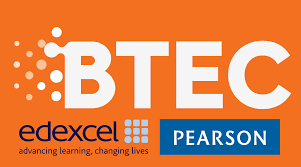
**BTEC Level 5 HND Computing and Systems Development**



A Project Object Oriented Programming Presented

by

Nguyễn Đình Tùng(GC00952)



Submitted to the Graduate School of the

University of Greenwich Vietnam in partial fulfillment

Friday, 20th April 2018

Lecturer: Đỗ Hồng Quân

**Table of Contents**

[**Introduces** 4](#_Toc513274429)

[**Task 1: Discuss the principles, characteristics and features of object orientated programming.** 5](#_Toc513274430)

[1. What is Object Oriented Programming(OOP)? 5](#_Toc513274431)

[1. Benefits of Object Oriented Programming 5](#_Toc513274432)

[2. Principles, characteristics and feature of OOP 5](#_Toc513274433)

[**Task 2: Identify the objects and data and file structures required to implement a given design and Design an object orientated programming solution for a given problem.** 12](#_Toc513274434)

[1.1 Use-case diagram for Software management students and lecturers of FPT Academy 12](#_Toc513274435)

[1.2 Class diagram 14](#_Toc513274436)

[1.3 Flowchart 15](#_Toc513274437)

[**References** 19](#_Toc513274438)

**Table of Figures**

[Figure 1.class Car.java 7](#_Toc513274457)

[Figure 2. Class main to create objects 7](#_Toc513274458)

[Figure 3. Class abstract 9](#_Toc513274459)

[Figure 4. Class super Employee 10](#_Toc513274460)

[Figure 5. Class sub Programmer 11](#_Toc513274461)

[Figure 6.Use-case diagram for Management FPT Academy 13](#_Toc513274462)

[Figure 7. Class diagram for Management FPT Academy 14](#_Toc513274463)

[Figure 8. Search ID with data text saved 15](#_Toc513274464)

[Figure 9. Search Name with data text saved 16](#_Toc513274465)

[Figure 10. flowchart for add a student 16](#_Toc513274466)

[Figure 11. Flowchart for update student 17](#_Toc513274467)

[Figure 12. Flowchart for delete student 17](#_Toc513274468)

[Figure 13. Flowchart for search student 18](#_Toc513274469)

# **Introduces**

This report I will write about What is Object Oriented Programming(OOP)?, Benefits of Object Oriented Programming, Principles, characteristics and feature of OOP, Use-case diagram for Software management students and lecturers of FPT Academy, Class diagram and flowchart for main functions.

# **Task 1: Discuss the principles, characteristics and features of object orientated programming.**

## What is Object Oriented Programming(OOP)?

The define this [1] write about what is OOP below:

**“**Object-oriented programming (OOP) is a software programming model constructed around objects. This model compartmentalizes data into objects (data fields) and describes object contents and behavior through the declaration of classes (methods).

OOP features include the following:

* Encapsulation: This makes the program structure easier to manage because each object’s implementation and state are hidden behind well-defined boundaries.
* Polymorphism: This means abstract entities are implemented in multiple ways.
* Inheritance: This refers to the hierarchical arrangement of implementation fragments.
* Abstraction: Abstraction helps you focus on the object rather than the way it works.

Object-oriented programming allows for simplified programming. Its benefits include reusability, refactoring, extensibility, maintenance and efficiency.**”**

## Benefits of Object Oriented Programming

As object-oriented programming comes into play, it overcomes all the weaknesses of earlier programming methods. Specifically it has the following advantages:

- Easily manage code when program changes.

- Easy to extend project.

- Saves significant resources for the system.

- High security.

- Highly recyclable.

## Principles, characteristics and feature of OOP

* 1. **Principles**

Referring to the OOP principle we are reminded of the SOLID principle, the SOLID principle makes it easy for designers to change the software architecture in response to the latest changes. "SOLID" is a set of 5 principles:

- **Single responsibility principle:** A class should only hold a single responsibility (ie, only modify that class for a single reason).

- **Open/closed principle:** It is possible to extend the extension of a class, but not modify it inside the class (open for extension but closed for modification).

- **Liskov substitution principle:** In a program, objects of the subclass can substitute the parent class without altering the correctness of the program.

- **Interface segregation principle:** Instead of using a large interface, we should split into several small interfaces, for many specific purposes.

- **Dependency inversion principle:** High level modules should not depend on low level modules. Both should depend on abstraction. Interface (abstraction) should not depend on details, but vice versa. (Classes communicate with each other through an interface, not through implementation.)

* 1. **Characteristics and Feature**

**b1: Object/Class**

**\* Object:**

* In object-oriented programming, the object is interpreted as an entity: a person, object or a data table. . .
* An object consists of two pieces of information: properties and methods.
* Main attribute is the information, characteristics of the object. For example: a person will have a name, date of birth, color, hair style,. . .
* Modes are actions, actions that the object can perform. For example: a person will be able to perform the act of speaking, walking, eating, drinking,…

**\* Class:**

* Objects with similar characteristics are grouped into one object class.
* Inside the class there are two main components that are attributes and methods.
* In addition, the class is used to define new types of data.

**\* Different between Object and Class:**

* A class is a template, and the object is a concrete instance based on that template.
* To make it easier to understand I will take a practical example:
* Talking about the cat, the main class is the cat. Cats have:
* Information, such as 4 legs, 2 eyes, tail, height, weight, coat color…
* The actions such as: meow, go, eat, sleep,…
* So all animals of the cat will have these characteristics.
* The main object is a specific cat like a kitten lying at his feet.
* Another example. For this example I will use code to illustrate:

This below is a class Car.java

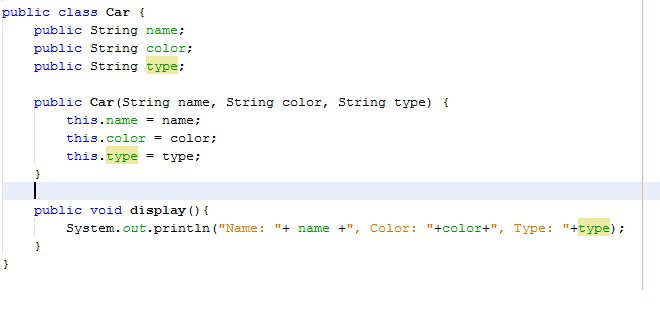


Figure 1.class Car.java

* Class Car.java include fields String name, String color, String type. It also have method display() and a constructor with three fields
* And I have class main() below:

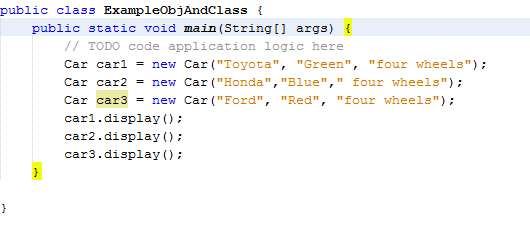


Figure 2. Class main to create objects

* This class, we can see have three objects this car1, car2 and car3. These objects contains the fields of class Car.java and it is a specific car with name, color and type.
* The emergence of two new concepts is that the class and object are characteristics of the object-oriented programming method. It solves the shortcomings of structured programming methods left behind. In addition, these two concepts help to better represent the real world on the computer.

**b2: The four main characteristics of object-oriented programming**

**b2.1: Abstraction, virtual mode**

**+ Definition:** It is the ability of the program to ignore or disregard certain aspects of the information that it is directly working on, ie it is possible to focus on the core essentials (only newspaper). Each object serving as a "kinetic" can accomplish tasks internally, reporting, changing its status, and communicating with other objects without telling how. The statue conducts the operations (the override from the subclass). This is often referred to as the abstraction of data.

+ A virtual method is a method defined in the base class that the derived classes (subclasses) want to use must be redefined. Use the virtual (c ++) or abstract (java) keyword to declare the virtual method:

* + Virtual Classes may have virtual or not, but virtual methods must be in the virtual class
  + The virtual method does not contain the body
  + When inheriting a virtual class, it is imperative to rewrite its virtual method

**+ Example:**

+ When have a method abstract in class and we want change type of class become abstract we will must add keyword abstract before class

+ method abstract is not body

+ Below is class abstract employee with method abstract is SayHelloAbstract

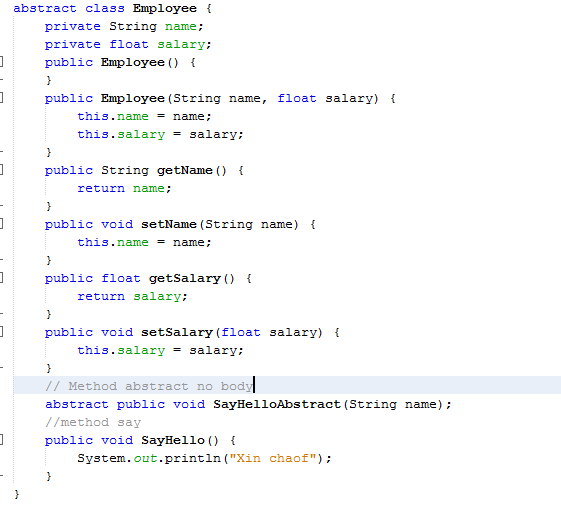


Figure 3. Class abstract

**b2.2: Encapsulation and information hiding**

+ There are four types of Access Modifiers: private, default, protected, and public.

* + Default: Access within the package
  + Private: Access within class
  + Public: Public component, free access from outside
  + Protected: Protected component, restricted from external access

+ A method in Java is a set of commands that are grouped together to perform an action.

+ Encapsulation is data packages (data, variables, status) and program code (code, method) into a class called class for easy management. In this case the data is often very confusing, not convenient for people not responsible for direct access, so often we will hide the data away, only to go out method

**b2.3: Inheritance**

**+ Concept:** Inheritance is the way to create new classes from predefined classes

+ The parent class can share data and methods for subclasses, subclasses must redefine the common logic, make the program short. If the parent class is an interface, then the child class inherits abstract contracts from the superclass

+ Suppose we have two classes A and B:

- Class A: base class (parent class) - supper class

- Class B: derived class (subclass) - sub class

+ Class B can use most of the methods and functions (variables) of class A except for methods and private property functions.

**+ Attention:**

- A parent class can have many subclasses

- In turn, each child class can have other children

- In C ++, multiple inheritance (one subclass can receive more than one superclass)

- Java only allows each subclass to inherit 1 and only one superclass / implementation keyword will replace the multiple inheritance from C ++.

**+ Example:**

The Employee class has the name and salary properties

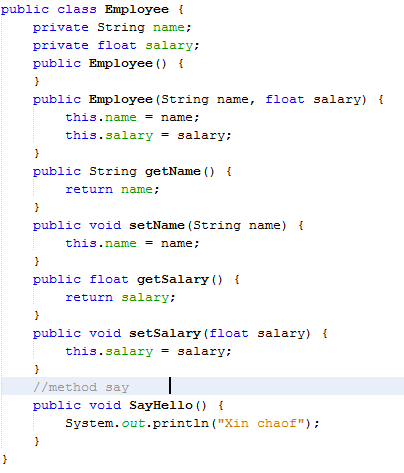


Figure 4. Class super Employee

Class Programmer have a new field this bonus and extends from class Employee

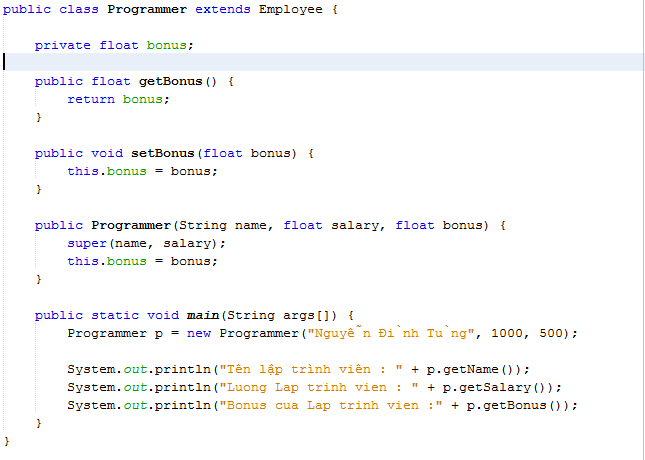


Figure 5. Class sub Programmer

**+ Explain:** Class Programmer I do not declare name and salary variables but it is still used. Because Programmer inherits the Employee class, it can reuse constructors and get / set from the parent class. Call them "super".

**b2.4: Polymorphism**

Polymorphism is expressed through rewriting methods (functions) from the parent class through class inheriting it or implementing interfaces.

* **Conclusions:** This task I explained about what is object oriented programming(OOP)?, benefits of OOP, what is object and class?, differences between object and class, for example, the characteristics and feature of OOP and for example for each feature. This task help understands more OOP in java.

# **Task 2: Identify the objects and data and file structures required to implement a given design and Design an object orientated programming solution for a given problem.**

## Use-case diagram for Software management students and lecturers of FPT Academy

+ Use case diagram is a diagram to show how users in the system can interact with the system with what features. It will cover the main parts:

- Types of User in the System (Actors)

- Features used in the system (Use cases)

- Relationship between Actors and Use cases (Associations)

- The scope of your system (System boundary boxes-optional)

+ Diagrams are seen as a logical and understandable way of interpreting the document for both the Developer, the testers, and the Business side. BA teams often create these diagrams to show the relationship between features and flow logic to implement a feature. Because to design a software management students and lecturers of FPT Academy I have a use-case diagram for system below:

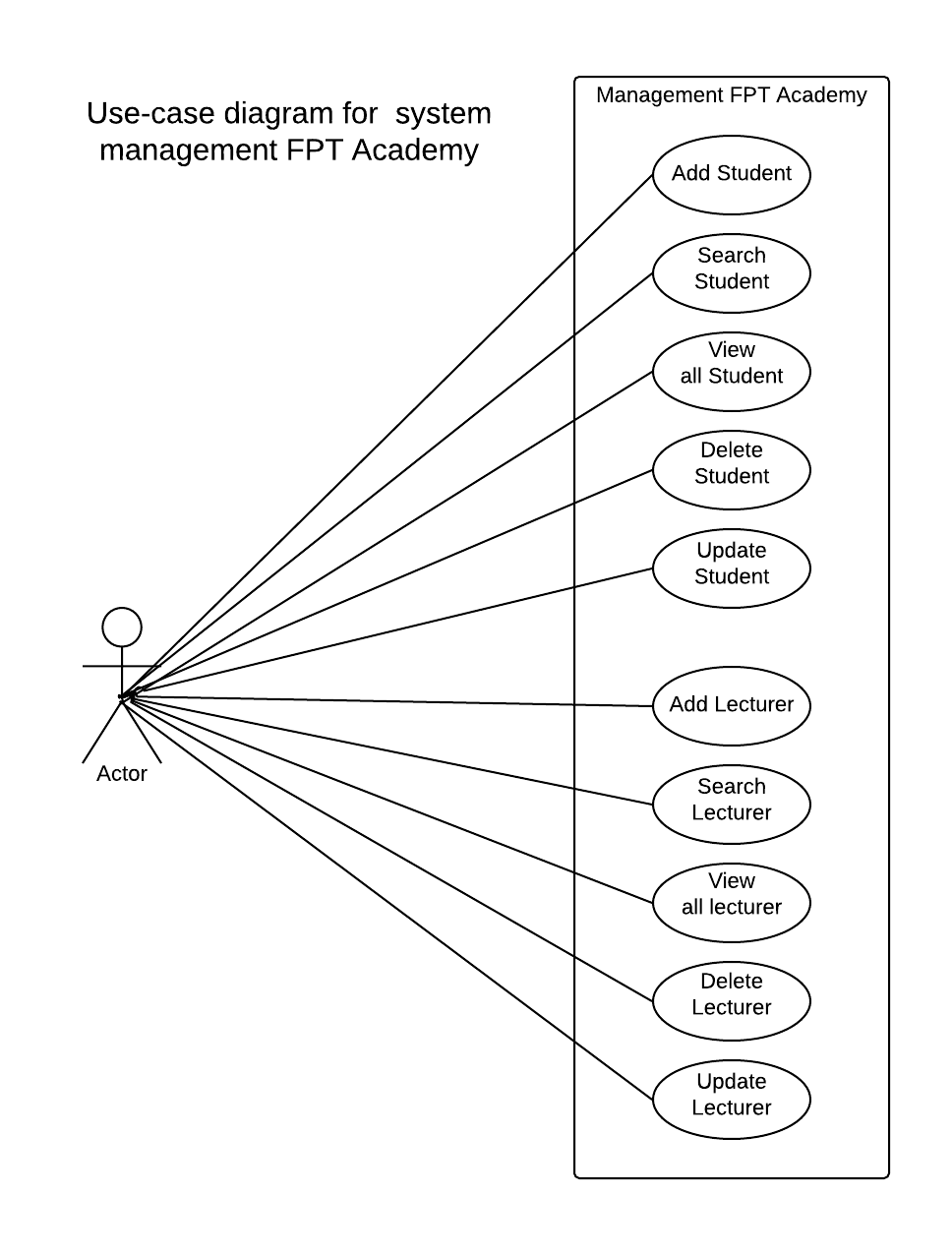


Figure 6.Use-case diagram for Management FPT Academy

+ You can see Manager can management students and lecturers, they can add new student, view all students, search, delete, update student same with lecturers management.

## Class diagram

+ Class Diagram is one of the most important drawings of software design, it shows the structure and relationship between the components that make up the software. During the construction of the Class Diagram we will have to decide a lot of design elements so it is the most difficult to draw. This drawing will show the static structure of the software.

+ I have design a class diagram to support you understand about system management students and lecture of FPT Academy below:

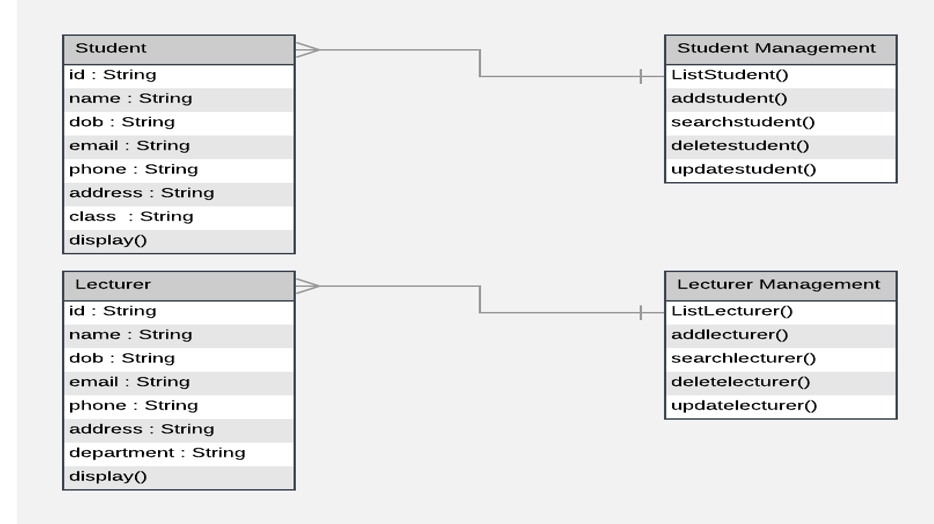


Figure 7. Class diagram for Management FPT Academy

+ Through the class diagram you can see we have four class this is Student, Lecturer, Student Management, Lecturer Management. Class student includes fields String id, String name, String dob, String email, String phone, String address and String class same with class student, class lecturer also have fields this except field String class replace String department. Class Student Management include list student and methods as addstudent(), searchstudent(), deletestudent(), updatestudent() same with class Student Management, class Lecturer Management also this. More we can see one Student Management can management multi student and one Lecture Management can management multi lecturer.

## 1.3 Flowchart

+ A flowchart is a flowchart of steps, and conditions are arranged in a certain order to solve a problem. As a programmer, developers often use it to represent the logic that they want to code before they start working. To say it is the easiest kind of chart for Developer.

+ With the diagrams, it is easy for QA

- Communicate with Developer through logic.

- There is a common language to discuss in the team.

- Easily define the Statement branch coverage for your test case.

+ To prepare for design software management students and lecturers of FPT Academy, I drew some important flowchart, diagram of management lecturer same with flowchart of management student. Below is some diagram about management student.

Flowchart of search ID for student with data text saved:

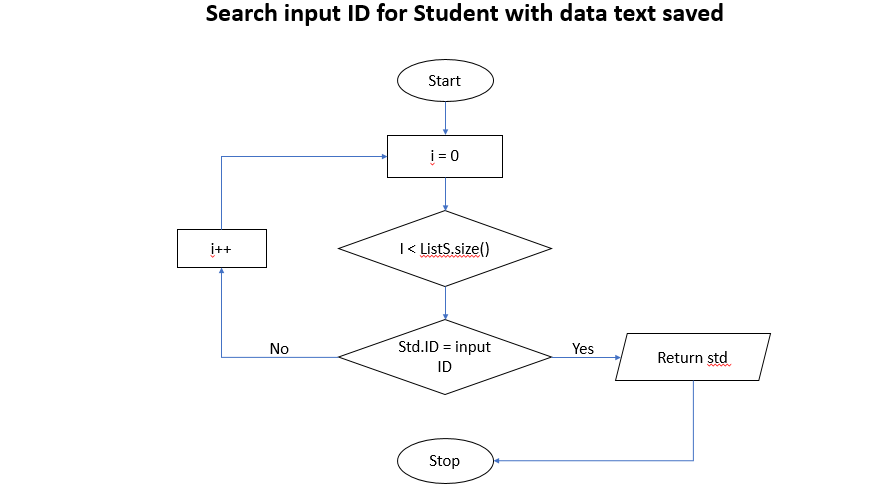


Figure 8. Search ID with data text saved

Flowchart of search name for student with data text saved:

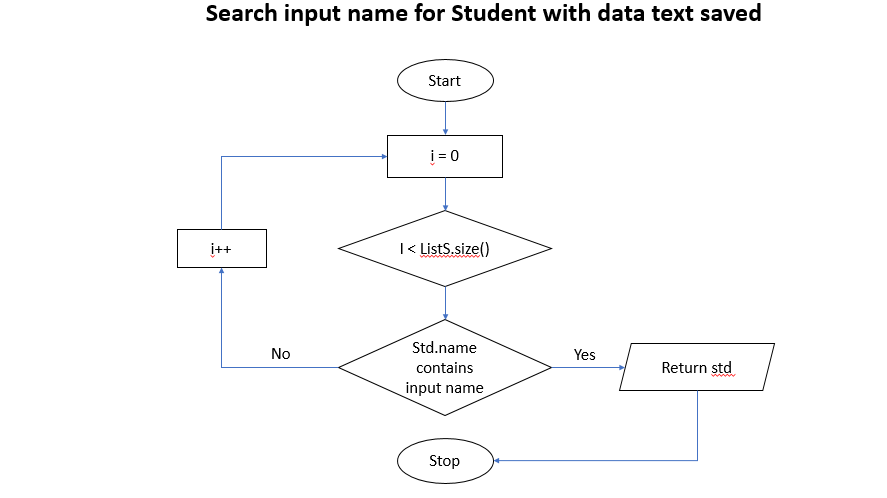


Figure 9. Search Name with data text saved

Flowchart for add a student:

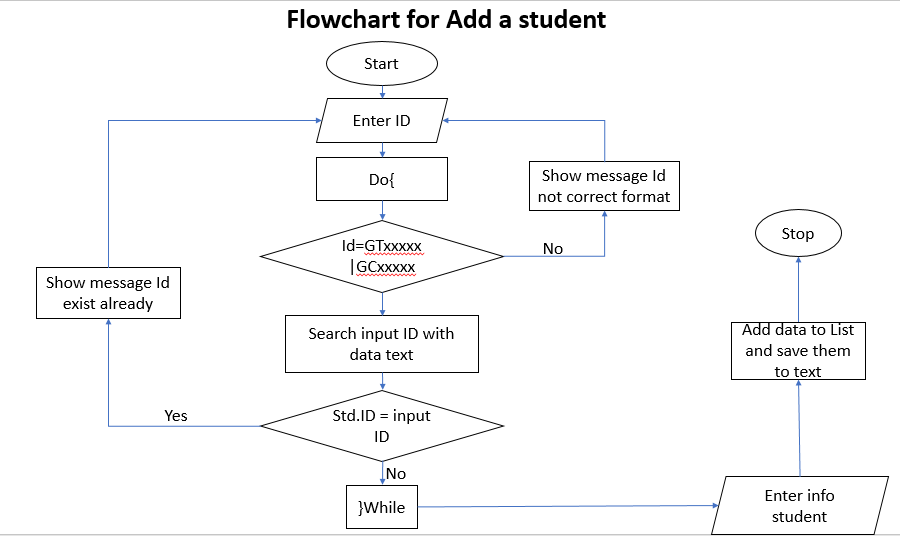


Figure 10. flowchart for add a student

Flowchart for update student:

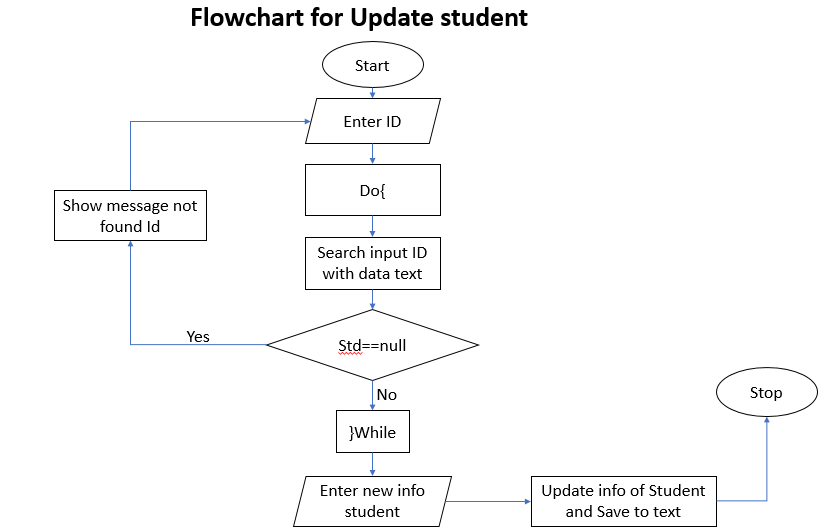


Figure 11. Flowchart for update student

Flowchart for delete student:

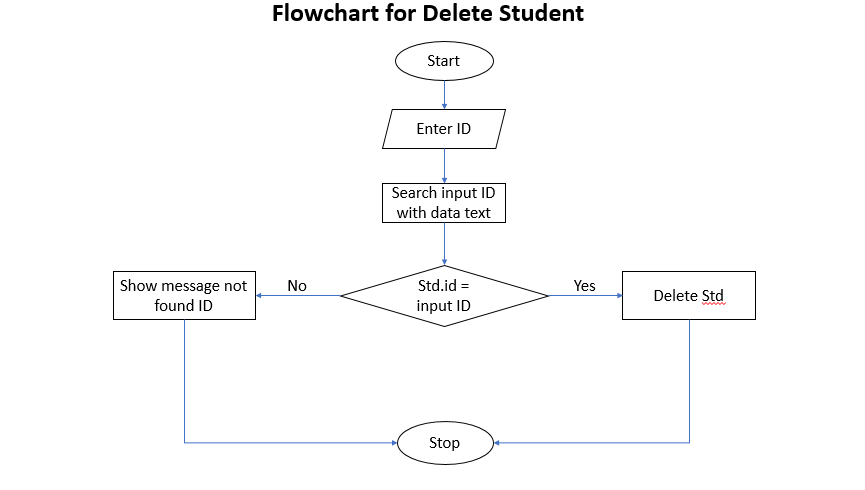


Figure 12. Flowchart for delete student

Flowchart for search student:

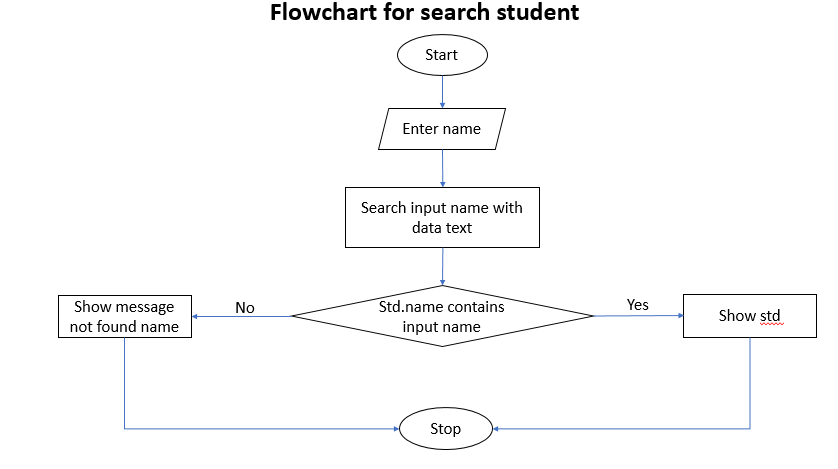


Figure 13. Flowchart for search student

* **Conclusions:** This task I focus to problems prepare for design a software management students and lecturers of FPT Academy as use-case diagram for software, class diagram and flowchart for method important in software. Explain about use-case diagram, class diagram.

# **References**

[1] Title: What does Object-Oriented Programming(OOP) mean?, accessed date: 19 April 2018, <https://www.techopedia.com/definition/3235/object-oriented-programming-oop>