponding patterns illustrated by UML class diagrams.  To summarize, you should analyze the relationship between the object-orientated paradigm and design patterns.  The presentation should be about approximately 20-30 minutes and it should be summarized of the team report. |

|  |  |  |
| --- | --- | --- |
| **Learning Outcomes and Assessment Criteria** | | |
| **Pass** | **Merit** | **Distinction** |
| **LO1** Examine the key components related to the object-orientated programming paradigm, analysing design pattern types | | |
| **P1** Examine the characteristics of the object-orientated paradigm as well as the various class relationships. | **M1** Determine a design pattern from each of the creational, structural and behavioural pattern types. | **D1** Analyse the relationship between the object-orientated paradigm and design patterns. |
| **LO2** Design a series of UML class diagrams | | |
| **P2** Design and build class diagrams using a UML tool. | **M2** Define class diagrams for specific design patterns using a UML tool. | **D2** Define/refine class diagrams derived from a given code scenario using a UML tool. |

**Table of Contents**

**[P1 Examine the characteristics of the object-orientated paradigm as well as the various class relationships. 1](#_Toc25505)**

[1. Define of Object-orientated paradigm. 1](#_Toc16231)

[2. Object-orientated paradigm characteristics. 2](#_Toc10162)

[2.1 Class. 2](#_Toc20271)

[2.2 Object. 2](#_Toc32268)

[2.3 Encapsulation. 3](#_Toc28201)

[2.4 Abstraction. 4](#_Toc8611)

[2.5 Polymorphism. 4](#_Toc19960)

[2.6 Inheritance. 5](#_Toc15902)

[3. Object-orientated class relationships. 6](#_Toc29163)

[3.1 Association. 6](#_Toc24245)

[3.2 Aggregation. 6](#_Toc252)

[3.3 Composition. 7](#_Toc7337)

**[P2 Design and build class diagrams using a UML tool. 7](#_Toc19655)**

[1. Scenario in Detail. 7](#_Toc21224)

[2. Use-case diagram. 8](#_Toc23879)

[3. Class diagram. 8](#_Toc1878)

[4. Sequence diagram. 9](#_Toc3923)

**ASSIGNMENT 1 ANSWERS**

**P1 Examine the characteristics of the object-orientated paradigm as well as the various class relationships.**

1. **Define of Object-orientated paradigm.**

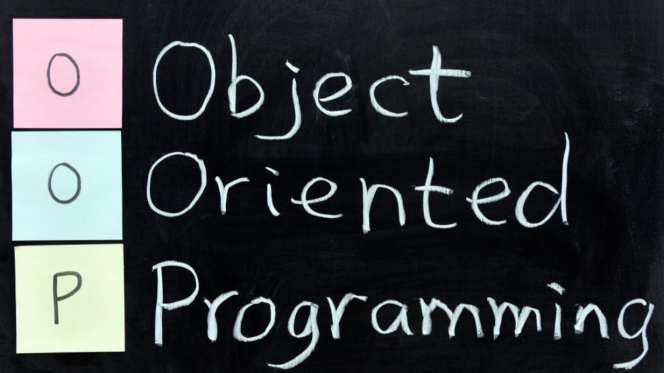
Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

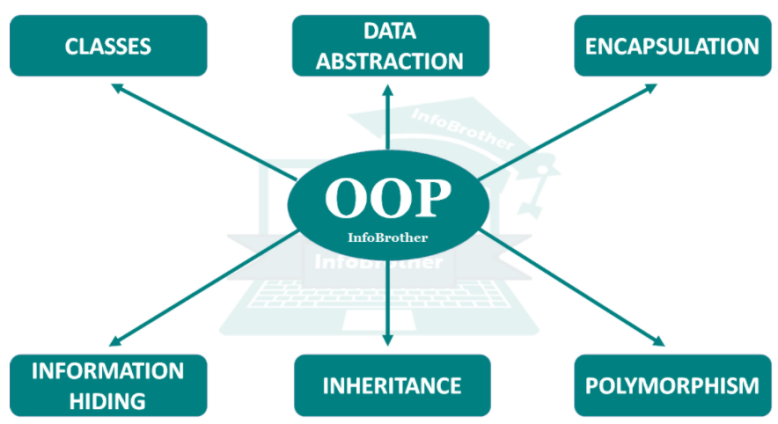
The first step in OOP is to collect all of the objects a programmer wants to manipulate and identify how they relate to each other -- an exercise known as data modeling.

**Examples** of an object can range from physical entities, such as a human being who is described by properties like name and address, to small computer programs, such as widgets. Once an object is known, it is labeled with a class of objects that defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. Objects can communicate with well-defined interfaces called messages



1. **Object-orientated paradigm characteristics.**

In the Object-oriented paradigm has six characteristics:



* 1. **Class.**

The building block of C# that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

For Example: Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

-A Class is a user-defined data-type which has data members and member functions.

- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behaviour of the objects in a Class.

- In the above example of class Car, the data member will be speed limit, mileage etc and member functions can apply brakes, increase speed etc.

* 1. **Object.**

An Object is an identifiable entity with some characteristics and behaviour. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Object take up space in memory and have an associated address like a record in pascal or structure or union in C.

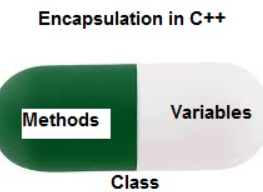
When a program is executed the objects interact by sending messages to one another.

Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other’s data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

* 1. **Encapsulation.**

In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.

Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.



Encapsulation also leads to data abstraction or hiding. As using encapsulation also hides the data. In the above example, the data of any of the section like sales, finance or accounts are hidden from any other section.

* 1. **Abstraction.**

Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is.

**- Abstraction using Classes:** We can implement Abstraction in C++ using classes. The class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to the outside world and which is not.

**- Abstraction in Header files:** One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header file. Whenever we need to calculate the power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers.

* 1. **Polymorphism.**

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

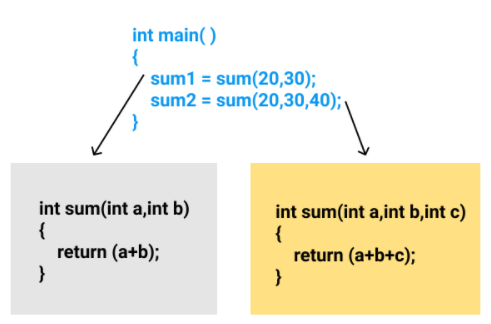
An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation.

C# supports operator overloading and function overloading.

**- Operator Overloading:** The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.

**- Function Overloading:** Function overloading is using a single function name to perform different types of tasks. Polymorphism is extensively used in implementing inheritance.

**Example:** Suppose we have to write a function to add some integers, some times there are 2 integers, some times there are 3 integers. We can write the Addition Method with the same name having different parameters, the concerned method will be called according to parameters.



* 1. **Inheritance.**

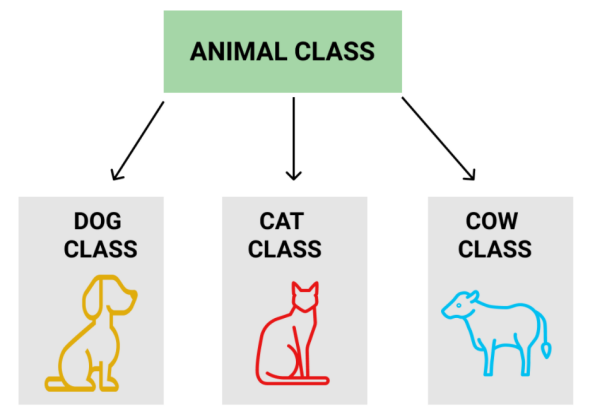
The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

**- Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.

**- Super Class:**The class whose properties are inherited by sub class is called Base Class or Super class.

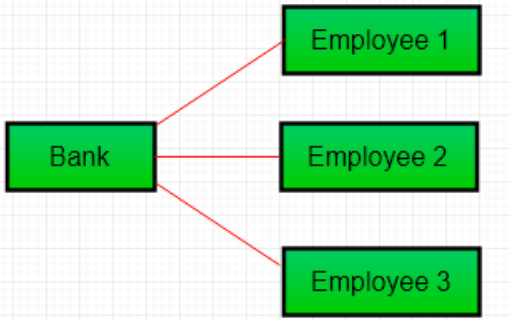
**- Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Example: Dog, Cat, Cow can be Derived Class of Animal Base Class.



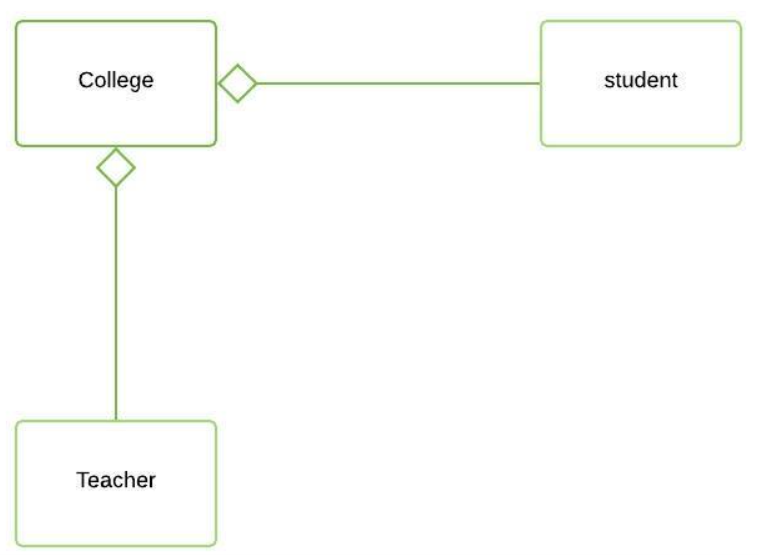
1. **Object-orientated class relationships.**
   1. **Association.**

Association is a relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many. In Object-Oriented programming, an Object communicates to another object to use functionality and services provided by that object. Composition and Aggregation are the two forms of association.



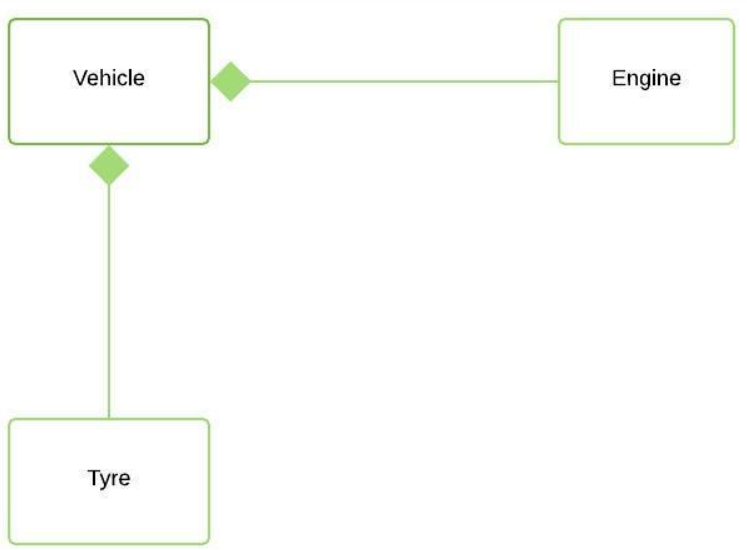
* 1. **Aggregation.**

It is a special form of Association where: It represents Has-A’s relationship. It is a unidirectional association i.e. a one-way relationship. For example, a department can have students but vice versa is not possible and thus unidirectional in nature. In Aggregation, both the entries can survive individually which means ending one entity will not affect the other entity.



* 1. **Composition.**

Composition is a restricted form of Aggregation in which two entities are highly dependent on each other. It represents part-of relationship. In composition, both entities are dependent on each other. When there is a composition between two entities, the composed object cannot exist without the other entity.

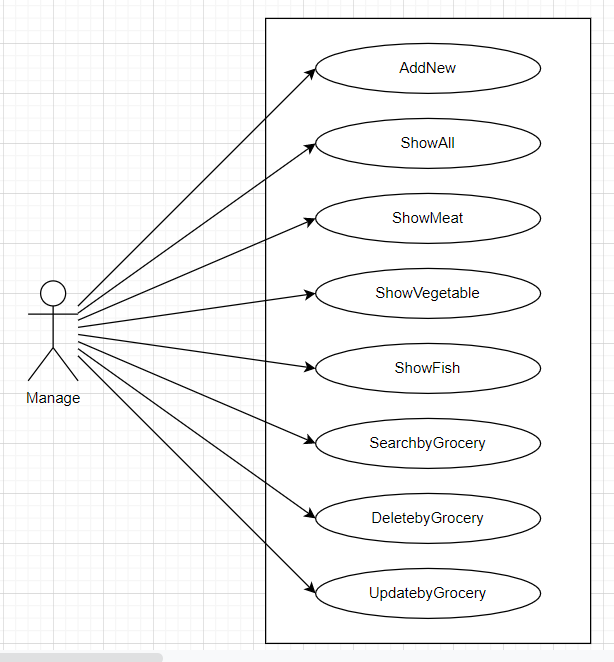


**P2 Design and build class diagrams using a UML tool.**

1. **Scenario in Detail.**

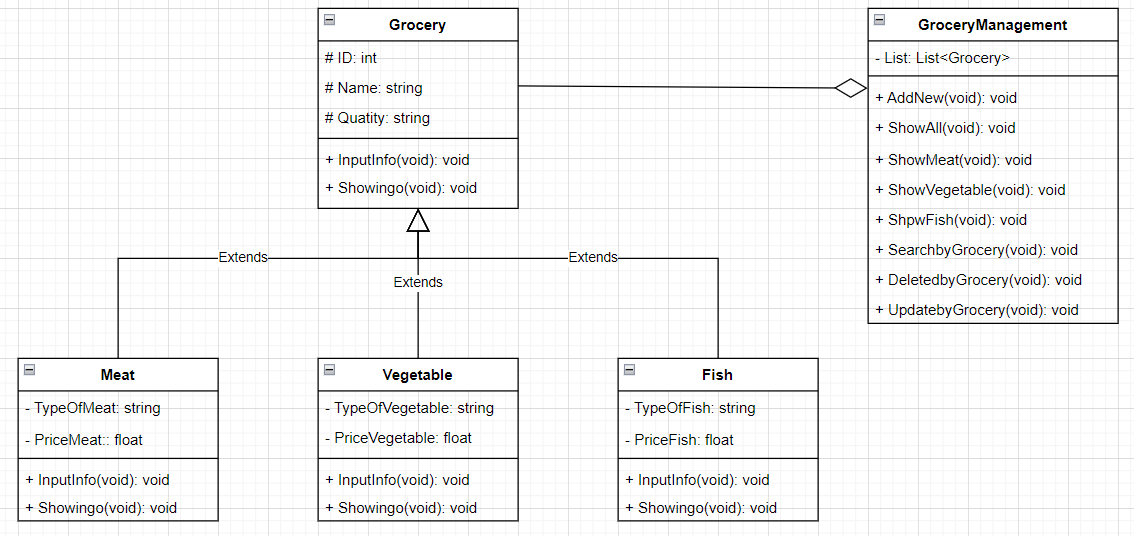
A friend of mine is about to open a grocery store to earn extra income for his family. he needs an app to manage the foods that will be sold in the store. So he asked me to design for him. I will draw a use-case diagram, class diagram and sequence diagram for him first.

1. **Use-case diagram.**



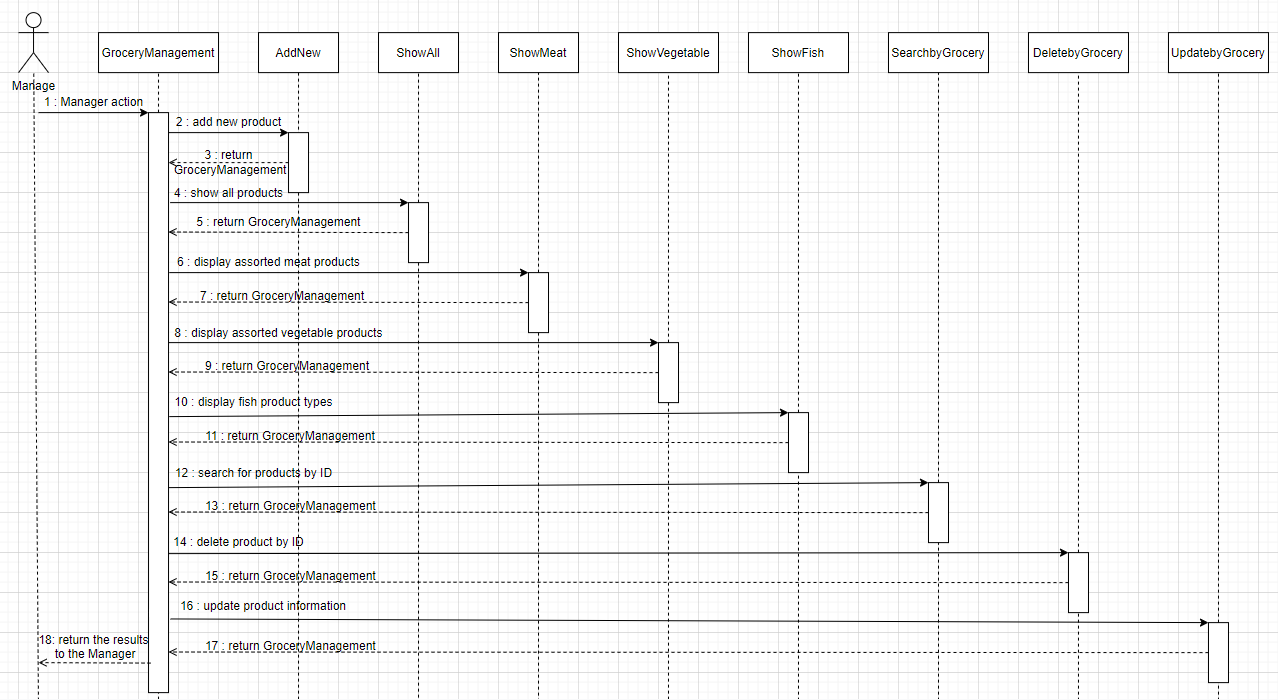
In the use case diagram, manager will manage the store using the above functions. AddNew to add new product information to the store. ShowAll will display all the product information of the store. ShowMeat will display information about meats. Similarly ShowVegetable and ShowFish will display the information of vegetables and fish. SearchbyGrocery helps to search products by that product ID. DeletebyGrocery will delete the product with that product ID. UpdatebyGrocery to update product information.

1. **Class diagram.**



In the Grocery class, there will be Attributes(ID, Name, Quantity) and Operation(InputInfo and OutputInfo). Other classes (Meat, Vegetable and Fish) will inherit the Attributes of the Grocery class and have its own new Attributes: Meat(TypeOfMeat,PriceMeat), Vegetable(TypeOfVegetable,PriceVegetable) and Water(TypeOfWater,PriceWater). class GroceryManagement will manage the products in the store from List<Grocery> with the Operations: AddNew(add new product); ShowAll(show all products); ShowMeat, ShowVegetable and ShowFish (showing different foods); SearchbyGrocery(search for products by ID); DeletebyGrocery(delete products by ID); UpdatebyGrocery(change product information).

1. **Sequence diagram.**



Starting GroceryManagement (class) will be where the user works. Every time the function is done, it returns GroceryManagement . The first function is to add new products to the store. the second function is to display all products in the store. Functions three, four and five display products of each type (Meat, Vegetable and Fish, respectively). Next, function six is used to search for products by ID, and function seven is used to delete products by ID. The last function is used to update product information. After completion, the results will be returned to Manage and finished.

**REFERENCES**

1. Wikipedia (2019) “ Object Oriented Programming “. [Online] Availabe at: https://vi.wikipedia.org/wiki/L%E1%BA%ADp\_tr%C3%ACnh\_h%C6%B0%E1%BB%9Bng\_%C4%91%E1%BB%91i\_t%C6%B0%E1%BB%A3ng [Accessed February 16, 2022]
2. techtarget (2020) “ object-oriented programming “. [Online] Availabe at: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP [Accessed February 16, 2022]
3. Geeksforgeeks (2019) “ Object Oriented Programming in C# “. [Online] Availabe at: <https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/> [Accessed February 16, 2022]
4. Geeksforgeeks (2019) “ Association, Composition and Aggregation in Javat “. [Online] Availabe at: https://www.geeksforgeeks.org/association-composition-aggregation-java/ [Accessed February 16, 2022]
5. Viblo (2020) “ Learn how to design Class Diagram “. [Online] Availabe at: https://viblo.asia/p/tim-hieu-ve-cach-thiet-ke-class-diagram-L4x5xLyY5BM [Accessed February 16, 2022]
6. iviettech (2018) “ DRAWINGS USE CASE (USE CASE DIAGRAM) “. [Online] Availabe at: https://iviettech.vn/blog/543-ban-ve-use-case-use-case-diagram.html [Accessed February 16, 2022]
7. Codelearn (2018) “ UML sequence diagram “. [Online] Availabe at: https://codelearn.io/sharing/sequence-diagram-trong-uml [Accessed February 16, 2022]
8. Drive (2022) “ Record microsoft team “. [Online] Availabe at: https://drive.google.com/file/d/1ybtwe46lF4uYksTUpIWuAVviFupo8TN2/view?fbclid=IwAR2NLvAh1n\_wvQHARhp7N9AmJmcvD3kFfDHBET02c7\_QhUFT6HvBLkbo\_Ro [Accessed February 16, 2022]
9. IBM (2019) “ Use-case diagrams “. [Online] Availabe at: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case [Accessed February 16, 2022]
10. Tutorialspoint (2020) “ UML - Use Case Diagramsa “. [Online] Availabe at: https://www.tutorialspoint.com/uml/uml\_use\_case\_diagram.htm [Accessed February 16, 2022]