HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY



**PROJECT REPORT**

**Demonstration of types of viruses and its mechanism**

**Group 07:**

Nguyễn Bá Dương - 20214886

Nguyễn Hải Dương - 20214887

Nguyễn Mạnh Dương - 20210243

Trần Hoàng Dương - 20214888

**Object-Oriented Programming**

**Instructor: Ph.D Nguyễn Thị Thu Trang**

# 1. Assignment of members

Nguyễn Bá Dương - 20214886:

* UML use case diagram and general class diagram.
* Collect viruses’ data.
* Virus abstract class and subclasses (VirusWithEnvelope, VirusWithoutEnvelope, and 5 families of virus).
* Virus structure classes (AcidNucleic, Capsid, Envelope, and Glycoprotein).
* VirusManager class.
* Virus mechanism screen (screen, controller, and fxml).
* UML detailed class diagrams
* Write report. (50%)
* Make powerpoint presentation.

Nguyễn Hải Dương - 20214887:

Nguyễn Mạnh Dương - 20210243:

* MainApp class.
* HomeScreen class (screen, controller).
* Main screen (screen, controller, and fxml).
* About screen (screen, controller, and fxml).
* Help screen (screen, controller, and fxml).
* Virus structure screen (screen, controller, and fxml).
* Refactor code.
* Write report. (50%)
* Make demo video.

Trần Hoàng Dương - 20214888:

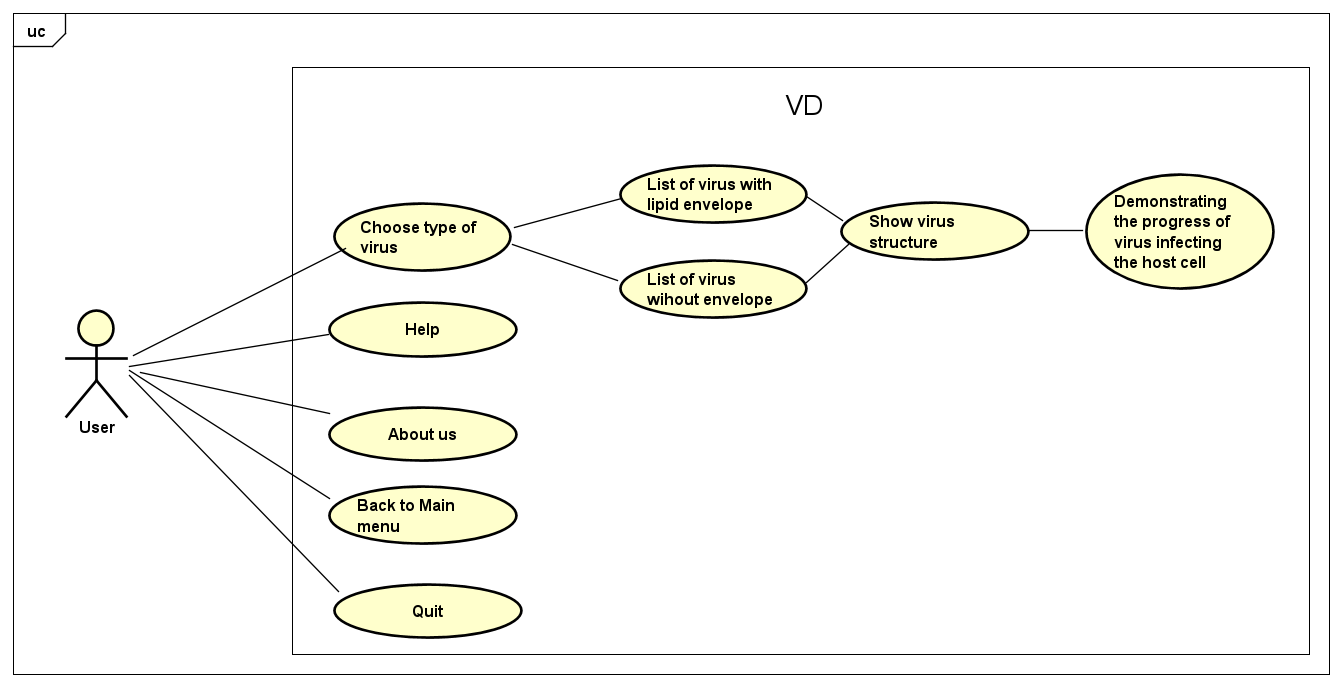
# 2. Mini-project description

## 2.1. Introduction

In a world where viruses continuously threaten global health, it has become more crucial than ever to comprehend their diverse forms and the methods they employ to infect host cells.

Project "VirusDemonstration", an interactive and educational application designed to provide users with fundamental knowledge about various viruses such as their structure and their infection mechanisms. Using an Object-Oriented Programming (OOP) approach, this application offers a seamless and intuitive experience, making it accessible to users of all backgrounds.

2.2. Use case diagram explanation



* In the main menu, users have the option to select a type of virus. Once they choose a virus type, a list of viruses belonging to that type is displayed. Upon selecting a specific virus from the list, the structure screen appears, providing detailed information about the virus's structure and characteristics. On the structure screen, there is a "Mechanism" button, which allows users to navigate to the mechanism demonstration screen. This screen presents a video illustrating the virus's infection process along with a summary of the progression of the infection.
* Each screen in the application includes a menu bar featuring three essential buttons. The "Help" button provides access to guidelines and instructions for the users. The "Virus Menu" button allows users to return to the main menu at any time. Additionally, clicking on the "About" button allows users to view information about the project.

# 3. Design

## 3.1. General class diagram

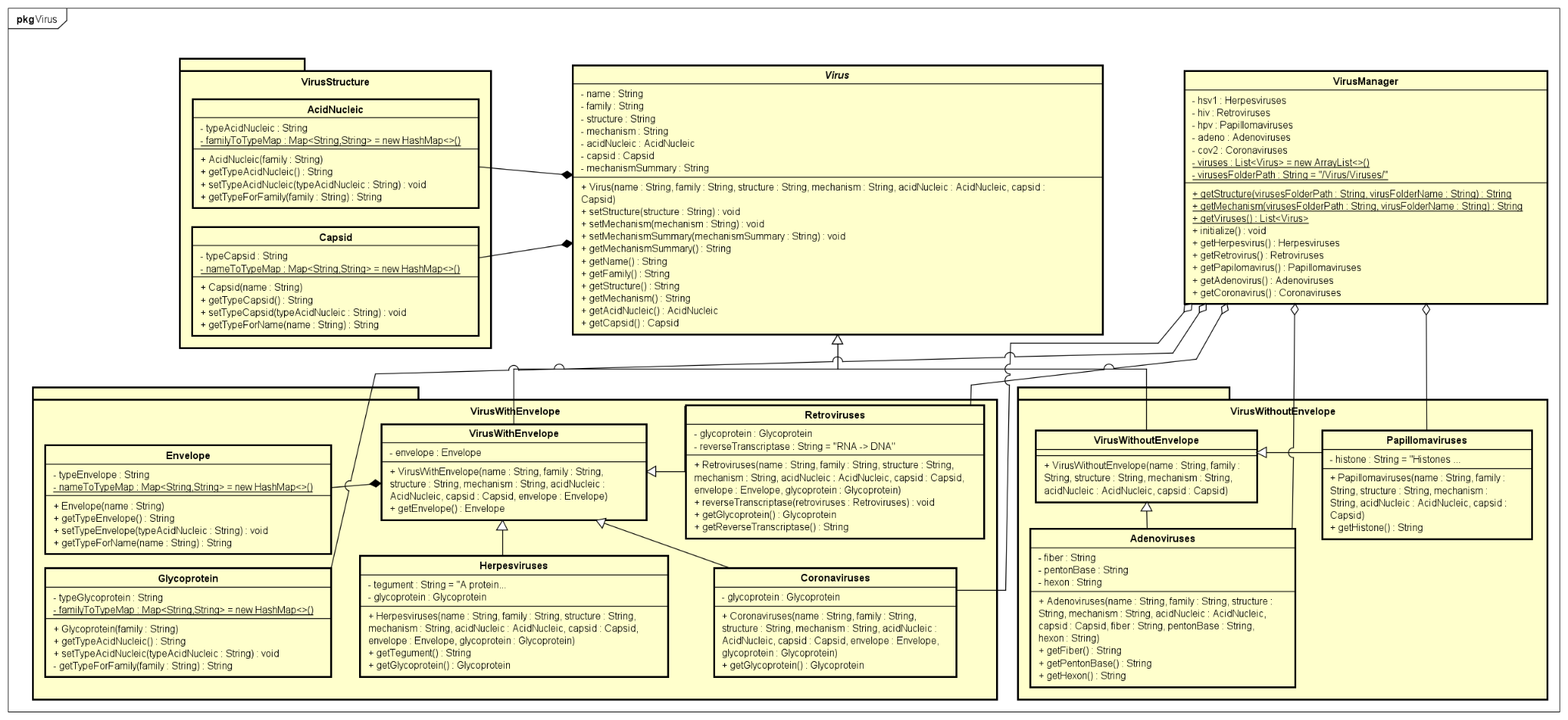
**Virus package**:

* Abstract class Virus.
* The VirusStructure subpackage consists of two essential classes, namely AcidNucleic and Capsid, both of which are present in every virus.
* There are two virus-subpackages of Virus:
  + VirusWithEnvelope: There is a class called VirusWithEnvelope, which is a derived class from the abstract Virus class. This subclass includes extra features like Envelope and Glycoprotein. Additionally, several virus families that possess an envelope are all derived from the VirusWithEnvelope class.
  + VirusWithoutEnvelope: There is a class called VirusWithoutEnvelope, which is a derived class from the abstract Virus class. There are several virus families that are all derived from the VirusWithoutEnvelope class.
* VirusManager class: To initialize all the viruses to be used later.

**Screen package:**

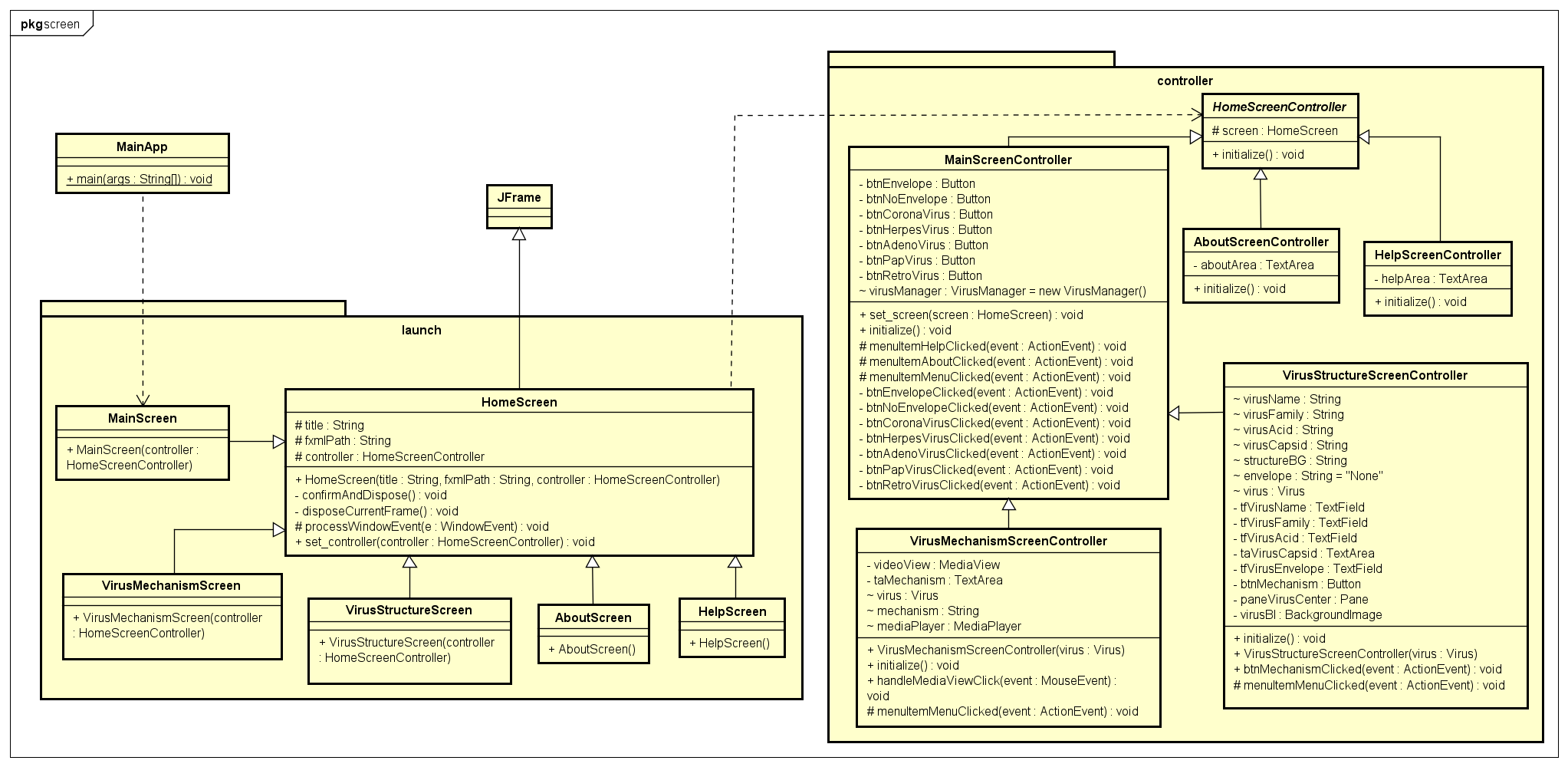
* MainApp class has the main method to run the application.
* There are two subpackages of screen:
  + launch: There are several classes to initialize screens. Out of those classes, there is a class HomeScreen which is the parent of the remaining classes.
  + controller: Controller classes of all screens. In this package, there is an abstract class named HomeScreenController which is used as an instruction for all the controller classes. HelpScreenController and AboutScreenController are similar, inherited directly from HomeScreenController. MainScreenController is a bit different which is the parent class for VirusStructureScreenController, VirusMechanismScreenController.

## 3.2. Detailed class diagram

**Package :** Virus

*Some important analysis of OOP in our code implementation:*

* The **abstract** Virus class **encapsulates** privatecommon data of viruses. It enables **inheritance** by subclasses to extend and specialize the Virus class. There are many methods in the Virus class, which allow us to preserve the **polymorphism** property. The class comprises other complex types like AcidNucleic and Capsid rather than directly containing their data.
* VirusWithEnvelope and VirusWithoutEnvelope are inherited from the Virus class. This allows reuse and specialization of structure.
* Each subclass of VirusWithEnvelope and VirusWithoutEnvelope represents a specific family of viruses, with unique attributes that describe their individual characteristics and behaviors.
* The VirusManager class contains an initialize() method responsible for setting up all the viruses to be utilized in the graphical user interface (GUI).

**Package** : screen

*Some important analysis of OOP in our code implementation:*

* All the screens are wrapped into JFrame then should follow a standard class as a parent. Thus, HomeScreen was set up to be the parent class for all the remaining classes in package *screen.launch*. The scene appeared is decided by fxml file and controller, which leads to the *constructor HomeScreen(String title, String fxmlPath, HomeScreenController controller)* to preserve **inheritance**.
* HomeSceenController is an abstract class which is set up as a representative for other controllers. We override a reasonable method for each controller. This setting lets us implement **polymorphism** into the code and make the implementation simpler.
* In the package screen.controller MainScreenController is set up with MenuBar and its interaction. It is straightforward to let all the classes which we want to include the menu bar **inherit** from the MainScreenController. In this project, VirusStructureScreenController and VirusMechanismScreenController are those classes.
* Importantly, in order to dispose of the old screen whenever we click a new button, we set a method for the controller: *.set\_screen(HomeScreen screen)* which allows us to pass screen reference as an attribute in controller class. By this method, we will be able to dispose screen through the Controller. Using the abstract class *HomeScreen* as a parameter allows us to preserve the **polymorphism** property of Object Oriented Programming technique.