HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION COMMUNICATION AND TECHNOLOGY

-----□□&□□-----


# OBJECT-ORIENTED PROGRAMMING PROJECT REPORT

**Instructor: Nguyen Thi Thu Trang**
**Student names:**

| | |
|---|---|
| **Tran Huu Hien** | **20204966** |
| **Phan Huy Hiep** | **20210328** |
| **Pham Dinh Hai** | **20215043** |
| **Pham Cong Hao** | **20215045** |

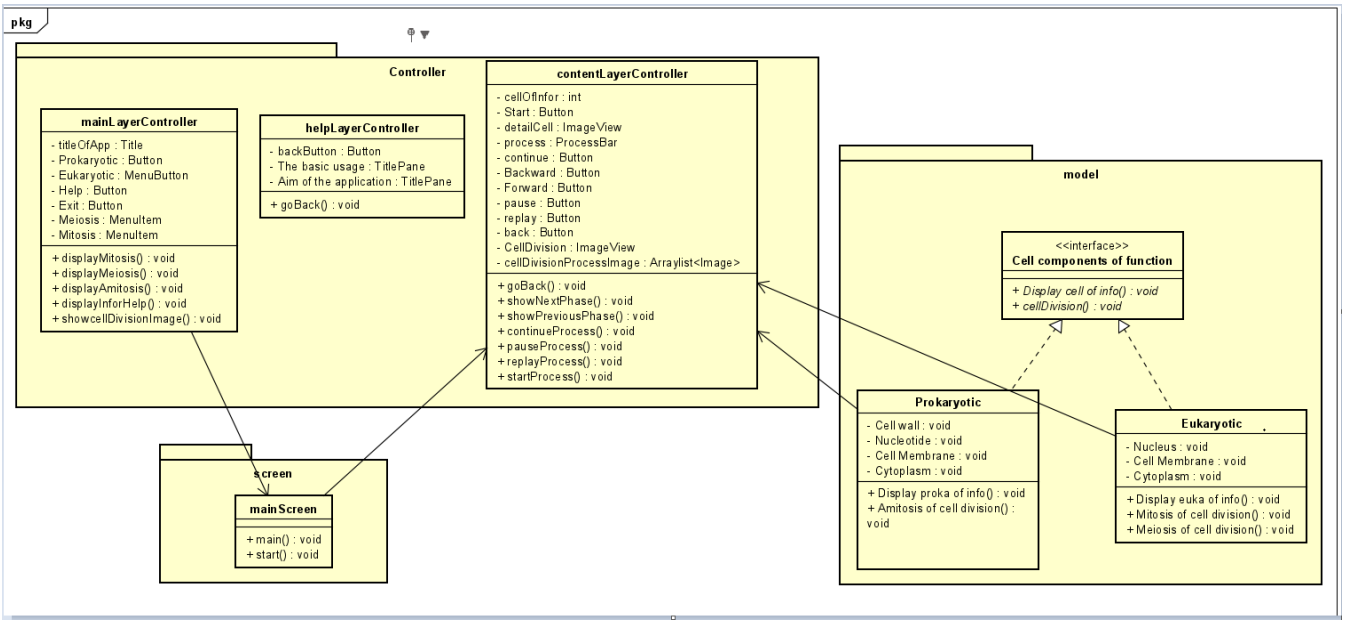
***Hanoi, January 2024***

# Table of contents

## I. CONTRIBUTION

| Student ID | Full name | Work | Contribution |
|---|---|---|---|
| 20204966 | Tran Huu Hien | - Design javafx , show data<br>- Class diagram<br>- Usecase diagram | |
| 20210328 | Phan Huy Hiep | - Show data , report<br>- Design javafx<br>- usecase diagram | |
| 20215043 | Pham Dinh Hai | - Slide, video<br>- Fixing<br>- class diagram | |
| 20215045 | Pham Cong Hao | - Word, video<br>- class diagram<br>- Fixing | |

## II. UML CLASS DIAGRAM




## III. BUILD & EXPLAINATION

In our program, there are two main parts including saving and exporting images using JavaFX. So that, our source code consists of for main packages: ***controller, image, model and screen***.

1. In **controller** package, there are 3 classes:
   + The **contentLayerController** class is responsible for displaying and controlling the progress of a process, perhaps the cell splitting process.
   + The **helpLayerController** class is responsible for controlling the interface and handling events when the user performs an action such as pressing the "BackButton" button to return to the main page.
   + The **mainLayerController** is responsible for controlling the interface and handling events when the user performs actions such as displaying the cell division process or exiting the application.

2. In **image** package
   + Stores images of cells during each stage of separation
3. In **screen** package.
   We need the image package to save the images we want to display using JavaFX.
   + The **content.fxml** file retrieves images stored in the package and displays them on the screen.

+ The **helpLayer.fxml** file defines the user interface for a screen that helps guide the use of the application or provides information about the application's goals.

Besides, we also used another package that is **vemanhinh** to printf members information on the JavaFX interface.

## IV. OBJECT-ORIENTED PROGRAMMING APPLICATION

### 1. Encapsulation

Encapsulation is used to **hide the internal data of an object**. Besides, it blocks direct access to the object's internal elements. We can take advantage of Encapsulation when we want to protect the data inside of the object.

In our project:

```
24  public class contentLayerController {
25
26      Timer timer = new Timer();
27      private boolean started = false;
28      @FXML
29      private Button BackButton;
30
31      @FXML
32      private ImageView detailCell;
33
34      @FXML
35      private ImageView CellDivision;
36
37      @FXML
38      private Button nextbutton;
39
40      @FXML
41      private Button previousButton;
42
43      @FXML
44      private VBox Vbox;
```

Applying encapsulation, private data is only used in **contentLayerController()** function, outside classes cannot access this data. Thereby ensuring that the data is used for the right purpose.

### 2. Inheritance

+ Inheritance can be defined as the process in which a class (class) has get the properties of another class. Those properties can be a method

or a school. The inherited class will be called the parent class and the derived class will be called
is a child class.

+ Inheritance increases reusability. When a class inherits or inherits another class, it can access all the functionality of the class it inherits from. Reusability enhances reliability. We just need to test and debug by super class code, no need to test each subclass.

+ When code is reused, it reduces development and maintenance costs.Hence, Inheritance helps limit code redundancy and aids code extensibility.

In our project:

```
13  public class main extends Application {
14      @Override
15      public void start(Stage stage) throws Exception {
16
17          try {
18              Parent root = FXMLLoader.load(getClass().getResource("main.fxml"));
19              Scene scene = new Scene(root);
20              stage.setTitle("Cell Division");
21              stage.setScene(scene);
22              stage.show();
```

### 3. Abstraction

Abstraction refers to the process of hiding the actual implementation of an application from users. Instead of we only emphasize how to use the application. The programmer can hide all the extraneous data or processes of the application. Regarding the user, those are just unnecessary details. Therefore, we can reduce the complexity and increase the efficiency of the software.

In our project:

```java
package model;

public interface cellOfFuntion {
    void DisplayCellOfInfor();

    void cellDivision();
}
```

### 4. Some other applications

Besides the above properties, we also use other basic techniques in object-oriented programming such as Object, Class, Interface, Abstract, Method overloading, Method, Override, Constant member class member, Exception handling and so on.

## V. REFERENCE

1. https://bom.so/Y7gt1c
2. https://bom.so/lVUQno
3. https://bom.so/4KfWxL
4. https://bom.so/WbJJ2v
5. https://bom.so/yyKsw7
6. https://bom.so/bqtJdu
7. https://bom.so/E1xprk
8. https://bom.so/5zsBsU
9. https://bom.so/XWlU0d
10. https://bom.so/kOeeQ7