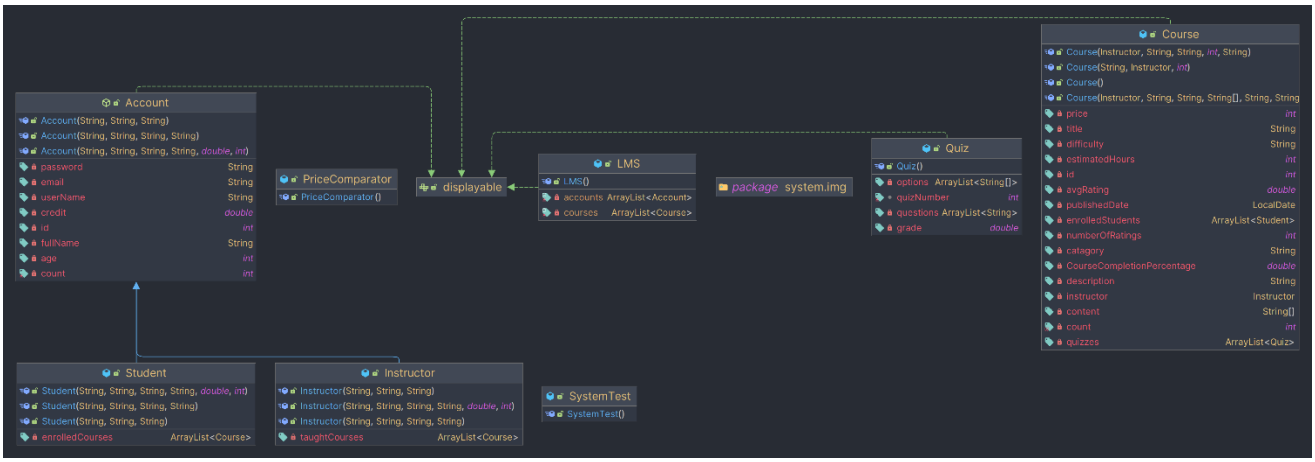
AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING

Name	ID
Abdelrahman Ezz Eldin Ismail	2101000
Farah amr mohamed	2101034
Omar Ashraf Abdelsatar	2100354
Rana Ayman Hamdy	2100830
Ahmed Ashraf Ali	2100255

Online-Courses-System

Picture of the design using UML:



Examples of Encapsulation:

In class LMS:

```
public class LMS implements displayable
{

    static private final ArrayList<Course> courses = new ArrayList<>();
    static private final ArrayList<Account> accounts = new ArrayList<>(); //

    public static ArrayList<Course> getCourses()
    {
        return courses;
    }

    static public Account findAccount(String userName)
    {
        for (Account account : accounts)
        {
            if (account.getUserName().equals(anObject: userName))
            {
                return account;
            }
        }
        return null;
    }

    public static ArrayList<Account> getAccounts()
    {
        return accounts;
    }
```

Examples of Inheritance:

Classes Student and Instructor inherits from public abstract class Account

```
public class Student extends Account
{

   public Student(String userName, String fullName, String password, String email, double credit, int age)
   {
      super(userName, fullName, password, email, credit, age);
   }
```

Examples of using polymarphism:

```
@Override
public void initialize(URL url, ResourceBundle rb)
{
    scrollPaneLayout.setVisible(true);
    addCourseBox.setVisible(false);
    if (App.account instanceof Instructor)
    {
        addCourseB.setVisible(true);
    }
    for (var course : LMS.getCourses())
    {
        addCard(course);
    }
}
```

Examples of using Interface:

```
package system;

/**
 *
 * @author 20109
 */
public interface displayable
{
   void display();
}
```

```java
@Override
public void display()
{
    System.out.println(x: this);

}
```

```java
public abstract class Account implements displayable
{

```

```java
private void displayInfo(ArrayList< ? extends displayable>objects)
{
    for (var obj : objects)
        obj.display();
}
```

exception handling:

```java
private boolean isValidFields() throws IllegalArgumentException
{
    if (username.getText().isEmpty() || email.getText().isEmpty() || password.getText().isEmpty() || confirmPassword.getText().isEmpty())
    {
        throw new IllegalArgumentException(s: "Please provide all data fields");
    }

    if (LMS.findAccount(userName: username.getText()) != null)
    {
        throw new IllegalArgumentException(s: "Username already exists");
    }

    if (!password.getText().equals(confirmPassword.getText()))
    {
        throw new IllegalArgumentException(s: "Passwords don't match");
    }

    return true;
}
```

```
public void setAge(int age)
{
    if (age < 0)
    {
        throw new IllegalArgumentException(s: "Input cannot be negative.");
    }
    try
    {

        this.age = age;

    }
    catch (InputMismatchException e)
    {

        System.out.println(x: "InputMismatchException");

    }

}
```

Sorting:

```
@FXML
private void sortCourses()
{
    ObservableList<Node> boxes = coursesPane.getChildren();
    List<Node> boxesList = new ArrayList<>(initialCapacity: boxes);
    PriceComparator sortingCondition = new PriceComparator();
    Collections.sort(list: boxesList, c: sortingCondition );
    coursesPane.getChildren().clear();
    coursesPane.getChildren().addAll(boxesList);
}
```

```
package system;

import java.util.Comparator;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;

/**
 *
 * @author A.Ashraf
 */
public class PriceComparator implements Comparator<Node>
{

    @Override
    public int compare(Node n1, Node n2)
    {
        VBox v1 = (VBox) n1, v2 = (VBox) n2;
        return Integer.valueOf(s: v1.getAccessibleText()).compareTo(Integer.valueOf(s: v2.getAccessibleText()));
    }
}
```