<u>Object Oriented Programming</u>
<u>CSE 1325 Project</u>

Submission Date - November 28th, 2023
Course Name- Object Oriented Programming
Course Code- CSE 1325
Project Name- University Portal System
Team Members - Samanza Ahmed, Jason Araujo, Adedoyin Omopariola

<u>UNIVERSITY PORTAL SYSTEM</u>

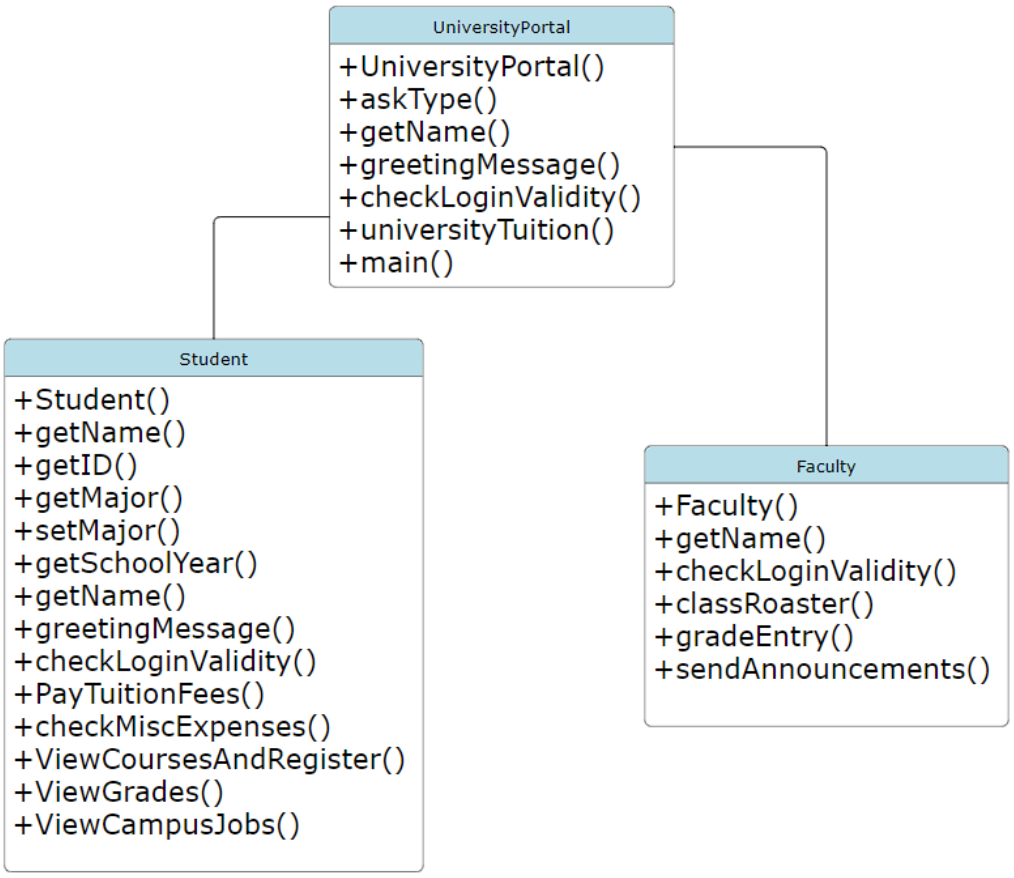
The University Portal System is an online platform that provides students and faculties of different institutions centralized and secure access to various academic and administrative resources.

A large range of transactions that take place at educational agencies, companies or corporations contribute to a large extent to the institutional experience of the users. That is colleges and universities with an enormous population or class size often utilizes an Administrative Software or a School Records Management System to facilitate both students and faculties to retrieve organizational information, campus activities or academic correspondences. While applications as such provide students with a gateway to view assignments, download class materials, enroll for courses or schedule advising for upcoming semesters, professors and faculties use it for course management, communication with students, obtain student performance reports and use it for many more purposes.

Acknowledging the importance of a school management software, our team has created a University Management portal system that provides users with various features and functions that are needed to upgrade user experience. We have implemented the concepts of Object Oriented Programming such as Encapsulation, Inheritance, Method Overriding, Classes and Objects in order to simulate the interface of an official University Management framework.

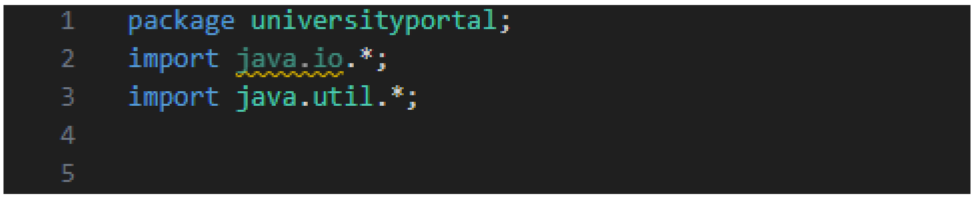
Program Organization

UML Diagram:



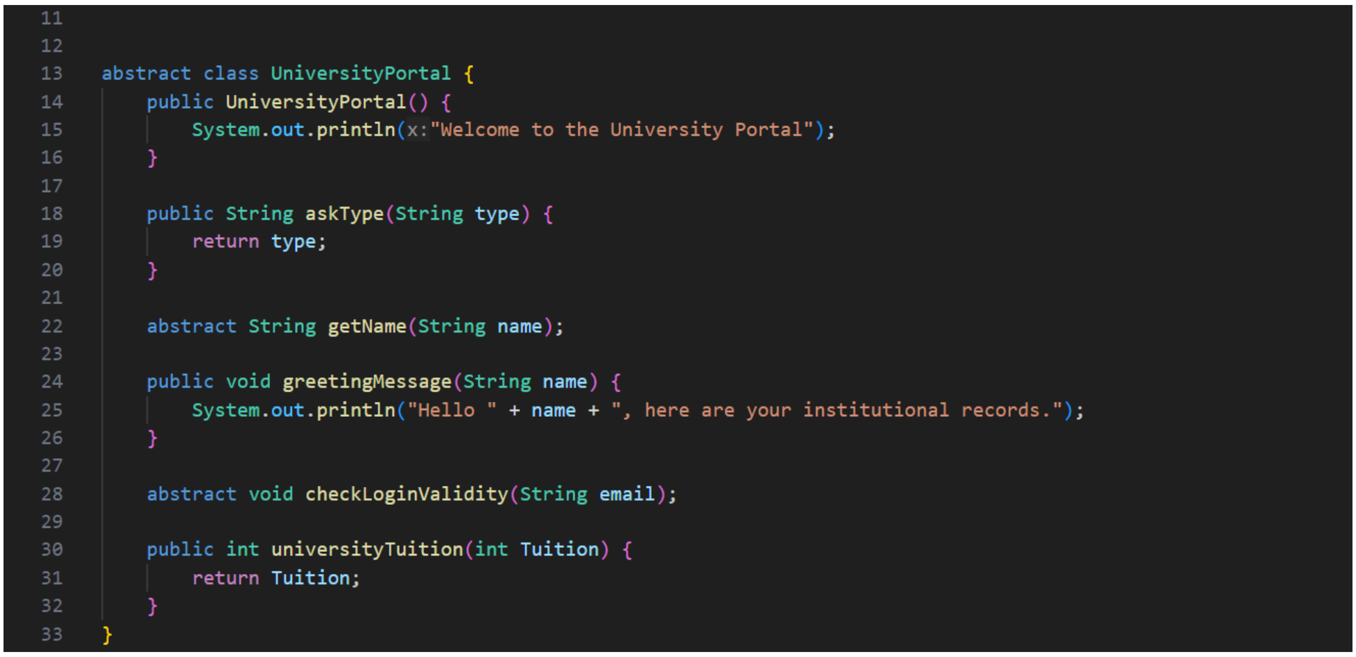
UniversityPortal Class:

The program starts off with importing the classes and packages that are needed to run the code.



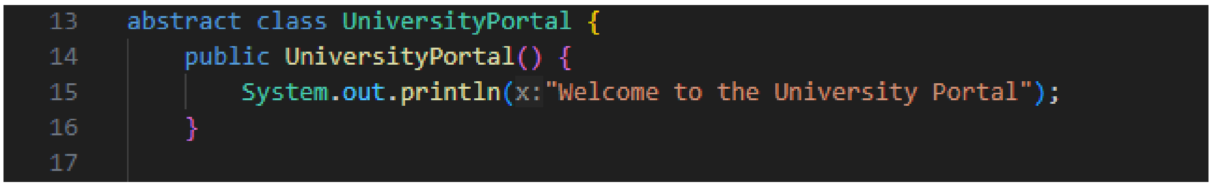
```java
package universityportal;
import java.io.*;
import java.util.*;


```

The UniversityPortal is an abstract class that has a constructor that is invoked each time objects are created from subclasses that extends the UniversityPortal class. It also has utility methods for asking things such as returning tuition and asking type, such as student or staff. This serves as a base foundation for both the Student and Faculty Class.



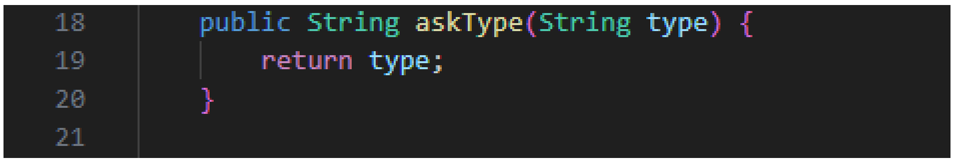
```java


abstract class UniversityPortal {
    public UniversityPortal() {
        System.out.println(x:"Welcome to the University Portal");
    }

    public String askType(String type) {
        return type;
    }

    abstract String getName(String name);

    public void greetingMessage(String name) {
        System.out.println("Hello " + name + ", here are your institutional records.");
    }

    abstract void checkLoginValidity(String email);

    public int universityTuition(int Tuition) {
        return Tuition;
    }
}
```

The first line starts by declaring the abstract class UniversityPortal. It is then followed by a constructor of the same name as the class which prints a welcome message when an instance of a subclass is created.



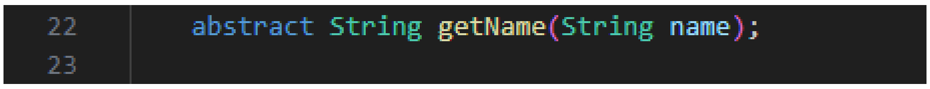
```java
abstract class UniversityPortal {
    public UniversityPortal() {
        System.out.println(x:"Welcome to the University Portal");
    }

```

A public method that returns a string is called askType, as well as takes a parameter of a string variable named “type”. This method returns the same string that was given through the parameter.



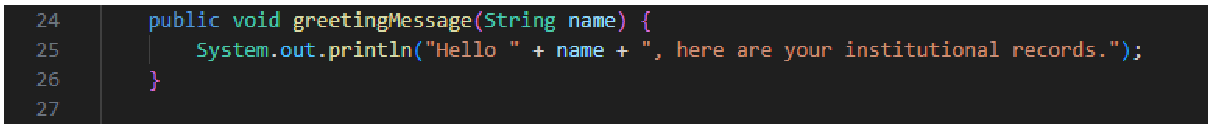
```java
    public String askType(String type) {
        return type;
    }

```

An abstract method that is meant to be implemented by subclasses that returns a string is called getName, that has a parameter of a string variable called “name”.



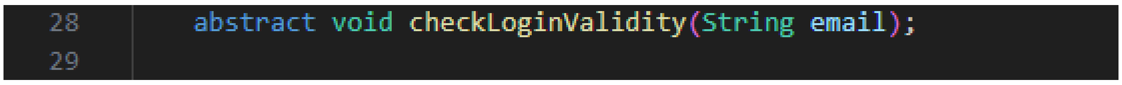
```java
    abstract String getName(String name);

```

A public method that does not explicitly return a type is called greetingMessage, and it has a parameter of a string variable called “name” which was used previously. This method prints out the greeting, “Hello (name), here are your institutional records.” where name is the parameter that was passed in when called.



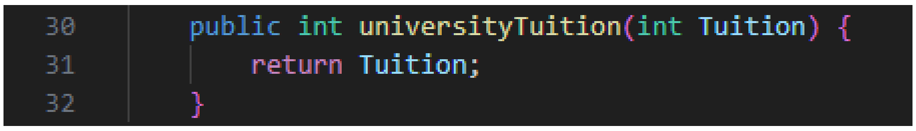
```java
    public void greetingMessage(String name) {
        System.out.println("Hello " + name + ", here are your institutional records.");
    }

```

An abstract method that will be implemented by subclasses that does not return a type is called checkLogInValidity, it has a parameter of the string variable “email”.



```java
    abstract void checkLoginValidity(String email);

```

The final method within the UniversityPortal class is the public method that returns an integer which is called UniversityTuition, it has the parameter integer variable “Tuition” and also returns that same variable within itself.



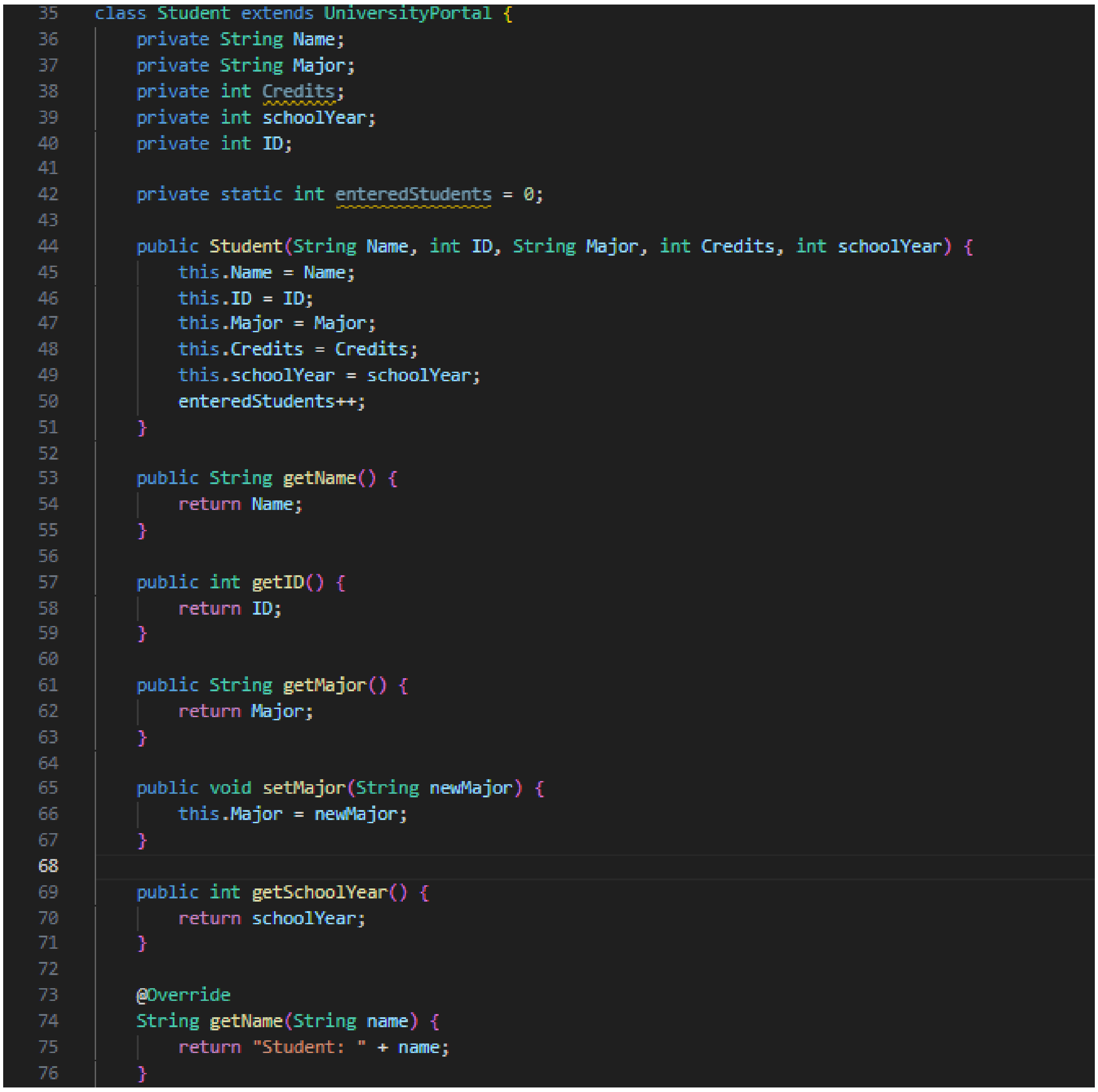
```java
public int universityTuition(int Tuition) {
    return Tuition;
}
```

The main method is located within the UniversityPortal class. It begins by creating an object as a student named “John Doe” with some predetermined information. It prompts the user by asking if they are a student or faculty member, if they choose a student then they are prompted with the following: asking for name, ID, major, credits completed, and school year. Once that information is entered a Student object is created with the entered information. It proceeds to send a greeting message and checks the login validity of the email you submit. If the login is valid, it presents a menu for options such as enrolling in classes, paying tuition fees, viewing CGPA, and viewing campus jobs. If the user chooses a faculty member, it prompts them for their email and name and creates a Faculty object with the entered information. It then proceeds to display a menu for options such as viewing class rosters, entering grades for students, and sending announcements.

Student Class:

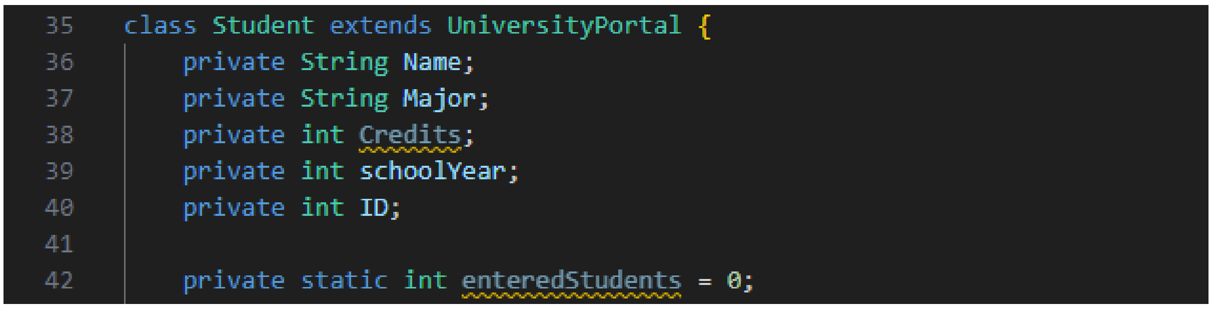
The Student class is a subclass of the abstract UniversityPortal class that integrates certain characteristics and attributes relating to students into the management structure of the university. It has private instance variables like Name, Major, Credits, schoolYear, and ID to retain personal student data. The class uses a static variable called enteredStudents to keep track of the total number of students entered.

The constructor initializes the data for each student as well as the total number of students who have been entered. The class provides getter and setter methods for accessing and changing student properties. It overrides some superclass methods (such as getName and checkLoginValidity) in order to provide student-specific implementations. Students can view and register for courses using the ViewCoursesAndRegister method, which saves the information in a file named RegisteredCourses.txt. Based on enrolled courses, the PayTuitionFees method calculates and returns the total tuition expenses. The ViewCoursesAndRegister method allows students to look over and register for courses; the information is kept in a file named RegisteredCourses.txt. The ViewGrades function retrieves and returns a student's GPA using the provided ID. All things considered, the Student class functions as a model for the qualities and conduct of a university student within the confines of the system.



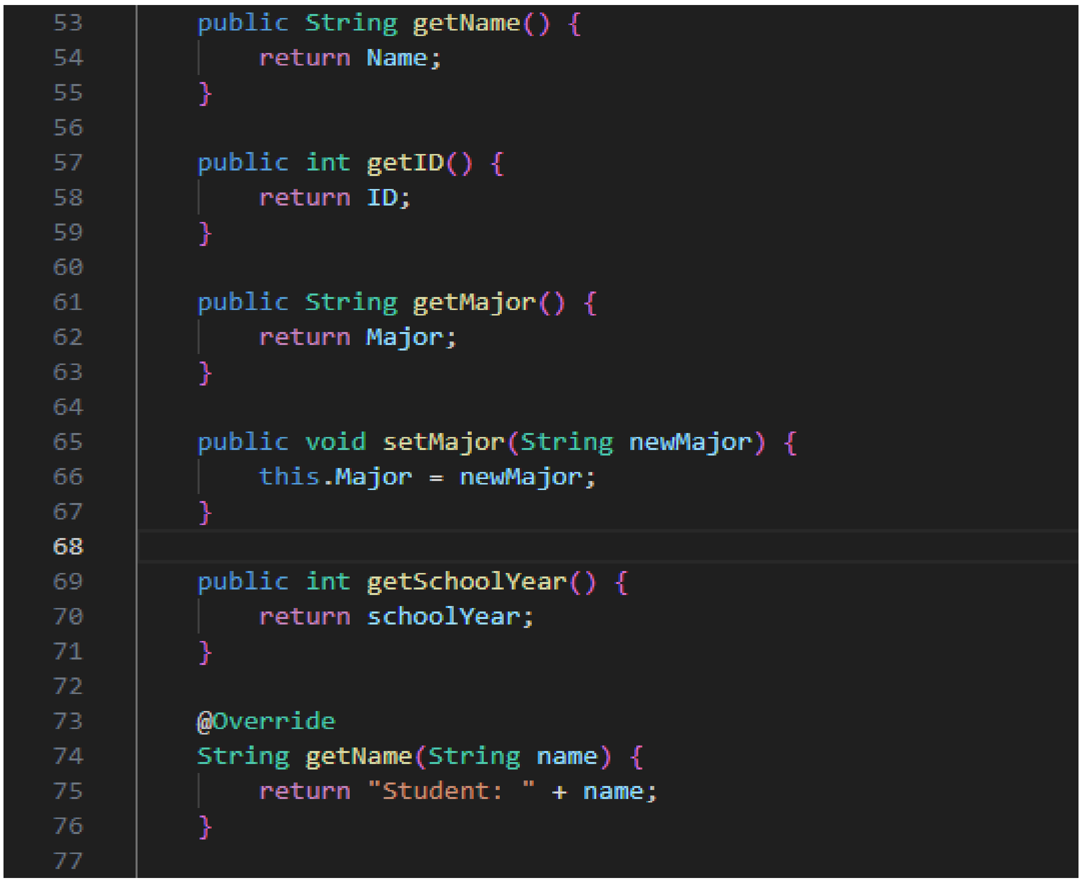
```java
class Student extends UniversityPortal {
    private String Name;
    private String Major;
    private int Credits;
    private int schoolYear;
    private int ID;

    private static int enteredStudents = 0;

    public Student(String Name, int ID, String Major, int Credits, int schoolYear) {
        this.Name = Name;
        this.ID = ID;
        this.Major = Major;
        this.Credits = Credits;
        this.schoolYear = schoolYear;
        enteredStudents++;
    }

    public String getName() {
        return Name;
    }

    public int getID() {
        return ID;
    }

    public String getMajor() {
        return Major;
    }

    public void setMajor(String newMajor) {
        this.Major = newMajor;
    }

    public int getSchoolYear() {
        return schoolYear;
    }

    @Override
    String getName(String name) {
        return "Student: " + name;
    }
```

The class begins by establishing the class named Student that is a subclass of the UniversityPortal. Following that are the instance variables of type string, which are Name and Major, as well as of type integer, which are Credits, schoolYear, and ID. Additionally a private static integer variable named enteredStudents is initialized to 0.



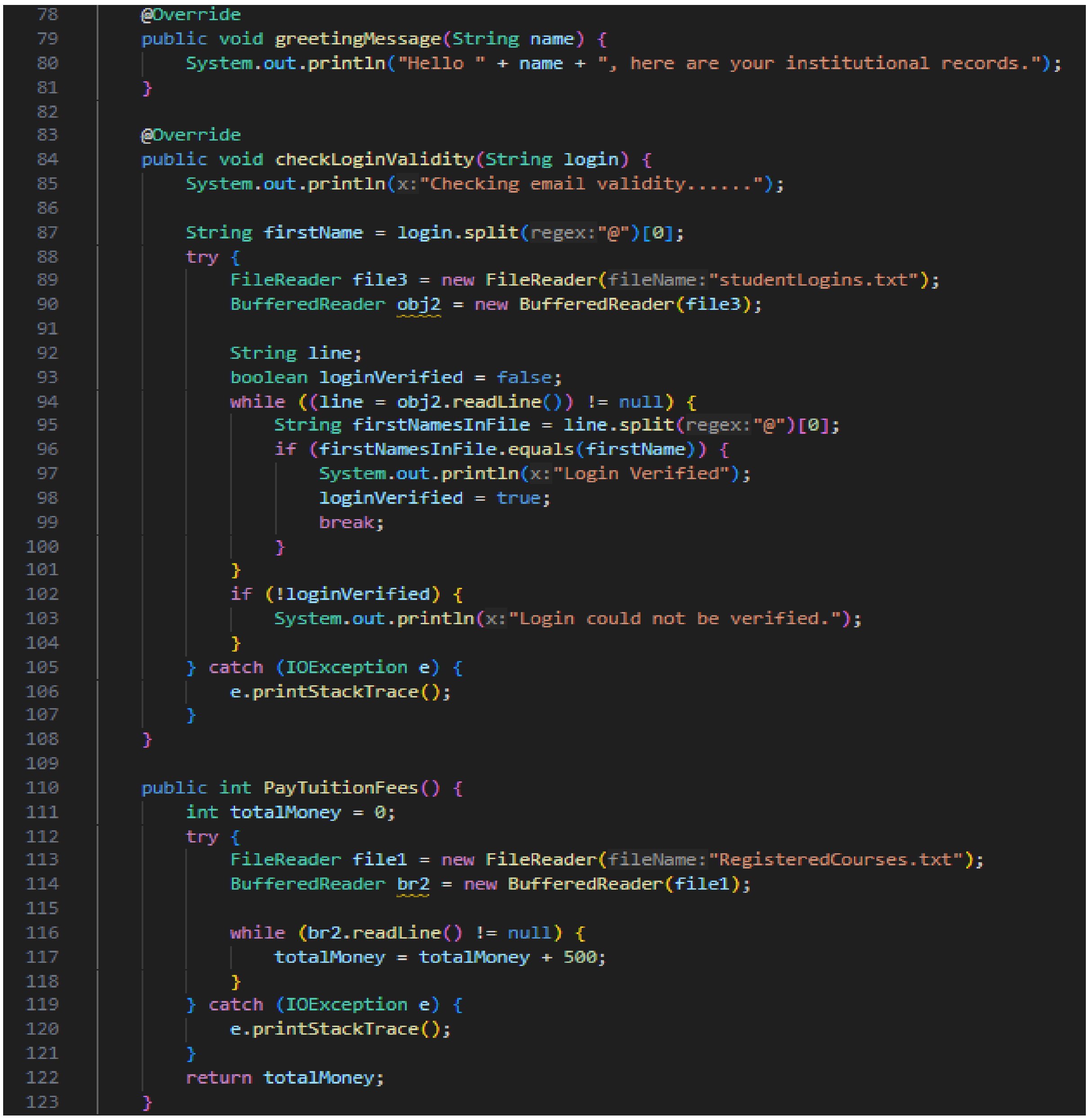
```java
class Student extends UniversityPortal {
    private String Name;
    private String Major;
    private int Credits;
    private int schoolYear;
    private int ID;

    private static int enteredStudents = 0;
```

The public Student constructor initializes a Student object with the parameters Name, ID, Major, Credits, schoolYear, and increments the enteredStudents variable. This is then followed by getter and setter methods, some overridden, for each variable that was initialized within the Student object, wherein they are either returning each variable or setting them equal to something.



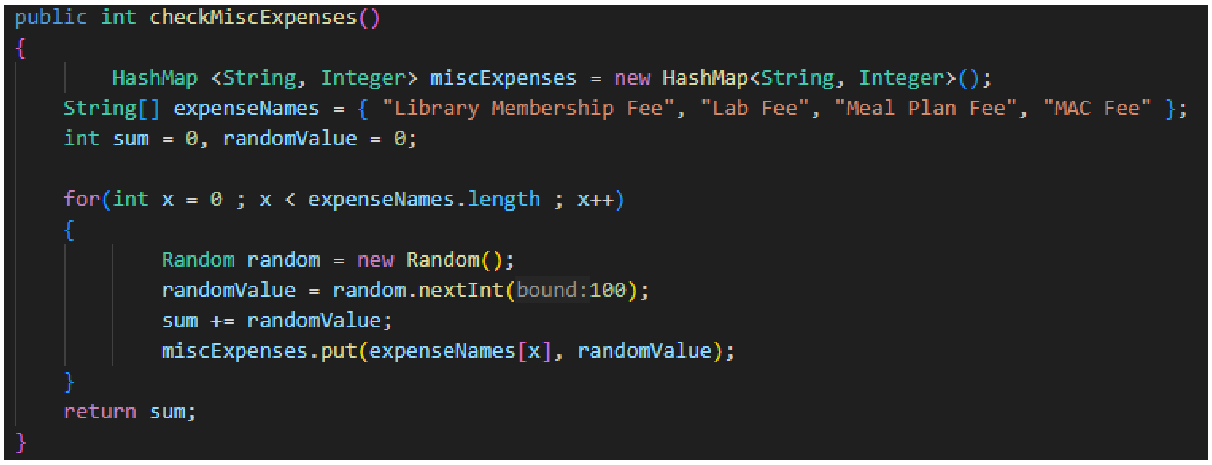
```java
    public String getName() {
        return Name;
    }

    public int getID() {
        return ID;
    }

    public String getMajor() {
        return Major;
    }

    public void setMajor(String newMajor) {
        this.Major = newMajor;
    }

    public int getSchoolYear() {
        return schoolYear;
    }

    @Override
    String getName(String name) {
        return "Student: " + name;
    }

```

These are followed by other overridden methods greetingMessage which prints a personalized message for the student, checkLogInValidity which prompts for an email and sifts through a separate file of emails and parses each email before the @ and checks to make sure that the email provided is valid before permitting entry or not. The public PayTuitionFees method which returns an integer reads through a separate file named RegisteredCourses.txt and for every course that is present, it increments the totalMoney variable by 500 and returns it at the end of the method.



```java
    @Override
    public void greetingMessage(String name) {
        System.out.println("Hello " + name + ", here are your institutional records.");
    }

    @Override
    public void checkLoginValidity(String login) {
        System.out.println(x:"Checking email validity......");

        String firstName = login.split(regex:"@")[0];
        try {
            FileReader file3 = new FileReader(fileName:"studentLogins.txt");
            BufferedReader obj2 = new BufferedReader(file3);

            String line;
            boolean loginVerified = false;
            while ((line = obj2.readLine()) != null) {
                String firstNamesInFile = line.split(regex:"@")[0];
                if (firstNamesInFile.equals(firstName)) {
                    System.out.println(x:"Login Verified");
                    loginVerified = true;
                    break;
                }
            }
            if (!loginVerified) {
                System.out.println(x:"Login could not be verified.");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public int PayTuitionFees() {
        int totalMoney = 0;
        try {
            FileReader file1 = new FileReader(fileName:"RegisteredCourses.txt");
            BufferedReader br2 = new BufferedReader(file1);

            while (br2.readLine() != null) {
                totalMoney = totalMoney + 500;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return totalMoney;
    }
```

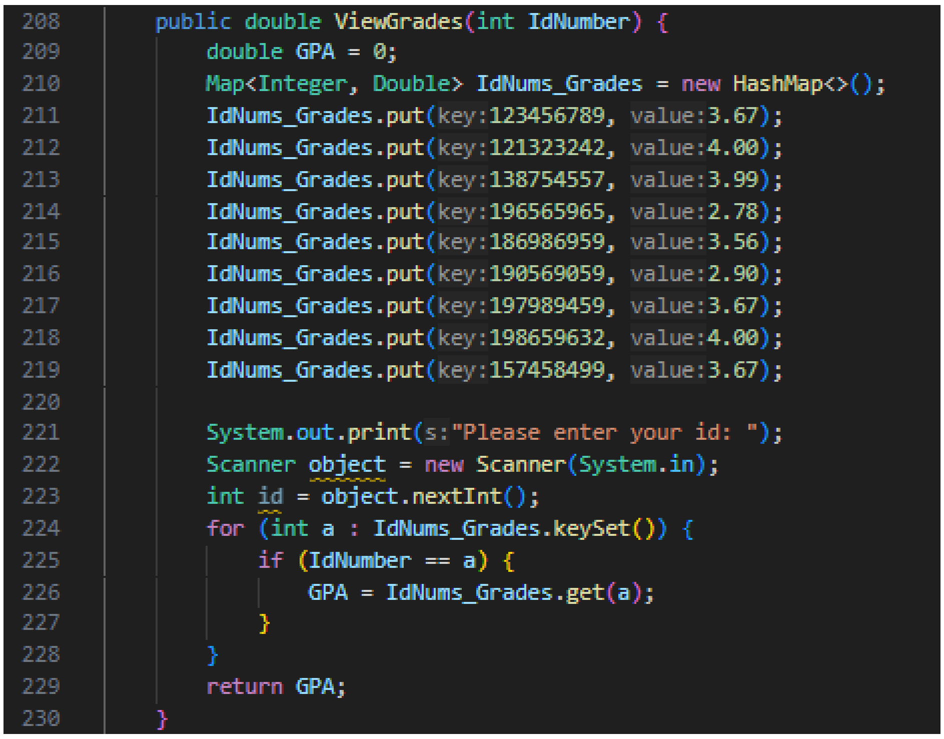
The public method named checkMiscExpenses is intended to return an integer value with no parameters. It begins by creating a hashmap named miscExpenses that takes a String and Int for its key and value data. A string array named expenseNames is created that contains all the names of the expenses the student has. A loop is run until the end of the expenseNames array, and each time is generating a random number to accompany each expense name when being added into the miscExpenses hashmap. Then the sum of all expenses is added and returned at the end.



```
public int checkMiscExpenses()
{
        HashMap <String, Integer> miscExpenses = new HashMap<String, Integer>();
    String[] expenseNames = { "Library Membership Fee", "Lab Fee", "Meal Plan Fee", "MAC Fee" };
    int sum = 0, randomValue = 0;

    for(int x = 0 ; x < expenseNames.length ; x++)
    {
        Random random = new Random();
        randomValue = random.nextInt(bound:100);
        sum += randomValue;
        miscExpenses.put(expenseNames[x], randomValue);
    }
    return sum;
}
```

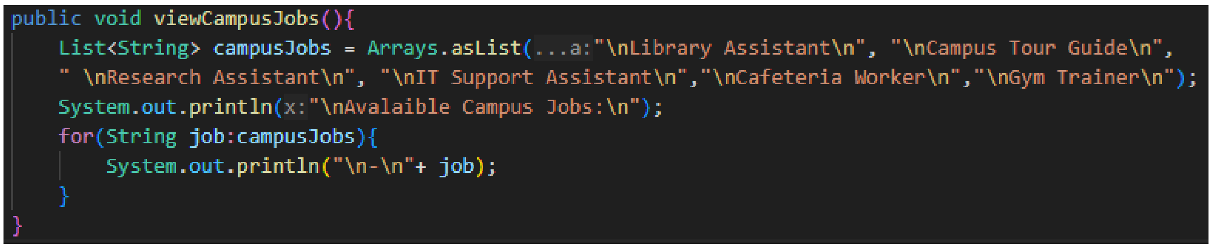
The public method named ViewCoursesAndRegister begins by making an Arraylist named Department, a hashmap named degreePlan, and an integer variable named i initialized to 0. It then presents available courses based on major, then stores 4 courses based on the user's decision within the text file RegisteredCourses.txt.

This method is followed by another public method named ViewGrades that returns a double and has a parameter of an integer variable named “IdNumber”. It retrieves and returns the GPA of a student based on the ID which is all stored in a hashmap called IdNums_Grades.



```java
    public double ViewGrades(int IdNumber) {
        double GPA = 0;
        Map<Integer, Double> IdNums_Grades = new HashMap<>();
        IdNums_Grades.put(key:123456789, value:3.67);
        IdNums_Grades.put(key:121323242, value:4.00);
        IdNums_Grades.put(key:138754557, value:3.99);
        IdNums_Grades.put(key:196565965, value:2.78);
        IdNums_Grades.put(key:186986959, value:3.56);
        IdNums_Grades.put(key:190569059, value:2.90);
        IdNums_Grades.put(key:197989459, value:3.67);
        IdNums_Grades.put(key:198659632, value:4.00);
        IdNums_Grades.put(key:157458499, value:3.67);

        System.out.print(s:"Please enter your id: ");
        Scanner object = new Scanner(System.in);
        int id = object.nextInt();
        for (int a : IdNums_Grades.keySet()) {
            if (IdNumber == a) {
                GPA = IdNums_Grades.get(a);
            }
        }
        return GPA;
    }
```

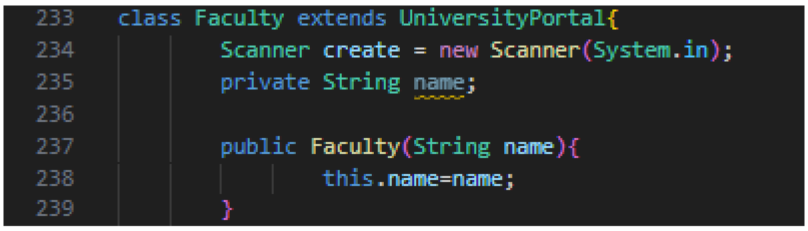
The public method viewCampusJobs does not explicitly return a type and has no parameters. It begins by creating an array list named campusJobs that contains all the jobs there are on campus. Using an enhanced for loop, for each job in the list, it prints a message indicating the job's availability, with a line break before and after each job.



```java
public void viewCampusJobs(){
    List<String> campusJobs = Arrays.asList(...a:"\nLibrary Assistant\n", "\nCampus Tour Guide\n",
    " \nResearch Assistant\n", "\nIT Support Assistant\n","\nCafeteria Worker\n","\nGym Trainer\n");
    System.out.println(x:"\nAvalaible Campus Jobs:\n");
    for(String job:campusJobs){
        System.out.println("\n-\n"+ job);
    }
}
```

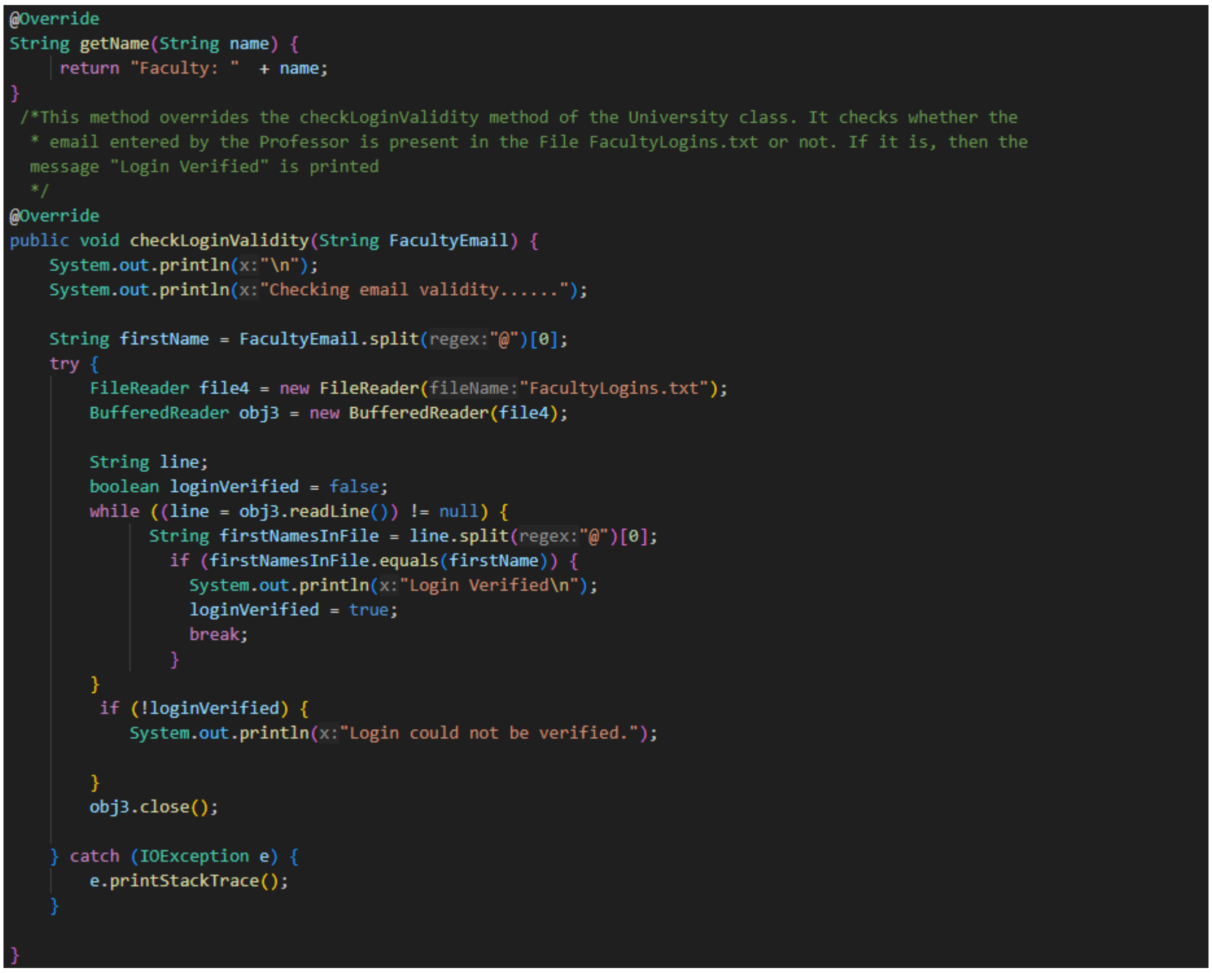
Faculty Class:

Faculty members can check the validity of their login, access class lists, enter student grades, and make announcements using the Faculty class methods. It adds unique functionality to faculty communication in the UniversityPortal class and extends functionality defined in the abstract UniversityPortal class.



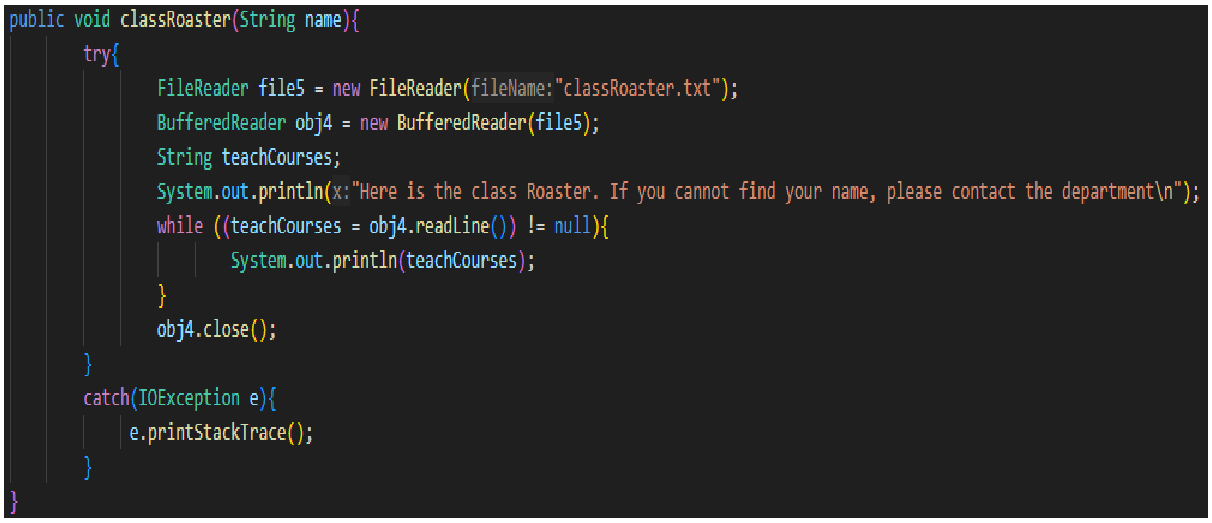
```java
class Faculty extends UniversityPortal{
        Scanner create = new Scanner(System.in);
        private String name;

        public Faculty(String name){
                this.name=name;
        }
```

Right after this, the recurring getName and checkLogInValidity methods are overridden into the faculty class again.

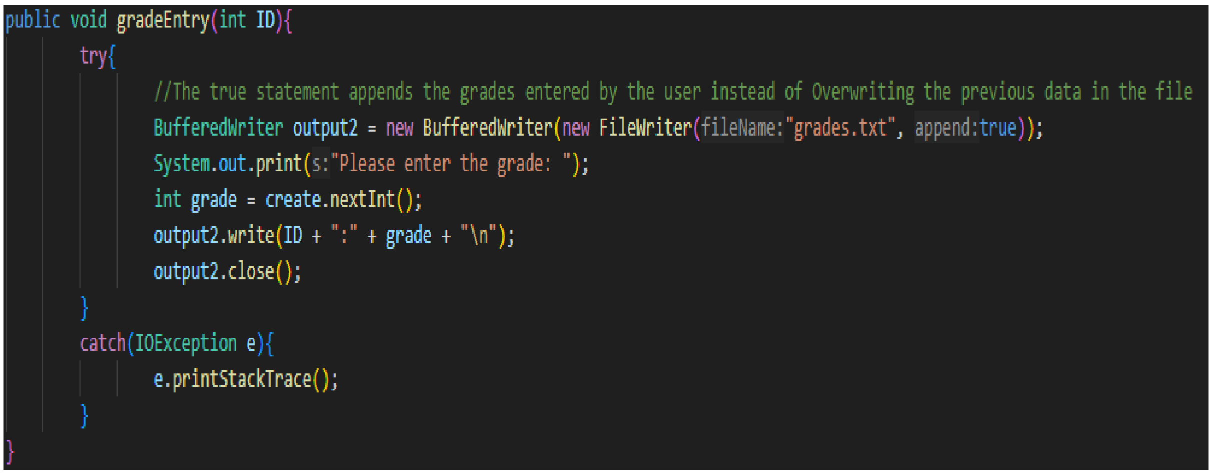


```java
@Override
String getName(String name) {
    return "Faculty: "  + name;
}
 /*This method overrides the checkLoginValidity method of the University class. It checks whether the
  * email entered by the Professor is present in the File FacultyLogins.txt or not. If it is, then the
  message "Login Verified" is printed
  */
@Override
public void checkLoginValidity(String FacultyEmail) {
    System.out.println(x:"\n");
    System.out.println(x:"Checking email validity......");

    String firstName = FacultyEmail.split(regex:"@")[0];
    try {
        FileReader file4 = new FileReader(fileName:"FacultyLogins.txt");
        BufferedReader obj3 = new BufferedReader(file4);

        String line;
        boolean loginVerified = false;
        while ((line = obj3.readLine()) != null) {
              String firstNamesInFile = line.split(regex:"@")[0];
                if (firstNamesInFile.equals(firstName)) {
                  System.out.println(x:"Login Verified\n");
                  loginVerified = true;
                  break;
                }
        }
         if (!loginVerified) {
            System.out.println(x:"Login could not be verified.");

        }
        obj3.close();

    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

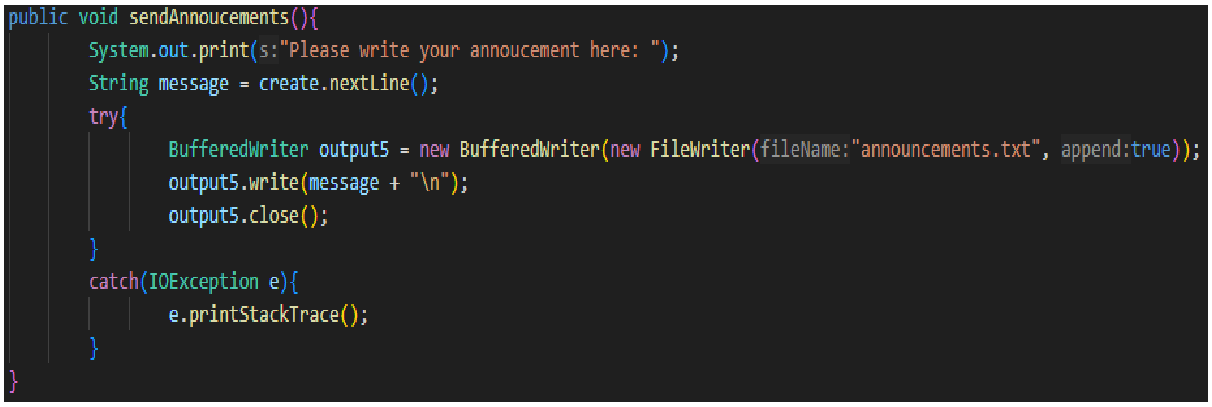
With an additional new public method named classRoaster that does not explicitly return a type and has a parameter of a string variable called “name”. This reads and prints the contents of the classRoaster.txt file to the faculty member.


```java
public void classRoaster(String name){
        try{
                FileReader file5 = new FileReader(fileName:"classRoaster.txt");
                BufferedReader obj4 = new BufferedReader(file5);
                String teachCourses;
                System.out.println(x:"Here is the class Roaster. If you cannot find your name, please contact the department\n");
                while ((teachCourses = obj4.readLine()) != null){
                        System.out.println(teachCourses);
                }
                obj4.close();
        }
        catch(IOException e){
            e.printStackTrace();
        }
}
```

A public method follows, named gradeEntry that doesn’t explicitly return a type but with a parameter of an integer variable called “ID”. This prompts the faculty member to enter a grade for a student identified by their ID, then those grades are added to the text file grades.txt.


```java
public void gradeEntry(int ID){
        try{
                //The true statement appends the grades entered by the user instead of Overwriting the previous data in the file
                BufferedWriter output2 = new BufferedWriter(new FileWriter(fileName:"grades.txt", append:true));
                System.out.print(s:"Please enter the grade: ");
                int grade = create.nextInt();
                output2.write(ID + ":" + grade + "\n");
                output2.close();
        }
        catch(IOException e){
                e.printStackTrace();
        }
}
```

Finally, a public method that does not explicitly return a type named sendAnnouncements with no parameters allows faculty to input announcements that are added to the announcements.txt text file.



```
public void sendAnnoucements(){
        System.out.print(s:"Please write your annoucement here: ");
        String message = create.nextLine();
        try{
                BufferedWriter output5 = new BufferedWriter(new FileWriter(fileName:"announcements.txt", append:true));
                output5.write(message + "\n");
                output5.close();
        }
        catch(IOException e){
                e.printStackTrace();
        }
}
```