***Course Submission Cover Sheet***

LONDON METROPOLITAN UNIVERSITY

*Module: CC4001 Programming*

*Assignment no: 003*

*Weighting: 60% of module mark*

*Deadline: Friday 7th of May 2025*

Module Leaders: Dr Sandra Fernando & Dr Sahar Al-Sudani

Student ID:

Please note that there are specific regulations concerning **the use of AI tools and Academic Misconduct**. Below are extracts from these regulations. By signing, you acknowledge that you have read and understood these extracts.

(signature:)_____Date: Tue May 6, 2025

This header sheet should be attached to the work you submit.

Academic Integrity means being honest in your academic work and your studies and making sure that you acknowledge the work of others and giving credit where you have used other people's ideas as part of presenting your arguments. Your assessment submissions must therefore always be entirely your own work, based on your own learning and appropriately referenced including how you have used Generative AI. The University regards the use of Generative AI applications by students to deceive to gain unfair advantage as **academic misconduct**. This usage includes:

- **Plagiarism**, where AI tools are used to generate output and ideas that are presented or submitted as if they were the student's own work, without proper citation or references.

- Where a complete assignment is created using Generative AI and represented as a student's own work, this will be regarded as contract cheating in the same way as commissioning an 'Essay Mill' or other third party to complete your work. Further information can be found on : Guidance on the use of Artificial Intelligence.

**Academic misconduct:** The University takes academic misconduct very seriously and seeks at all times to rigorously protect its academic standards. Plagiarism, collusion and other forms of cheating constitute academic misconduct, for which there is an explicit range of graduated penalties depending on the particular type of academic misconduct. The penalties that can be applied if academic misconduct is substantiated range from a reprimand to expulsion in very serious cases and for repeated instances of misconduct. You are also responsible for ensuring that all work submitted is your own and that it is appropriately referenced. The University does not tolerate cheating of any kind. You are strongly advised to familiarise yourself with the Academic Misconduct Policy and Procedure (Academic Misconduct), which list a range of categories of academic misconduct and associated penalties, covering instances of academic misconduct (plagiarism, collusion, exam cheating).

# ING College Staff Hiring System

Module CS4001
Jose Alejandro Pestana Felibert
ID 23025663

Submission Date, Tue May 6, 2025

https://github.com/HeyItsAruCoding/StaffRecruitmentSystem

---

This Java-based application allows ING College to manage the hiring of full-time and part-time staff. It provides a GUI to add vacancies, appoint staff, terminate part-time staff, and display staff information. All data is saved to a file and reloaded automatically on startup.

**The next Features can be found on my app:**

- Add Full-Time and Part-Time staff vacancies
- Appoint staff to vacancies with validation
- Prevent duplicate vacancy numbers
- Prevent staff re-appointment or re-termination
- Terminate part-time staff and reflect status in records
- Display all staff data in a scrollable view
- Save all data to a staff.txt file automatically
- Load data from file at startup (persistent storage)
- Clear all input fields using a single button
- Error handling and input validation throughout

**Object-Oriented Design:**

The project is structured using strong object-oriented programming (OOP) principles:

- **StaffHire (Abstract Class):**
  Defines common attributes and methods for any type of staff hire, including vacancy number, designation, and job type.

- **FullTimeStaffHire (Subclass):**
  Inherits from StaffHire. It adds salary and working hours attributes and logic for appointing a

full-time staff member.

- **PartTimeStaffHire (Subclass):**
  Also inherits from StaffHire. Includes attributes like wages per hour, shift, and termination status. It contains custom methods for hiring and terminating part-time staff.

- **INGCollege (Main Class):**
  Controls the GUI using JFrame, JLabel, JTextField, and JButton. Manages user interactions and connects front-end to back-end logic. It handles file reading and writing using Java I/O classes (FileWriter, BufferedReader, etc.).

Instructions for the App and System:

1. **Launching the Application:**
   Open the project in BlueJ, right-click the INGCollege class, and select void main(String[] args).

2. **Adding Vacancies:**
   a. Use the Full-Time or Part-Time section on the left and right, respectively.
   b. Fill in vacancy number, designation, salary/wage, working hours, etc.
   c. Click **Add Full-Time Vacancy** or **Add Part-Time Vacancy**.

3. **Appointing Staff:**
   a. Enter a valid vacancy number and staff details.
   b. Click **Appoint Full-Time Staff** or **Appoint Part-Time Staff**.

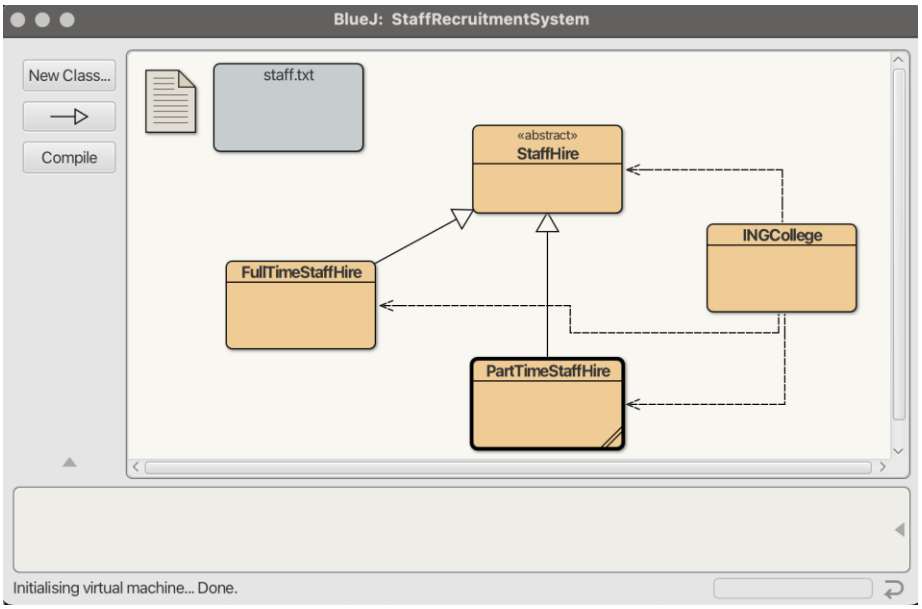
4. **Terminating Part-Time Staff:**
   a. Enter the part-time vacancy number.
   b. Click **Terminate Part-Time Staff**.

5. **Viewing All Staff:**
   a. Click **Display Staff** to see all current and appointed records.

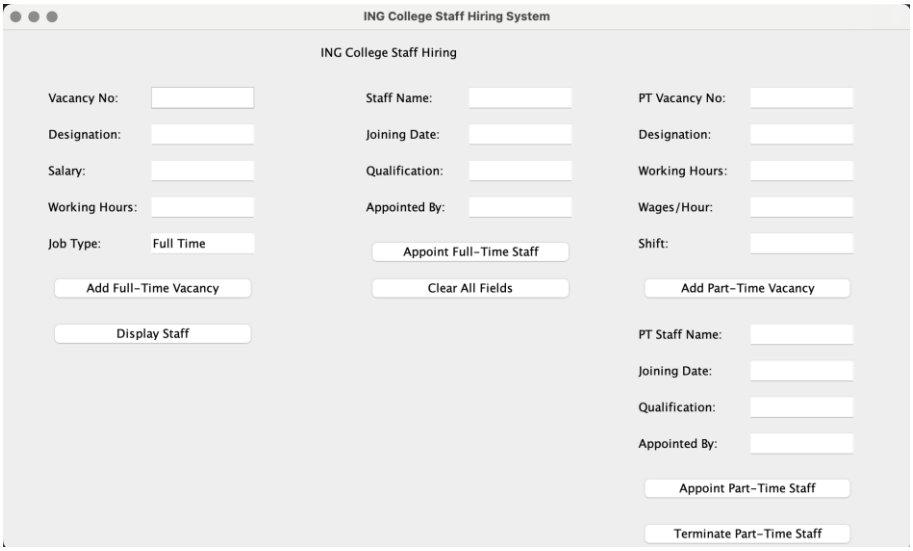
6. **Clearing Fields:**
   a. Click **Clear All Fields** to reset all input boxes.

**<u>Heres some samples of how the App runs</u>**:

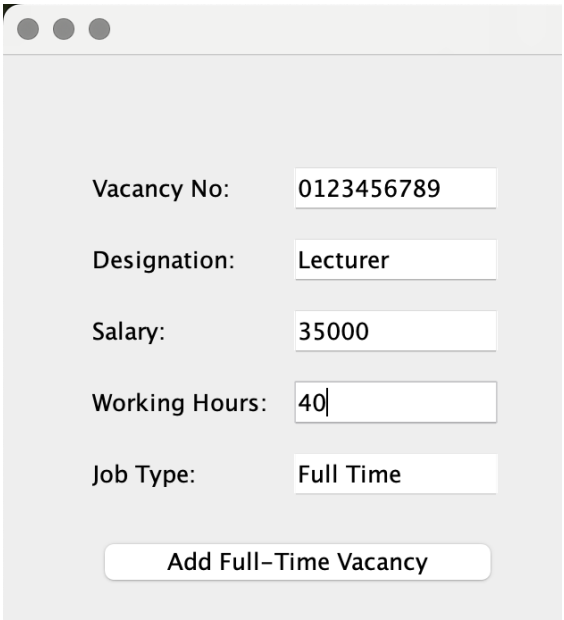
- Here's the opened diagram on BlueJ

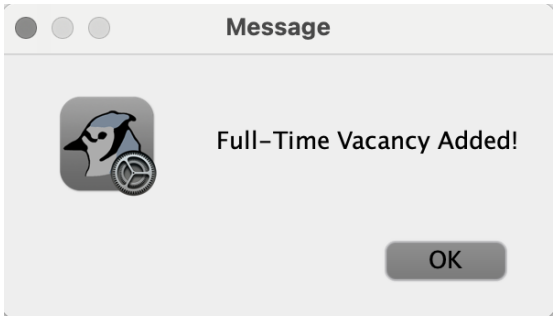



- The moment we Run the App will show us the GUI, showing all the options and input choices for Staff

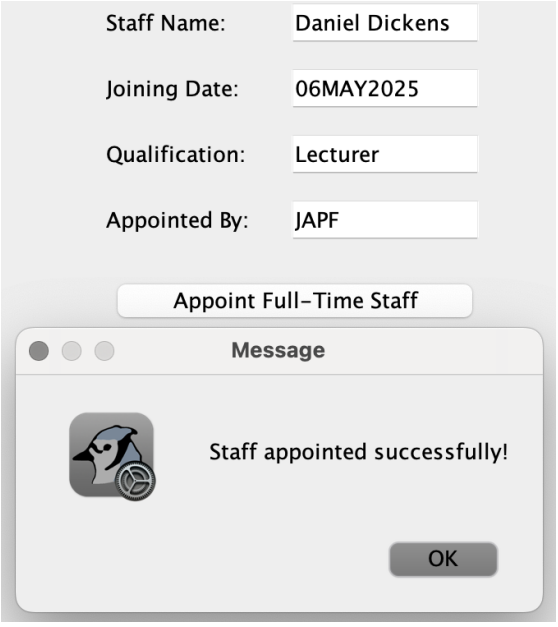



- Now we can add a Full-Time staff member, and we will get the next pop-up

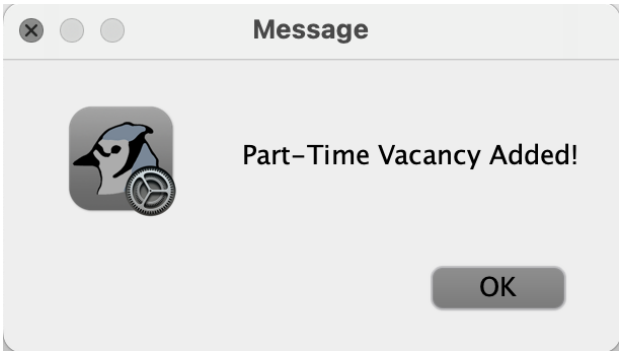


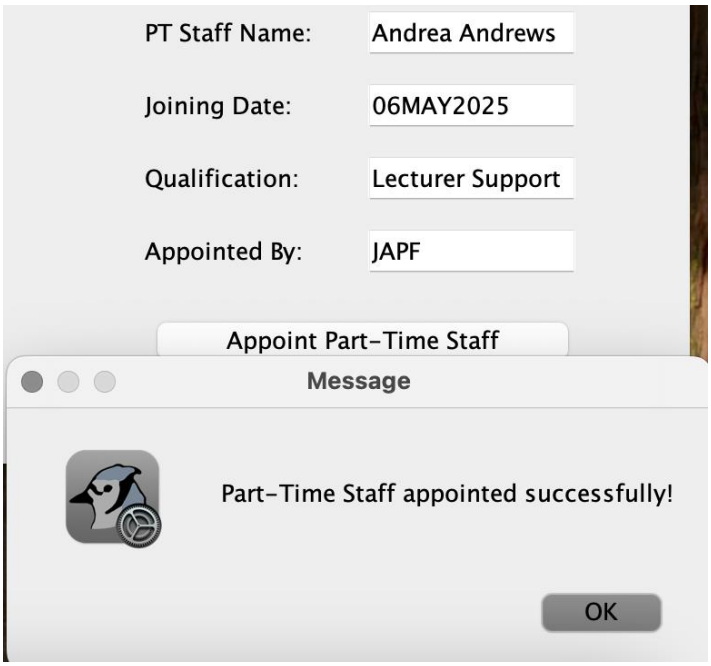

- Same when we want to Appoint a Full-Time staff



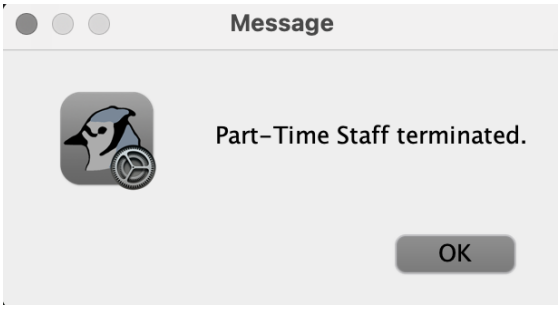

- Now we can add a Part-Time staff member, and we will get the next pop-up

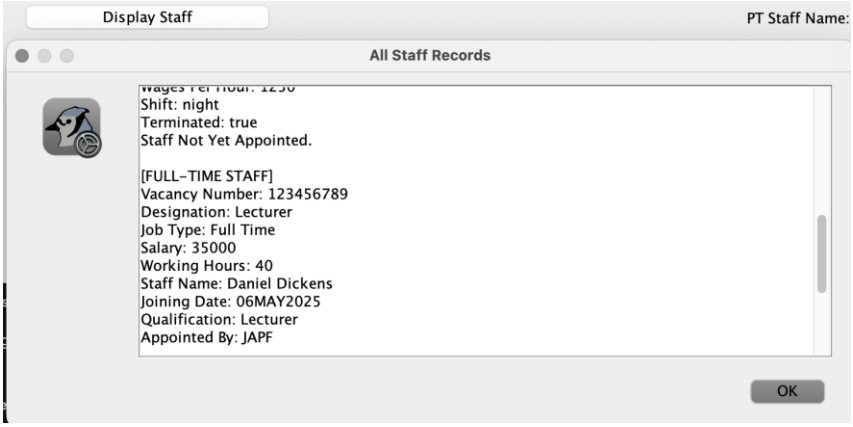
- Same when we want to Appoint a Part-Time staff

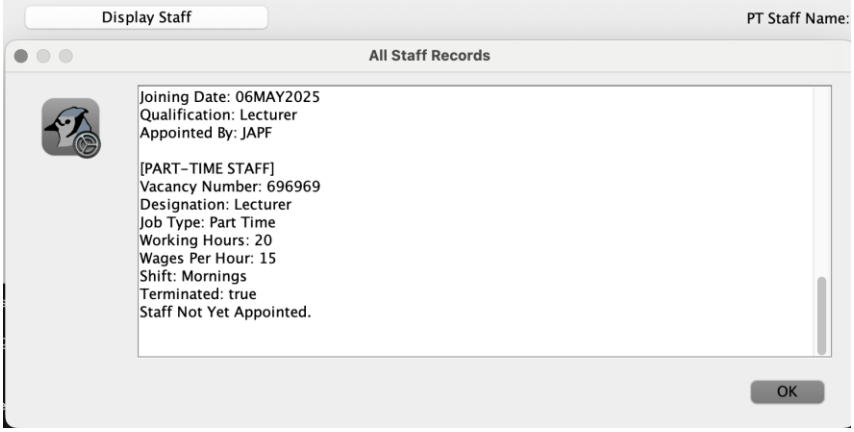



- Now let's terminate a Part-Time staff member, for the termination to work all the info of this staff must be written on the app, so we can get...

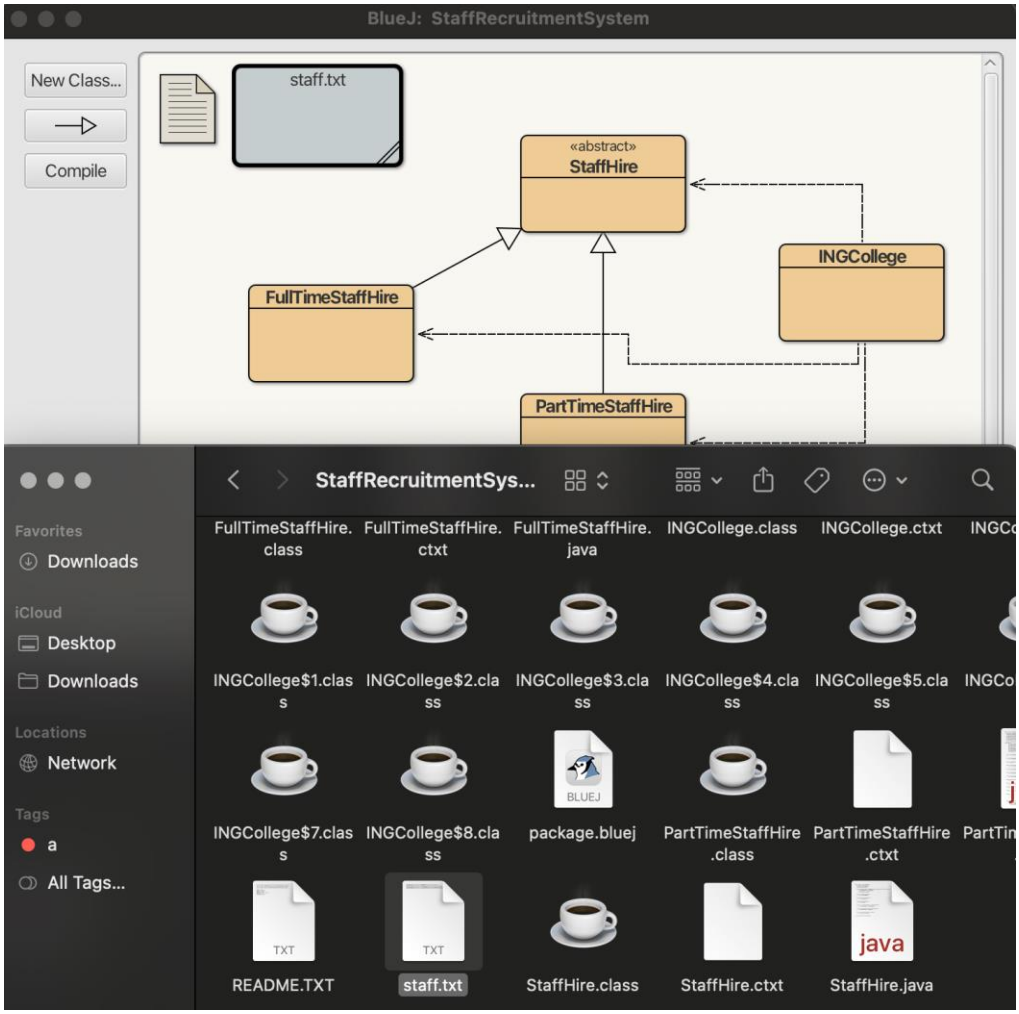



- Now, as a personal touch and apart from the project i thought would be better to have displayed all the info and staff that was included on the Hiring process, so if we click the option that says “Display Staff” well get the next pop-up






- All this information can be checked and edited on the staff.txt file inside the app that will be created after the inputs, it will also save itself and remember until changed.




**Reflection:**

During this project, I gained a much deeper understanding of GUI development in Java — especially using BlueJ, which I had never worked with before. Learning how to design and implement interactive interfaces using Swing was both challenging and rewarding. We also explored core object-oriented programming principles such as inheritance and polymorphism, and I now feel much more confident in applying them correctly in practice.

One of the most valuable aspects of this project was getting hands-on experience with file handling and input validation, which made the application feel completer and more realistic. A key challenge I encountered was finding ways to improve the system beyond the basic requirements. For example, I wanted to ensure the staff data would persist even after closing the program, so I researched and implemented file saving and loading using .txt files. I also thought it would be more user-friendly to add a “Clear All Fields” button to reset the form easily — and learning how to implement that on my own was a great learning moment.

Overall, this project pushed me to think critically, improve the user experience, and apply theory in practical ways.

**<u>Apendix A:</u>**

Source Files Included ->

- StaffHire.java *Abstract class defining shared properties of all staff types*

- public int getVacancyNumber()
  *Returns the vacancy number of the staff hire.*

- public String getDesignation()
  *Returns the job designation (e.g., Lecturer, Assistant)*

- public String getJobType()
  *Returns the job type (e.g., Full-Time, Part-Time)*

- public void setDesignation(String designation)
  *Updates the designation for this vacancy.*

- public void setJobType(String jobType)
  *Updates the job type for this vacancy.*

- public void display()
  *Abstract method to be implemented by subclasses to print staff info.*

- FullTimeStaffHire.java *Subclass for full-time staff, includes salary, hours, etc.*

- PartTimeStaffHire.java *Subclass for part-time staff, includes shift, wage, etc.*

- INGCollege.java *Main GUI class that handles all interactions and logic*

*External Files Generated ->*

- staff.txt *Auto-generated file storing all staff information*

*Additional Files Included ->*

- *README.txt* Step-by-step guide on how to use the program
- CS4001_Documentation_JAPF.pdf *This documentation report*

**<u>Apendix B:</u>**

**1. Add Full-Time Vacancy**

START
IF input fields are filled and valid
PARSE vacancy number, salary, and hours

IF vacancy number does not already exist in staffList
CREATE FullTimeStaffHire object
ADD to staffList
SAVE to file
SHOW success message
ELSE|
SHOW error: vacancy already exists
ELSE
SHOW error: invalid input
END

## 2. Appoint Full-Time Staff

**START**
**PARSE vacancy number**
**SEARCH staffList for FullTimeStaffHire with matching vacancy number**
**IF found AND staff not already appointed**
**ASSIGN staff name, qualification, joining date, and appointed by**
**SAVE to file**
**SHOW success message**
**ELSE IF already appointed**
**SHOW message: staff already appointed**
**ELSE**
**SHOW message: vacancy not found**
**END**

## 3. Add Part-Time Vacancy

START
IF input fields are filled and valid
PARSE vacancy number, working hours, wages, and shift
IF vacancy number does not exist in staffList
CREATE PartTimeStaffHire object
ADD to staffList
SAVE to file
SHOW success message
ELSE
SHOW error: vacancy already exists
ELSE
SHOW error: invalid input
END

## 4. Appoint Part-Time Staff

START
PARSE vacancy number
SEARCH staffList for PartTimeStaffHire with matching vacancy number
IF found AND staff not yet appointed
ASSIGN staff name, qualification, joining date, and appointed by
SAVE to file
SHOW success message
ELSE IF already appointed
SHOW message: staff already appointed
ELSE

```
SHOW message: vacancy not found
END
```

### 5. Terminate Part-Time Staff

```
START
PARSE vacancy number
SEARCH staffList for PartTimeStaffHire with matching vacancy number
IF found AND staff not already terminated
CLEAR staff name, qualification, joining date, and appointed by
SET joined to false, terminated to true
SAVE to file
SHOW termination message
ELSE IF already terminated
SHOW message: already terminated
ELSE
SHOW message: vacancy not found
END
```

### 6. Display Staff

```
START
IF staffList is not empty
CREATE text area
FOR each staff in staffList
  IF FullTimeStaffHire → call getDisplayText()
  IF PartTimeStaffHire → call getDisplayText()
SHOW text area in scrollable popup
ELSE
SHOW message: no records found
END
```

## Apendix C:

Class: StaffHire (abstract)
+ int vacancyNumber
+ String designation
+ String jobType
+ getVacancyNumber(): int
+ getDesignation(): String
+ getJobType(): String
+ setDesignation(String): void
+ setJobType(String): void
+ display(): void

Class: FullTimeStaffHire extends StaffHire
+ int salary
+ int workingHour

+ String staffName
+ hireFullTimeStaff(...)
+ getDisplayText(): String
+ toString(): String

Class: PartTimeStaffHire extends StaffHire
+ int wagesPerHour
+ String shift
+ boolean terminated
+ hirePartTimeStaff(...)
+ terminateStaff()
+ getDisplayText(): String
+ toString(): String

Class: INGCollege
+ JFrame frame
+ ArrayList<StaffHire> staffList
+ saveStaffToFile(): void
+ loadStaffFromFile(): void
+ GUI setup and button logic

**Apendix D:**

## 1. NullPointerException

While attempting to appoint a part-time staff member, a NullPointerException occurred because the program tried to call hirePartTimeStaff() before verifying that the object existed in the staffList.

**Fix:**
Moved the method call inside the if block that checks for a matching vacancy. This ensured the object was not null before calling its methods.

**Example Error Message:**

```
php
CopyEdit
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
  at INGCollege$4.actionPerformed(INGCollege.java:211)
```

## 2. Unresolved Class: FileWriter

**T**he compiler threw an error stating that FileWriter could not be resolved. This happened because the necessary import statement was missing at the top of the file.

**Fix:**
Added the following import statement:

java
CopyEdit
import java.io.FileWriter;

Also wrapped the file-writing logic inside a try-catch block to handle any I/O exceptions safely.

## 3. Data Not Saving Between Sessions

Initially, all staff records were lost after closing and reopening the application. The GUI displayed an empty list even after appointing staff.

**Fix:**
Implemented two methods:

- saveStaffToFile() to write data to a text file after each action
- loadStaffFromFile() to load all records when the program starts

This ensured data persistence across sessions using a file named staff.txt.