School of Computer Science – Module Assessment Sheet for 2024-2025

Module	**Programming Paradigms** / COMP1029 (PDP) / Spring Semester
Module Convenor(s)	Doreen Ying Ying Sim

Assessment Name	COURSEWORK (Java Programming)	**Weight**	12%
Description of the Java programming developed system to be applied to fictitious large crowded public area or place, e.g. a public hospital	Develop a **Drink Serving Robotic System** for a fictitious large-scale restaurant in Malaysia using Java programming. Your system should attempt to tackle or control certain congestion and crowded situations in areas of a restaurant such as the waiting and dining areas for guests and/or customers to have meals at these five locations: **(1) Dining Foyer**; **(2) Main Dining Hall**; **(3) Dining Room One**; **(4) Dining Room Two**; and **(5) Family Dining Room**. Full details are provided in the coursework specification documented as below. The intended submission requirements and deliverable(s) of this coursework are: • A **report write-up**, mainly to include (i) a full description of the developed Java program by explaining the program coding with full comments, i.e. full explanation on the Java code ; (ii) overall experience (both practical and theoretical) in doing this coursework; noteworthy difficulties/challenges you faced while doing this coursework; and (iii) conclusion and discussions about doing this coursework (PDF file). • **Screen-shots** of all the significant outputs (PDF file). • All **program files** in (i) .working Java source code file (.java file), (ii) .class file (.class file), and/or (iii) probable .executable file (.exe file/application file) Submission is done by uploading all your files in **a compressed file type, i.e. a consolidated .zip file** electronically via Moodle. Please name your consolidated file in the file name of **studentID_Name_CW** while submitting via Moodle.		
Release Date	Tuesday 25-Mar-2025 (Week 30)		
Submission Date	Wednesday 30-Apr-2025 by 6pm (Week 35)		
Late Policy **(University of Nottingham default will apply, if blank)**	Work submitted after the deadline will be subject to a penalty of 5 marks (the standard 5% absolute) for each working day (out of the total 100 marks). Late submission deadline is 14 May 2025. Submissions after this date will only be accepted through the Extenuating Circumstances (EC) process.		
Feedback Mechanism and Date	Written individual feedback on this coursework will quite probably be done via Moodle. Feedback without marks may be returned tentatively by 15-May 2025 and marks tentatively to be returned on the 22-May 2025. Other mechanisms to return feedback could be voice recordings, annotations in submitted work or etc. Please note that the expectation is to return feedback within 15 working days if possible. However, no expectation to return feedback during the examination period. Marks should not be returned during the examination period.		
Details on the Coursework Specification	There is NO WORD LIMIT for the report write-up of this coursework, and there is NO PAGE LIMIT for the Java program coding and explanation/comments of this coursework. Tasks to be completed in this coursework are indicated in full details as below: **1. Your proposed ideas and concepts in Java program to develop the Drink Serving Robotic System are to ensure social distancing while a robot serves drinks in the above five(5) locations of a restaurant. This is also to ensure social and/or environmental comfort while customers and/or guests are having meals at the restaurant.** **Your Java program coding should have or suggested to have the following:** - Arrays; break statement and etc.; other structures and other data structures - Constructors; dot operator(s); AND OR, i.e. && and ||, and etc. - Classes, objects and instance variables		

- Loops such as for Nested-IF loops, WHILE-DO loops, WHILE loops and etc
- Control statements such as SWITCH, IF-ELSE, IF-THEN-ELSE and etc.
- **Random Number Generation (RNG)** (mandatory if possible)
- Inheritance; Method Overloading; Polymorphism
- Other features, methods and functions in Java programming

2. Problem Statements and Solutions Suggested

Problem I: In a congested area, there can be certain tragedies incurred (such as stampede, crowd crush and the like) if proper physical distancing in these crowded areas is not being taken into account. In this coursework, the maximum capacity of a spot/location in a dining area of a restaurant for people to be permitted to enter a certain area at a time should be calculated.

Problem II: Once a robot has entered in a spot/location of a restaurant, the physical distancing in between the robot and person/people around it should also be calculated and then to be determined as either the permitted or non-permitted comfortable physical distancing.

- **Suggested Solutions**

(1) A system (can be an app, through Social Internet of Things, i.e. SIoT) is needed to determine the maximum number of people to be allowed at one time to a certain area based on a standard distancing policy or social distancing guideline, i.e. 3 feet/1m physical distance (this is just a suggested distance guideline). This can be a class known as **StaticDrinkServing**.

(2) Once a robot has permitted to enter into the area of a restaurant, the app/ system should determine the dynamic distancing in between that robot and the surrounding person/people from four directions, i.e. LHS, RHS, Front and Back. This can be a class known as **DynamicDrinkServing**.

- **Suggested Solution Descriptions**

(1) You can create a class named as **RestrictedSpots** which stores at least **five locations** in a restaurant.

(2) Data associated with each spot, i.e. (a) Spot ID, (b) Spot Name, (c) Spot Area, (d) Spot Permitted Average Time per robot, and (e) Spot Maximum Capacity (abiding the social distancing rules and regulations, i.e. normal social distancing guidelines).

(3) Calculate maximum capacity based on the spot's area and the 3 feet or 1 meter social distancing guidelines - this is just a suggestion since you can develop your own program based on your own assumptions. The main concern is that your assumptions should make sense and your program should be functional, complete and running smoothly.

(4) You can use **Random Number Generator** to allocate number of current person/people/robot (i.e. in between 0 and maximum capacity of a restricted spot above).

(5) You can assume that each restricted spot is a free space and to make it more environmental-friendly and dynamic distance friendly, your app/system can take in as user-input, i.e. enter available space out of total spot area.

(6) The **StaticDrinkServing** class can have the main() function in your Java coding which attempts to ask user to key-in into which restricted spot in the restaurant that the robot would like to enter. Then, determine if the maximum capacity of the selected spot is reached or not and then gives permission or non-permission for entrance into the selected restricted spot. In summary, entrance permission or non-permission is based on the number of current

	person/people/robot in the selected restricted spot. If entrance is permitted, proceed to Steps 7 and 8 (for **DynamicDrinkServing** class), while if entrance is not permitted, proceed to Step 9. (7) If entrance is permitted to the selected spot, your app/system should ask the user to key-in the physical distance from other people in the immediate surrounding in four different directions, i.e. Left, Right, Front and Back. We assume that Bluetooth can implement this location distance sensing in the actual mobile app so that user can key-in to the program you developed. (8) If distances from all four directions are 3 feet /1 meter or more, display message 'you are safe in dynamic distancing!'. However, in this named class of **DynamicDrinkServing**, if any of the entered distance from any direction is less than 3 feet /1 meter, display message to advise the robot to move away 0.5 meter, or 1 meter (depending on how close is the entered distance by the user) from Left, or Right, or Front, or Back in order to practice Social or Physical Distancing policy and guideline. For example, if only from Left and Front, each distance is less than 3 feet/1m, ask the robot to move away from Left another 0.5m or 1m, and from Front another 0.5m or 1m. Finally, your system should alert the robot that it is in a close contact in that selected restricted spot. Therefore, set its status as casual contact and display its information as below: ● Robot's ID, Robot's Full Name, Selected Restricted Spot ID, Spot Name, Date, Time, Contact Status: Casual Contact **Note:** For the above Robot's ID & Robot's Full Name, please enter your own Nottingham Student ID and your own name. This is to show that you have tried your own Java program by keying in your particulars to the system you have developed. **Hint:** Dynamic distancing can help to determine close contact, distancing contact and casual contact. You can do the above without using **Inheritance** but there should be some places in your program where you can depict and deploy your knowledge of Inheritance logically and sensibly. As otherwise, **StaticDrinkServing** can inherit **DynamicDrinkServing** to determine the contact status of a robot. (9) If entrance is not permitted to the selected spot, your app/system should display the average time and ask the user if he or she wishes to wait for that long. If user's answer is 'Yes', let the robot enter the spot (we assume that the waiting time is reached or over and robot is allowed to enter to the spot); if user's answer is 'No', display the menu to allow user to choose if it wishes to visit another restricted spot within that hospital. **3. Documentation of your Ideas and Concepts in your report** (i) Describe fully if you have to make **any assumption(s)** while writing the report and developing the system mentioned as above (ii) A full description of your proposed opinions, ideas and concepts for your **Drink Serving Robotic System** (iii) Describe fully why you think the proposed ideas are important in terms of motivating you to implement them while developing the system.
Assessment Criteria	The break down of marks (out of **12 marks** in total) is shown as below: 1. A **report write-up** on your developed Java program to address all the criteria stated as above and full comments on your Java program coding. Extraordinary Features, Level of Sophistication and Professional Deployment of your developed Java system, program designs and ideas (**7 marks**) 2. A **functional** Java program involving switch statement, other associated or relevant structures/data structures, basic and application functions, control statements such as while do loops, for loops etc. (**5 marks**)

Important Notes	Please do not hold back your innovation and intuitiveness just because you find it difficult to implement (or for any other reason) certain features and/or functions in your Java program. This is your chance to take a step towards contributing to the society by being innovative and try to be able to deal with real-world daily routine tasks. You might take it further even after your studies. The following suggested areas will be taken into account for each part of the assessment for each coursework as assigned (wherever applicable): ● Innovation, Comprehensiveness and Novelty of the developed Java program to the fictitious case study hospital in Malaysia ● Level of Completeness, Sophistication and Comprehensiveness of the developed Java program ● Impact and significance of the daily routine tasks and problem(s) as well as your attempt to the solution(s) and ways of handling and enhancing the daily regular routine tasks. ● Demonstration of the knowledge of the area ● Quality of the concept, including appropriateness and novelty ● Quality of the technological design, including appropriate use of software design concepts, and appropriate good coding practice (abstraction, commenting, naming, and the like) ● Quality of the realization, including how well it works and elaborations over and above the basic requirements
Plagiarism	Use of third party assets (tutorials, images, example codes, libraries etc.) MUST be credited and referenced, and you MUST be able to demonstrate that they are available under a license that allows their reuse. Making significant use of tutorial code while referencing it is poor academic practice, and will result in a lower mark that reflects the significance of your own original contribution. Copying code from other students, from previous students, from any other source, or soliciting code from online sources and submitting it as your own is plagiarism and will be penalized as such. FAILURE TO ATTRIBUTE a source will result in a mark of zero - and can potentially result in failure of coursework, module, or degree. All submissions are checked using the plagiarism detection software and/or manually for signs of cheating.