### Logos - London Metropolitan UniversityCourse Submission Cover Sheet

### 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:

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.](https://student.londonmet.ac.uk/your-studies/student-administration/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](https://student.londonmet.ac.uk/your-studies/student-administration/rules-and-regulations/academic-misconduct/)), which list a range of categories of academic misconduct and associated penalties, covering instances of academic misconduct (plagiarism, collusion, exam cheating).

**Assignment**

This assignment will be marked out of 100 and carries 60% of the overall module weighting. Your .java files and report for this part must be uploaded to WebLearn and submitted by 3pm on Friday 2nd of May 2025. The assignment must be carried out individually so you must not obtain help from anyone other than the module teaching staff. You must not copy code from any source apart from the module core text and the module WebLearn site. Collusion, plagiarism (unreferenced copying) and other forms of cheating constitute Academic Misconduct, which can lead to failure of the module and suspension from the University.

**Aim**

The aim of this assignment is to create a class to represent a **StaffHire**, together with two subclasses to represent Full-time Staff and a Part-time Staff respectively and GUI for a system that stores details of vacancy and hired staff details in the list. The class will contain a main method and will be tested using the command prompt. You will also need to write a report about your program. Your report should be no more than 2500 words in length (excluding the table of contents, class diagram, pseudocode and code listing.

**Deliverables**

When you are ready to submit your solution, upload your RecruitmentSystem.java file, together with the StaffHire.java, FullTimeStaff.java and PartTimeStaffHire.java files to the GitHub. Provide a link to your CS4001-CW-repository in the first page of your assignment. You will also have to upload your word report to the WebLearn when you are absolutely sure that you have uploaded the correct files, press the submit button. You can include you programme codes in the appendix of your report. (ask your tutor if you need help completing those classes to the specification of the first part of the coursework.

**Program**

**The program should consist of the following 4 classes.**

1. The StaffHire class has eight attributes, which correspond to the Vacancy number, Designation, Job Type (Job Type indicates permanent, contact, temporary job) and the staffName, joiningDate, qualification, appointedBy and joined. The Designation, Job Type staffName, joiningDate, qualification and appointedBy are represented as strings of text , vacancy number as a number and joined as a boolean.

The vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined are initialized in the constructor by being assigned values of the constructor's parameters. Each attribute has a corresponding accessor method(i.e. getter and setter methods for all attributes). A display method will output (suitably annotated) the vacancy number, designation and job type, staff Name, join date, qualification, appointed by, joined.

1. Vacancy Number - a whole number
2. Designation Type -a string characters
3. Job Type - a string characters
4. staffName - a string of characters
5. joiningDate - a string of characters
6. qualification - a string of characters
7. appointedBy - a string of characters
8. joined - either true or false (boolean)

There is a method to amend the joined status.

1. The **FullTimeStaffHire**  class is a subclass of the **StaffHire** class and has 2 attributes:

salary - a double number

weeklyFractionalHours - a whole number

The constructor accepts 10 parameters, which vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy, joined, salary and weeklyFractionalHours. A call is made to the superclass constructor with 8 parameters, vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy, joined. The salary and weeklyFractionalHours are assigned the corresponding parameter values.

Each new attribute has a corresponding accessor method.

A method is required to set the salary as changes to salary inevitably occur. The method accepts a new salary as a parameter and, if the staff has joined yet, the new value is assigned to the salary attribute. If the staff is not appointed, then a suitable message is output to the user indicating that that there is no staff appointed to set the salary.

A method is required to set the weeklyFractionalHours as changes to new working hour. The method accepts a new weeklyFractionalHours as a parameter and assigns the new value to the weeklyFractionalHours attribute.

A method to display the details of the staff is required. It must have the same signature as the display method in the StaffHire class. It will call the method in StaffHire class to display the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined. If the staff salary is set then the salary, weeklyFractionalHours should also be displayed. Each output must be suitably annotated.

1. The **PartTimeStaffHire** class is also a subclass of the **StaffHire** class and it has 4 attributes:

workingHour - a whole number

wagesPerHour -a double number

shifts - a string of characters

terminated - either true or false (boolean)

The constructor accepts 11 parameters, which are the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined, working hours (hours per day), wages per hour, and shifts (morning, day or evening working shifts). A call is made to the superclass constructor with 8 parameters, the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined. The working hours, wages per hour and shifts attributes are assigned the corresponding parameter values. Additionally, in the constructor, assign terminated status is assigned to false.

Each new attribute has a corresponding accessor method.

A method is required to set the working shifts as changes to shifts inevitably occur. The method accepts a new shifts as a parameter and assigns the new value to the shifts attribute if the staff has joined.

There is a method to terminate the staff. If the staff is already terminated, a suitable message is displayed. Else, staff name, join date, qualification, appointed by are set to an empty string. Similarly, the joined status is set to false while the terminated status is changed to true.

A method to display the details of the part time staff is required. It must have the same signature as the display method in the StaffHire class. It will call the method in StaffHire class to display the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined. If the working hours are set then wages per hour, working hour, shifts and terminated should also be displayed. Income per day is the product of wages per hour and working hour per day should also be displayed. Each output must be suitably annotated.

1. Create a new class within the project called RecruitmentSystem GUI class to obtain inputs. An example GUI is displayed below that can be redesigned to include all the required GUI components. A system that stores details of vacancy and hired staff details in the list. The class will contain a main method and will be tested using the command prompt. You will also need to write a report about your program.

Graphical user interface, application

Description automatically generated

***This is just an incomplete example; students need to improve or correct this GUI.***

**Field Input Methods**

Your GUI should contain all required components, and you are free to use a different layout if you feel that it improves the aesthetics, ease of use etc. The Recruitment class should store an arraylist (not an array) of type StaffHire to hold the FullTimeStaffHire and PartTimeStaffHire. There should be fields for entering: **(11 marks)**

1. Vacancy Number
2. Designation Type
3. Job Type
4. staffName
5. joiningDate
6. qualification
7. appointedBy
8. joined
9. salary
10. weeklyFractionalHours
11. workingHour
12. wagesPerHour
13. shifts
14. terminate number

* The display number, which is the same as the index of the staff in the arraylist

The input method for the display number is different from the input methods for the other text fields. The method first initialises the display number to -1. If the input from the user via the GUI is valid, the display number will be changed to a value that corresponds to a staffHire in the array list.

The input from the display number text field must therefore be tested using a try/catch statement to ensure that the staff number entered is an integer and, if it is, that it is in the correct range. If the value entered is an integer but is not in the correct range, a suitable error message is displayed using a message dialog box. If the display number entered is not an integer then an alternative error message is displayed using a message dialog box. The method will therefore either return –1 or a value that corresponds to a staff in the array list, depending on whether or not there was an error in the input. Any method that gets the display number should check its value and only use it if its value is not equal to –1.

**Buttons**

The GUI should have the following buttons: **(14 marks)**

1. **Add Full Time Staff**

When this button is pressed, the input values of the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy, joined, salary and weeklyFractionalHours are assigned the corresponding parameter values create a new object of type FullTimeStaffHire which is added to an array list of StaffHire class.

1. **Add Part Time Staff**

When this button is pressed, the input values of the call is made to the superclass constructor with 8 parameters, the vacancyNumber, designation, jobType, staffName, joiningDate, qualification, appointedBy and joined, working hours, wages per hour and shifts attributes are assigned the corresponding parameter values, create a new object of type PartTimeStaffHire which is added to an array list of StaffHire class.

1. **Set Salary - Full Time Staff**

The vacancy number and staff name are entered in the GUI. When the button (set Salary) is pressed, the input value of vacancy number is compared to the existing vacancy number, and if valid vacancy number has been entered, it is used to set salary for a vacancy from the list. The method set salary from the FullTimeStaffHire class is called here.

**Hint:** *An object of StaffHire is cast as FullTimeStaffHire*

1. **Set Working Shifts - Part Time Staff**

The vacancy number and staff name are entered in the GUI. When the button is pressed, the input value of vacancy number is compared to the existing vacancy number, and if valid vacancy number has been entered a method call is made to accepts a new shifts as a parameter and assigns the new value to the shifts attribute if the staff has joined. The method to setShifts from the PartTimeStaffHire class is called here.

**Hint:** *An object of StaffHire is cast as PartTimeStaffHire*

1. **Terminate Part Time Staff**

The vacancy number is entered in the GUI. When the button is pressed, the input value of the vacancy number is compared to the existing vacancy number in the list. If a valid value has been entered, it is used to terminate the appropriate part time staff from the array list of StaffHire. The method to terminate staff from the PartTimeStaffHire class is called here.

**Hint:** *An object of StaffHire is cast as PartTimeStaffHire*

1. **Display Number Button**

When this button is pressed, the information relating to the appropriate staff member is displayed.

1. **Clear (create a new button)**

When this button is pressed, the values from text fields are cleared.

**Marks**

Marks will be awarded as follows:

1. GitHub Link
2. A class diagram of all 4 classes
3. GUI
4. The actionPerformed method
5. The main method
6. Functionality of buttons
7. Reading input
8. Use of try/catch, checking input and displaying appropriate message dialogs
9. Program style (see http://www.bluej.org/objects-first/styleguide.html)

**Report**

The report should contain:

1. The GitHub link to your CW repository **(5%)**
2. A class diagram including RecruitmentSystem GUI class and other 3 classes showing the name of the class, the data types and names of the fields, and the return types and signatures of all of the methods. **(5%)**
3. A short description of each of your methods in StaffHire to hold the FullTimeStaffHire and PartTimeStaffHire **(9%)**
4. Pseudocode for the following button-handling methods: **(18 marks)**

* Add a vacancy for a full-time staff member
* Add a vacancy for a part-time staff member
* Set Salary full-time staff member
* Set Working shifts part-time staff member
* Terminate a part-time staff member
* Displaying a staff members in the arraylist

1. Textboxes, input check with try/catch **(9%)**
2. Buttons and Action Performed Methods **(14%)**
3. Test that the programme can be run in command prompt **(3%)**
4. GUI: display, add full-time, add part-time, display, set salary/set working shift Terminate **(10%)**
5. You should give evidence (through appropriate screenshots) of the following testing that you carried out on your program: **(7%)**

Test 1: Add a full-time staff member

Test 2: Add a part-time staff member

Test 3: Set Salary - full-time staff member

Test 4: Set Shifts - part-time staff member

Test 5: Terminate a part-time staff member

Test 6: Test that the program can be compiled and run using the command prompt, including a screenshot similar to Figure 1 in the command prompt learning aid. Additional features and advanced coding

Test 7: Test that appropriate information appears for the display button.

(Include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered.)

1. The report should contain a section on error detection and error correction where you give examples and evidence of three errors encountered in your implementation. The errors (syntax and/or runtime) should be distinctive and not of the same type. **(5%)**
2. The report should contain a conclusion, where you evaluate your work, reflecting on what you learnt from assignment. The report should include a title page, a table of contents (with page numbers), and a listing of the code (in an appendix). Marks will also be awarded for the quality of writing and the presentation of the report. Good program style, particularly naming, layout and documentation. See http://www.bluej.org/objects-first/styleguide.html for details. **(10%)**
3. Additional marks will be awarded for new features **(5%)**

|  |  |
| --- | --- |
| **A picture containing text  Description automatically generated**  **School of Computing and Digital Media**  **Marking Criteria for Coursework** | |
| **Marking criteria** | **Mark range** |
| An exceptional submission, with extensive and detailed knowledge based on a high level of additional background research; a high degree of critical analysis, evaluation and original insight; excellent organization and presentation. | 90 – 100 |
| In addition to the requirements for grades of 70-79% (below), an outstanding submission incorporating a high level of originality, depth and critical insight and going well beyond expected work. | 80 – 89 |
| An excellent submission, going beyond expected work; commanding understanding and appreciation of the central points; well-written and effectively structured; evidence-based, critical and logical analysis; comprehensive and correct referencing of sources. | 70 – 79 |
| A very good and comprehensive submission that fulfills the assignment brief; relevance and accuracy; clear structure and evidence-based; a sound grasp of the subject and ability to think about it effectively and critically; correct referencing of sources. | 60 – 69 |
| A solid submission that fulfills most of the assignment brief; adequate structure; mostly accurate, with few errors or omissions; some limitations in scope, critical thinking and argument; a consistent attempt at referencing sources. | 50 – 59 |
| A basic but incomplete submission, with limited relevant information; lacks logical and coherent structure, with some significant errors or omissions; contains sparse and/or irrelevant information and lacks an evidence-based approach; some limitations with referencing of sources; poor grammar and spelling. | 40 – 49 |
| An unsatisfactory submission, with some substantial errors, omissions or irrelevancies; barely acceptable amount of information relevant to the question; poor structure, presentation and expression and referencing of sources. | 30 – 39 |
| An unsatisfactory submission, that doesn’t meet the learning outcomes with many substantial errors, misconceptions, omissions or irrelevancies; little information relevant to the question; very poor structure, presentation and expression. | 1 – 29 |
| Non-submission or submission of work which cannot be given any credit (e.g., blank submission, incorrect assignment). | 0 |