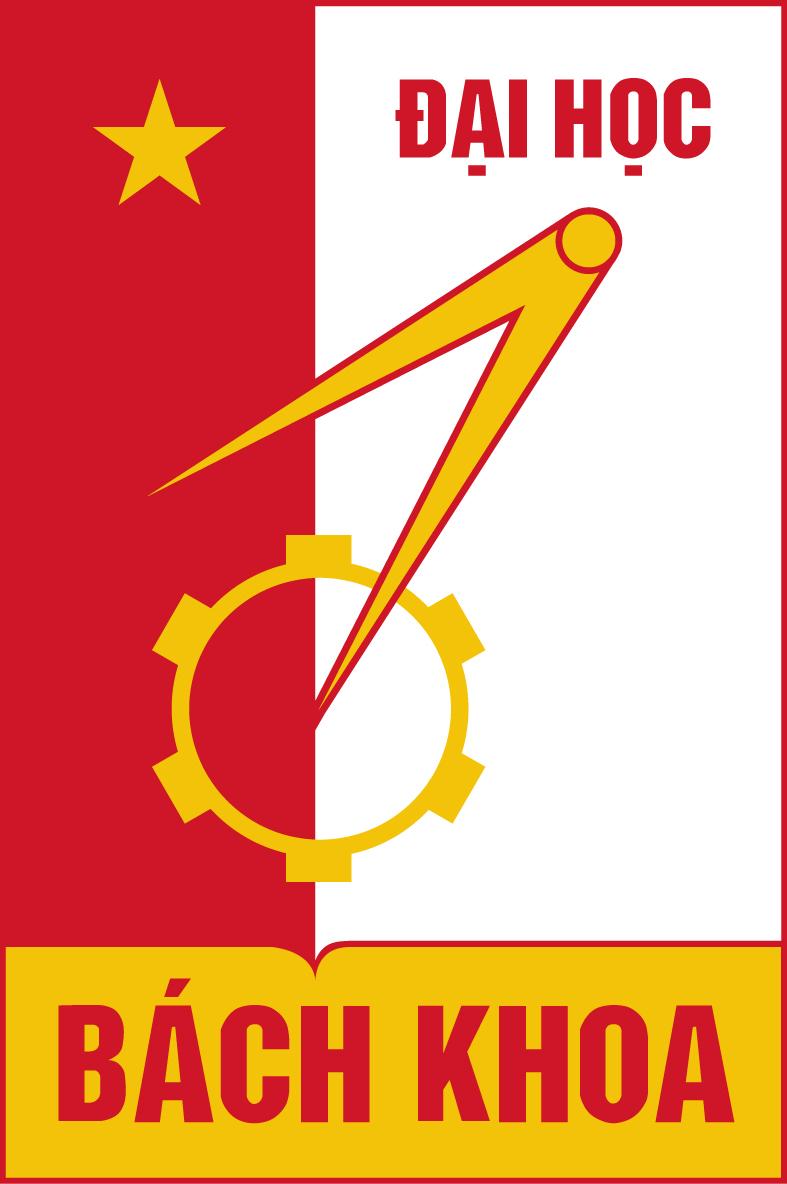
*Hanoi University of Science and Technology*

*School of Information and Communication Technology*

****

**Virus Information Application**

*Object Oriented Programming - IT3100E*

Group 10

*Hoang Trung Hieu 20226039*

*Ngo Minh Duc 20226028*

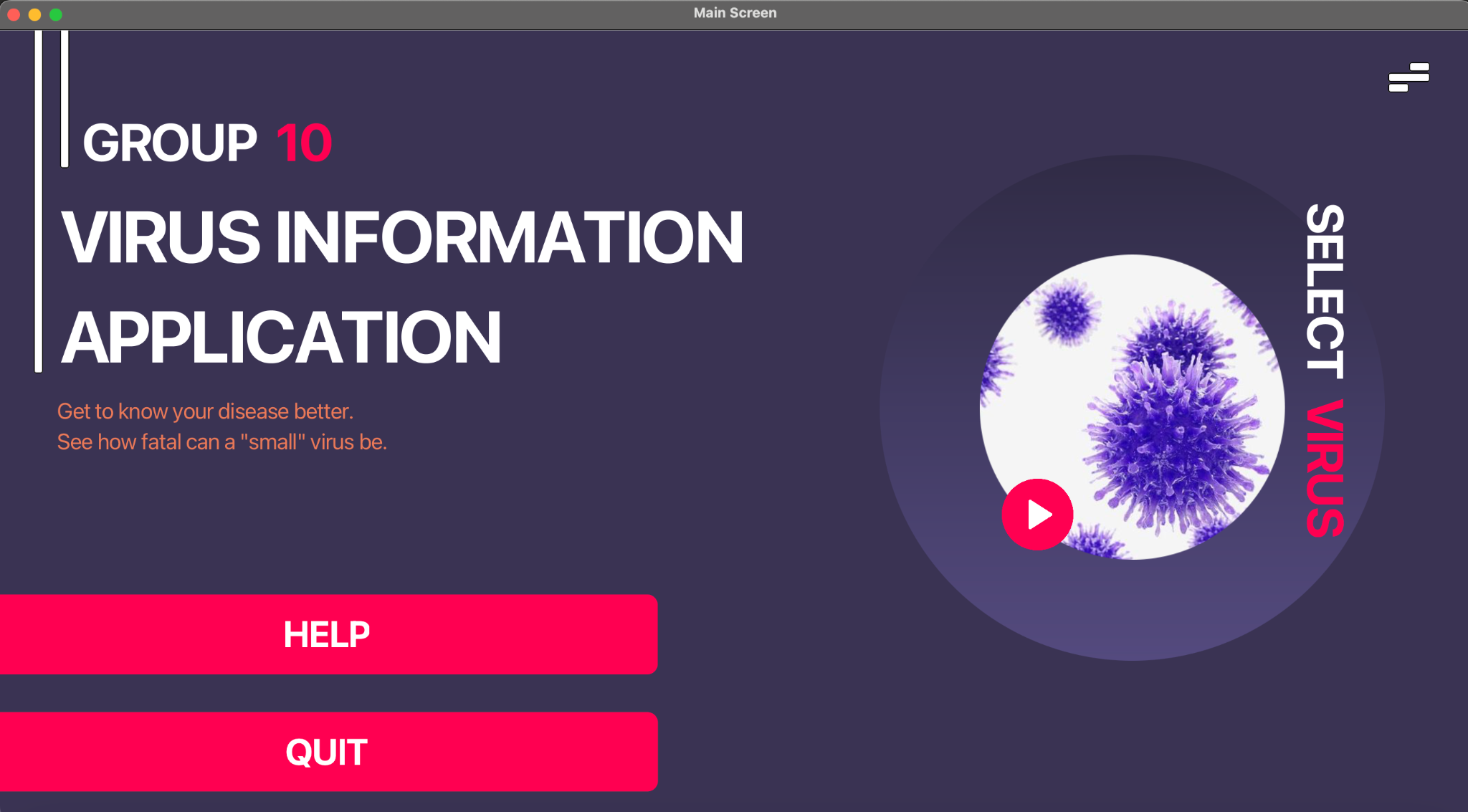
*Tran Tung Duong 20226033*

*Vu Nguyen Hao 20226037*

Table of contents

1. **Introduction………………………………………………………………**
2. **Design……………………………………………………………………...**
3. **Implementation Details…………………………………………………..**
4. **Object-Oriented Techniques……………………………………………..**
5. **Introduction**
6. *Project description*

A large number of health-related issues in the world have the involvement of viruses and they have caused great losses to humanity.

In this project, our application aims to provide valuable information about some of the most common viruses for users, including their basic information, structures, and infection processes.

*Figure 1. Virus Information Interface.*

1. *Assignment of members*

| Member name | Student ID | Responsibility |
| --- | --- | --- |
| Hoang Trung Hieu | 20226039 | Researching virus details, designing diagrams, application logic processing |
| Vu Nguyen Hao | 20226037 | Designing User Interface |
| Ngo Minh Duc | 20226028 | Researching virus details, designing diagrams, application logic processing |
| Tran Tung Duong | 20226033 | Designing User Interface, writing documents |

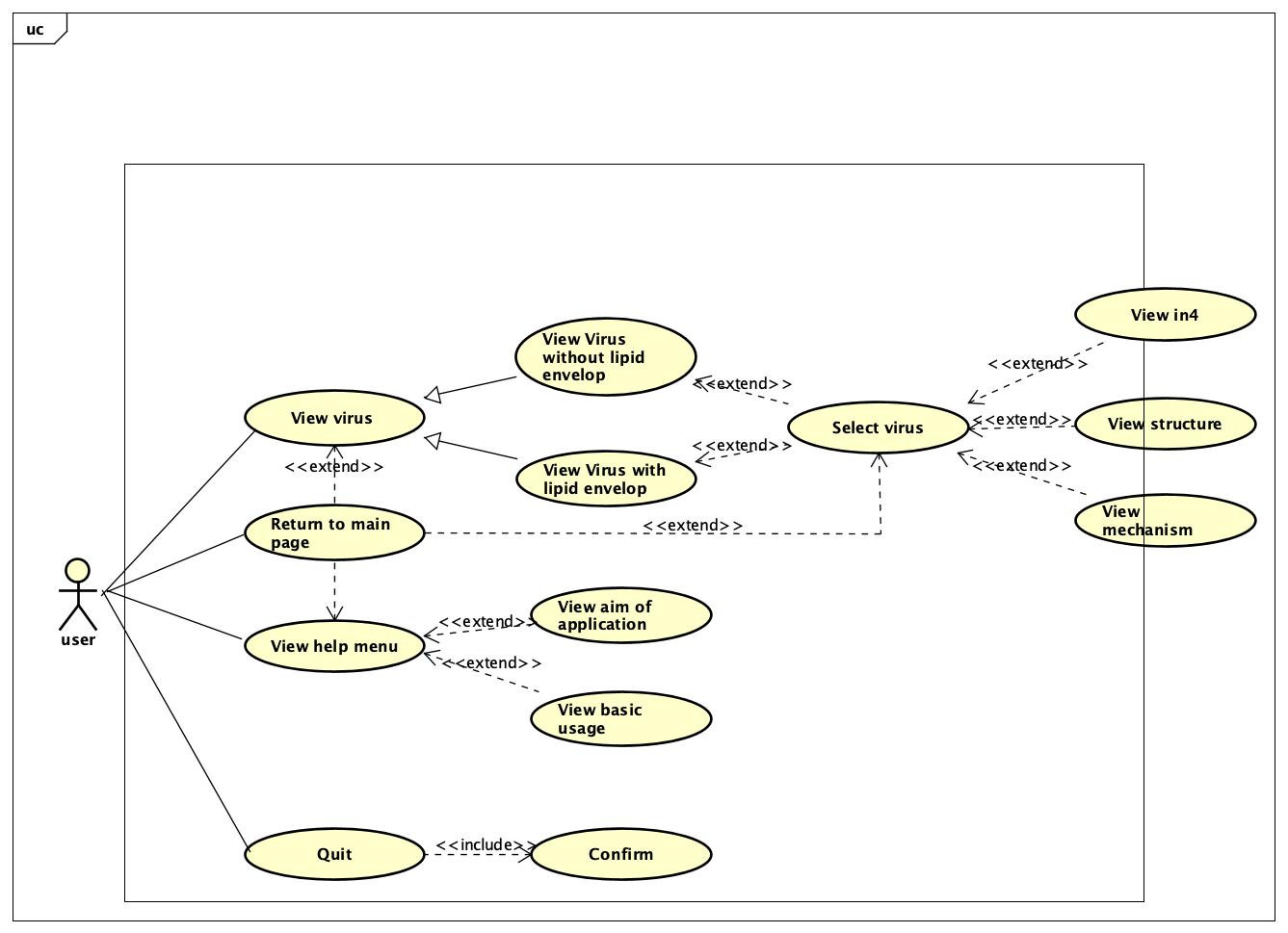
Every member contributes the same amount of work in this project

1. *Application requirements*

This project requires some platforms and toolkits in order to set up and run the simulator:

* Environment: Java Development Kit (JDK), JavaFX Runtime, IDE (Eclipse is recommended)
* Library: JavaFX

1. **Design**
2. Use case diagrams



*Figure 2. Use case diagram.*

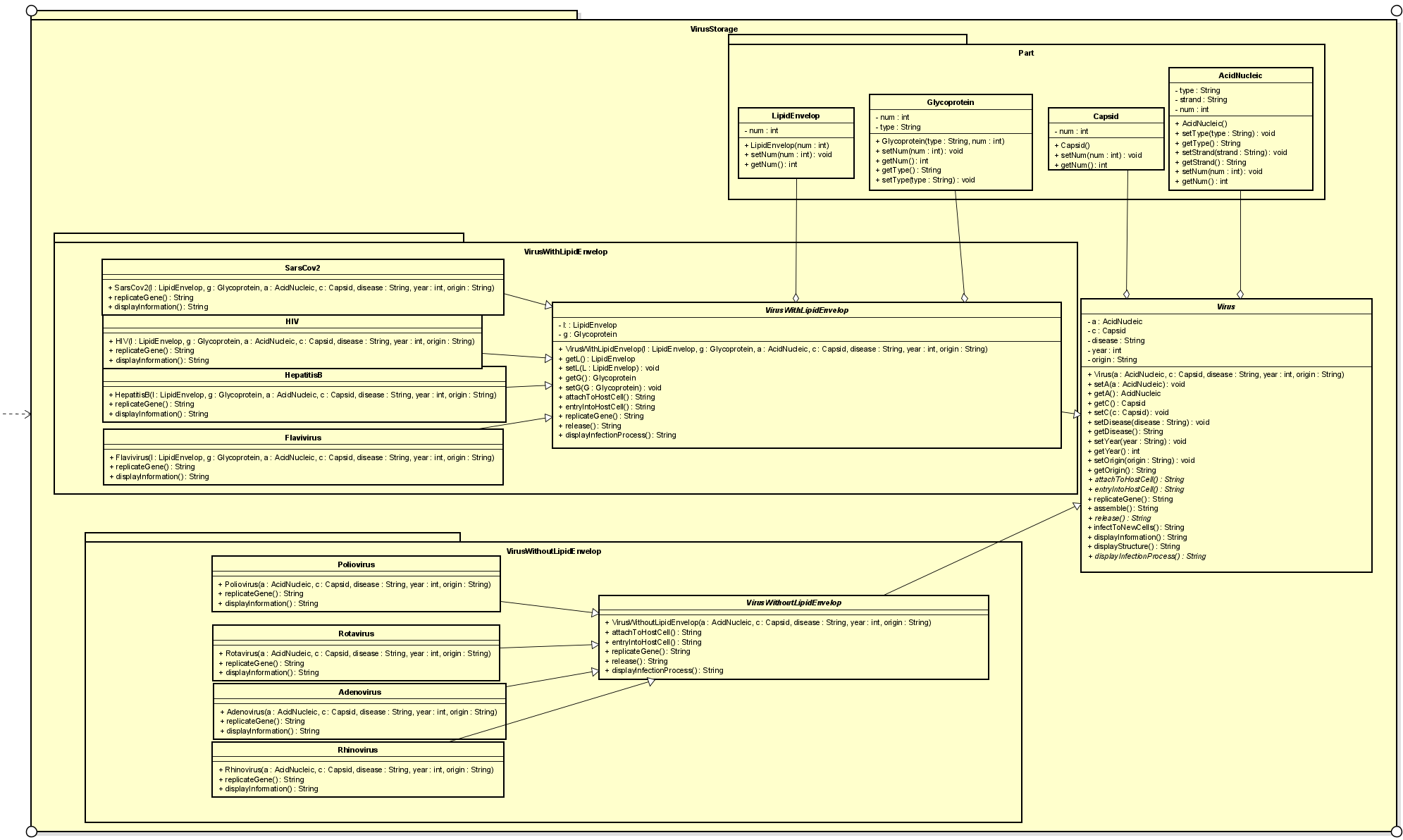
In detail:

* On the main screen: users have options to choose to view virus, view help menu, or quit.
* When users choose to view virus, there are options to choose between virus with lipid envelope and virus without lipid envelope.
* After choosing the desired type, the application will show 4 types of viruses for users to select.
* With each type of virus, the app displays:
* The basic information of the virus
* The structure of the virus
* The infection process of the virus
* The help menu shows the basic usage and aim of the application
* The quit button exits the application. The app also asks for confirmation before exiting.
* There is always a return button for users to get back to the main menu at any time.

1. Class diagrams

Our project can be divided into 2 big packages: *VirusStorage* and *GUIVirusFX*

1. *VirusStorage* package



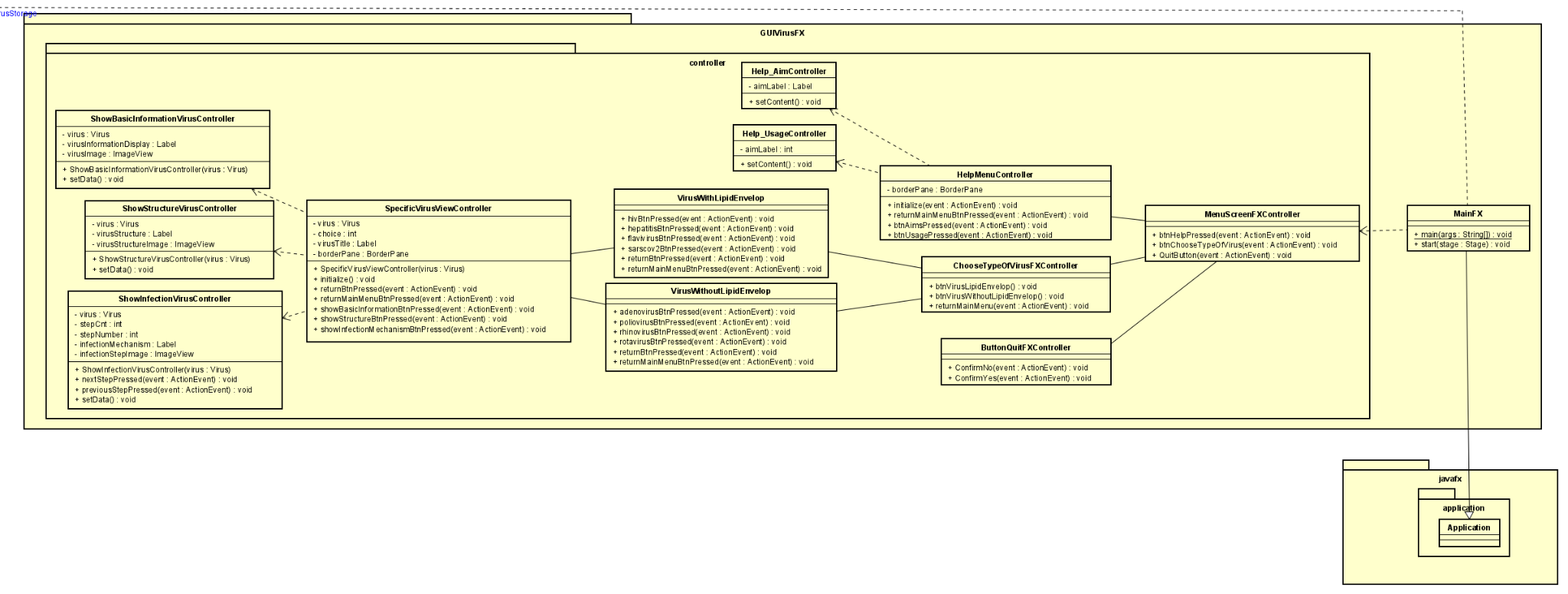
*Figure 3. VirusStorage package diagram.*

In detail:

| Packages | Classes | Contents |
| --- | --- | --- |
| *Part* | *AcidNucleic* | This package stores different parts of a virus. Every virus has AcidNucleic and Capsid, but only a virus of type VirusWithLipidEnvelop has Glycoprotein and LipidEnvelop |
| *Capsid* |
| *Glycoprotein* |
| *LipidEnvelop* |
|  | *Virus* | Has 3 important methods:   * displayInformation() * displayStructure() * displayInfectionProcess()   The infection process contains 6 steps:   * attachToHostCell() * entryIntoHostcell() * replicateGene() * assemble() * release() * infectToNewCells()   Besides, there are constructors, getters, and setters |
| *VirusWithLipidEnvelop* | *VirusWithLipidEnvelop* | Inherit from *Virus,* implement attachToHostCell(), entryIntoHostcell(), replicateGene(), release(), and displayInformation() |
| *HIV* | Inherit from *VirusWithLipidEnvelop*, implement replicateGene() and displayInformation() |
| *SarsCov2* |
| *HepatitisB* |
| *Flavivirus* |
| *VirusWithoutLipidEnvelop* | *VirusWithoutLipidEnvelop* | Inherit from *Virus,* implement attachToHostCell(), entryIntoHostcell(), replicateGene(), release(), and displayInformation() |
| *Rotavirus* | Inherit from *VirusWithoutLipidEnvelop*, implement replicateGene() and displayInformation() |
| *Rhinovirus* |
| *Adenovirus* |
| *Poliovirus* |

*Table 1. Packages and classes of VirusStorage package.*

1. *GUIVirusFX*



*Figure 4 . GUIVirusFX package diagram.*

In detail:

| Packages | Classes | Contents |
| --- | --- | --- |
|  | *MainFX* | Run application |
| *controller* | *MainScreenFXController* | The controller of the corresponding class |
| *HelpMenuController* |
| *Help\_AimController* |
| *Help\_UsageController* |
| *ChooseTypeOfVirusFXController* |
| *VirusWithLipidEnvelop* |
| *VirusWithoutLipidEnvelop* |
| *SpecificVirusViewController* |
| *ShowBasicInformationVirusController* |
| *ShowStructureVirusController* |
| *ShowInfectionVirusController* |

*Table 2. Packages and classes o VirusGUIFX package.*

1. **Implementation Details**
2. *main\_package.VirusStorage*

* Abstract class Virus:
* attribute *a*: is an instance of *AcidNucleic* class
* attribute *c*: is an instance of *Capsid* class
* attribute *disease*: String type, contains information about disease that the virus causes
* attribute *year*: int type, the year in which virus is discovered
* attribute *origin*: String type, location where the virus originated
* constructor *Virus()*
* method *getA()*: returns attribute *a*
* method *setA()*: set attribute *a*
* method *getC()*: get attribute *c*
* method *setC()*: set attribute *c*
* method *getDisease()*: get attribute *disease*
* method *setDisease()*: set attribute *disease*
* method *getYear()*: get attribute *year*
* method *setYear()*: set attribute *year*
* method *getOrigin()*: get attribute *origin*
* method *setOrigin()*: set attribute *origin*
* method *attachToHostCell():* abstract method, returns String type, shows information of how the virus attach to host cell
* method *entryIntoHostCell()*: abstract method, returns String type, shows information of how the virus entries into host cell
* method *replicateGene()*: returns String type, shows information of how the virus replicates its genome
* method *assemble()*: returns String type, shows information of how the virus assemble
* method *release()*: returns String type, shows information of how the virus gets out of host cell
* method *infectToNewCells()*: returns String type, shows information of how the virus continues the process of infecting new cells
* method *displayInformation()*: returns String type, shows basic information of the virus
* method *displayStructure()*: returns String type, shows the structure of the virus
* method *displayInfectionProcess()*: returns String type, shows the whole infection process of the virus

1. main\_package.VirusStorage.VirusWithLipidEnvelop

* Abstract class *VirusWithLipidEnvelop* inherits from *Virus*:
* attribute *l*: is an instance of *LipidEnvelop*
* attribute *g*: is an instance of *Glycoprotein*
* constructor *VirusWithLipidEnvelop()*
* method *getL()*: get attribute *l*
* method *setL()*: set attribute *l*
* method *getG()*: get attribute *G*
* method *setG()*: set attribute *G*
* method *attachToHostCell()*: returns String type, override method from *Virus*
* method *entryIntoHostCell()*: returns String type, override method form *Virus*
* method *replicateGene()*: returns String type, override method from *Virus*
* method *release()*: returns String type, override method from *Virus*
* method *displayInfectionProcess()*: returns String type, override method from *Virus*
* Class *SarsCov2* inherits from *VirusWithLipidEnvelop*
* constructor *SarsCov2()*
* method *replicateGene()*: returns String type, override method from parents’ classes
* method *displayInformation()*: returns String type, override method from *Virus*
* Class *HIV, HepatitisB, Flavivirus* has the same structures as *SarsCov2* except for their constructors

1. main\_package.VirusStorage.VirusWithoutLipidEnvelop

* Abstract class *VirusWithoutLipidEnvelop* inherits from *Virus*:
* constructor *VirusWithoutLipidEnvelop()*:
* method *attachToHostCell()*: returns String type, override method from *Virus*
* method *entryIntoHostCell()*: returns String type, override method from *Virus*
* method *replicateGene()*: returns String type, override method from *Virus*
* method *release()*: returns String type, override method from *Virus*
* method *displayInfectionProcess()*: returns String type, override method from *Virus*
* Class *Poliovirus*:
* constructor *Poliovirus()*
* method *replicateGene()*: returns String type, override method from parents’ classes
* method *displayInformation()*: returns String type, override method from parents’ classes
* Class *Rotavirus, Adenovirus, Rhinovirus* have the same structures as *Poliovirus* except for their constructors

1. main\_package.VirusStorage.Part

* Class *LipidEnvelop*:
* attribute *num*: int type
* constructor *LipidEnvelop()*
* method *setNum()*: set attribute *num*
* method *getNum()*: get attribute *num*
* Class *GlycoProtein*:
* attribute *num*: int type
* attribute *type*: String type
* constructor *GlycoProtein()*
* method *setNum()*: set attribute *num*
* method *getNum()*: get attribute *num*
* method *getType()*: get attribute *Type*
* method *setType()*: set attribute *Type*
* Class *Capsid:*
* attribute *num*: int type
* constructor *Capsid()*
* method *getNum()*: get attribute *num*
* method *setNum()*: set attribute *num*
* Class *AcidNucleic*:
* attribute *type*: String type
* attribute *strand*: String type
* attribute *num*: int type
* constructor *AcidNucleic()*
* method *setType()*: set attribute *type*
* method *getType()*: set attribute *type*
* method *getStrand()*: get attribute *strand*
* method *setStrand()*: set attribute *strand*
* method *getNum()*: get attribute *num*
* method *setNum()*: set attribute *num*

1. *main\_package.GUIVirusFX*

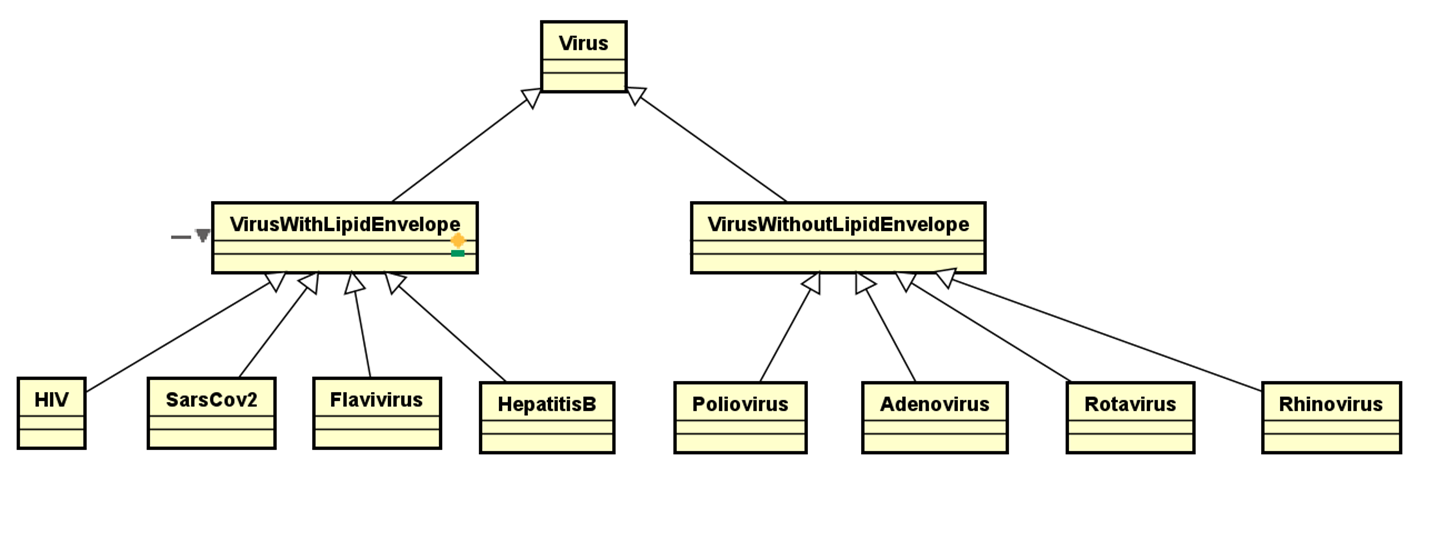
* Class *MainFX* inherits from *Application*:
* method *start():* Load MainScreenFX.fxml, set stage and scene.
* method *main()*: Run the application.

1. *main\_package.GUIVirusFX.controller*

* Class *ButtonQuitFXController*:
* method *ConfirmNo()*: Reload “*MainScreenFX.fxml*”, return to main menu scene.
* method *ConfirmYes()*: Quit the application.
* Class *ChooseTypeOfVirusFXController()*:
* method *btnVirusLipidEnvelop()*: Load Virus with lipid envelope choosing screen stage.
* method *btnVirusWithoutEnvelop()*: Load Virus without lipid envelope choosing screen stage.
* method *returnMainMenu()*: Load main menu stage.
* Class *Help\_AimController*:
* attribute *aimLabel*: Label type.
* method *setContent()*: set content (Aims of the application) of the label *aimLabel*.
* Class *Help\_UsageController*:
* attribute *aimLabel*: Label type
* method *setContent()*: set content (Usage of the application) of the label *aimLabel*.
* Class *HelpMenuController*:
* attribute *borderPane*: BorderPane type
* attribute *titleLabel*: Label type
* method *initialize()*: Load “*Help\_Aim.fxml*”, display Aim of the application beforehand.
* method *returnMainMenuBtnPressed()*: Load “*MainScreen.fxml*”, set main menu stage.
* method *btnAimsPressed()*: set controller class “*Help\_AimController*”, set Center of borderPane to a new *anchorPane* contains *aimLabel* => display Aim
* method *btnUsagePressed()*: set controller class “*Help\_UsageController*”, set Center of borderPane to a new *anchorPane* contains *aimLabel* => display Usage
* Class *MenuScreenFXController*:
* method *btnHelpPressed()*: Load Help Menu stage
* method *btnChooseTypeOfVirus()*: Load Choose Type of Virus Menu stage
* method *QuitButton()*: Load Quit Option menu stage
* Class *VirusWithLipidEnvelopeFXController*:
* method *hivBtnPressed()*: initialize a HIV object, transmit that object to “*SpecificVirusViewController*” class constructor. Start displaying the information display stage (of HIV).
* method *hepatitisbBtnPressed()*: initialize a *HepatitisB* object. The rest is the same as “*hivBtnPressed*”
* method *flavivirusBtnPressed ()*: initialize a *Flavivirus* object. The rest is the same as “*hivBtnPressed*”
* method *sarscov2BtnPressed()*: initialize a *SarsCov2* object. The rest is the same as “*hivBtnPressed*”
* method *returnBtnPressed()*: Load and set Choose Type of Virus stage.
* method *returnMainMenuBtnPressed()*: Load and set Main Menu screen stage.
* Class *VirusWithoutLipidEnvelopeFXController*:
* method *adenovirusBtnPressed()*: initialize a Adenovirus object, transmit that object to “*SpecificVirusViewController*” class constructor. Start displaying the information display stage (of Adenovirus).
* method *poliovirusBtnPressed()*: initialize a *Poliovirus* object. The rest is the same as “*adenovirusBtnPressed*”
* method *rhinovirusBtnPressed ()*: initialize a *Rhinovirus* object. The rest is the same as “*adenovirusBtnPressed*”
* method *rotavirusBtnPressed()*: initialize a *Rotavirus* object. The rest is the same as “*adenovirusBtnPressed*”
* method *returnBtnPressed()*: Load and set Choose Type of Virus stage.
* method *returnMainMenuBtnPressed()*: Load and set Main Menu screen stage.
* Class *ShowBasicInformationVirusController*:
* attribute *virus*: Virus type
* attribute *virusInformationDisplay*: Label type
* attribute *virusImage*: ImageView type
* constructor *ShowBasicInformationVirusController(Virus virus)*: set this.virus = virus
* method *setData()*: Display virus information and picture according to the chosen virus type
* Class *ShowStructureVirusController*:
* attribute *virus*: Virus type
* attribute *virusStructure*: Label type
* attribute *virusStructureImage*: ImageView type
* constructor *ShowStructureVirusController(Virus virus)*: set this.virus = virus
* method *setData()*: Display virus structure and picture according to the chosen virus type
* Class *ShowInfectionVirusController*:
* attribute *virus*: Virus type
* attribute *stepCnt*: int type =1
* attribute *stepNumber*: Label type
* attribute *infectionInformation*: Label type
* attribute *infectionStepImage*: ImageView type
* constructor *ShowInfectionVirusController(Virus virus)*: set this.virus = virus
* method *setData()*: set and display the first step information and image
* method *previousStepPressed()*: if *stepCnt* == 1 then nothing happen, else return to the previous step by changing the *infectionInformation* and *infectionStepImage*’s content
* method *nextStepPressed()*: if *stepCnt* == 6 then nothing happen, else move to the next step by changing the *infectionInformation* and *infectionStepImage*’s content
* Class *SpecificVirusViewController*:
* attribute *virus*: Virus type
* attribute *choice*: int type = 0
* attribute *virusTitle*: Label type
* attribute *borderPane*: BorderPane type
* attribute *virusName*: Label type
* attribute *contentVbox*: Vbox type
* constructor *SpecificVirusViewController(Virus virus)*: set this.virus=virus
* method *initialize()*: load the Show basic information scene beforehand, by setting Center of *borderPane* with the newly created *anchorPane*. Set *virusName* and *virusTitle* according to the previously selected virus
* method *returnBtnPressed()*: return to the choose virus menu by load and set the stage
* method *returnMainMenuBtnPressed()*: return to Main menu by load and set the main menu stage
* method *showBasicInformationBtnPressed()*: load the Show basic information scene, by setting Center of *borderPane* with the newly created *anchorPane*, which controlled by *ShowBasicInformationVirusController* class
* method *showStructureBtnPressed()*: load the Show basic Virus Structure scene, by setting Center of *borderPane* with the newly created *anchorPane,* which is controlled by *ShowStructureVirusController* class
* method *showInfectionMechanismBtnPressed()*: load the Show Infection Virus scene, by setting Center of *borderPane* with the newly created *anchorPane*, which is controlled by *ShowInfectionVirusController* class

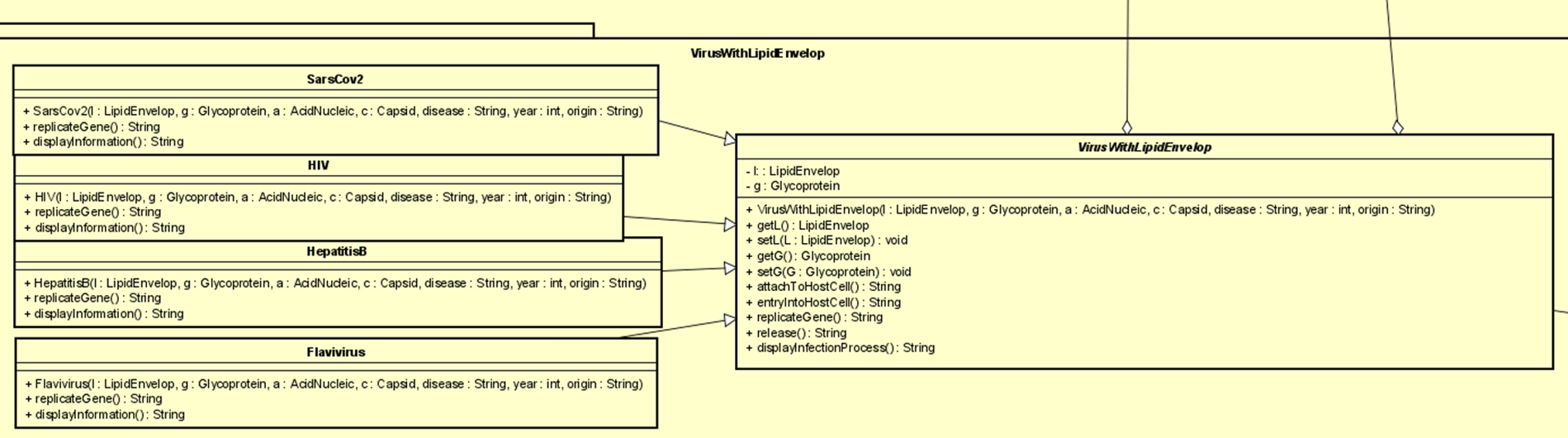
1. **Object-Oriented techniques**
2. *Inheritance*

* Abstract class *Virus*
* 2 abstract classes *VirusWithLipidEnvelope* and *VirusWithoutLipidEnvelope* inherit from the class Virus
* 8 classes of specific viruses inherit from 2 above classes

****

*Figure 5 . Demonstration of the inheritance design*

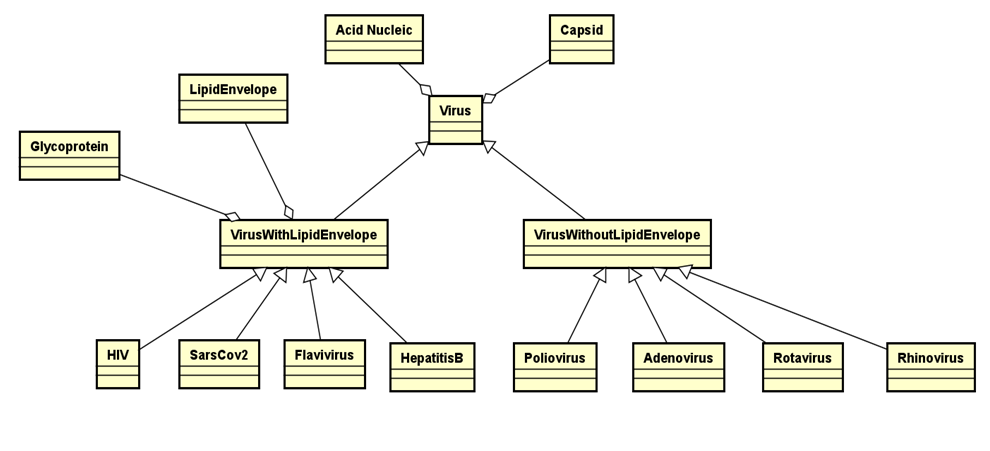
* Method overriding: The child class has a method with the same name and same signature of a method in its parent class



*Figure 6 . Demonstration of method overriding*

1. *Aggregation*

* A virus has some parts, including
* Acid Nucleic
* Capsid
* Lipid Envelope (for VirusWithLipidEnvelope)
* Glycoprotein (for VirusWithLipidEnvelope)



*Figure 7 . Demonstration of aggregation*

1. *Polymorphism*

* A single object can represent multiple different types (upcasting and downcasting)

E.g: HIV can be of HIV, VirusWithLipidEnvelope, and Virus data type