Assessment Proforma 2024-25

Key Information

Module Code	CM1210
Module Title	Object Oriented Java Programming
Module Leader	Dr Matt Morgan
Module Moderator	Ramalakshmi Vaidhiyanathan
Assessment Title	Java implementation Skills
Assessment Number	1 of 2
Assessment Weighting	50%
Assessment Limits	PDF report should NOT exceed TWO pages in length.

The Assessment Calendar can be found under ‘Assessment & Feedback’ in the COMSC-ORG-SCHOOL organisation on Learning Central. This is the single point of truth for (a) the hand out date and time, (b) the hand in date and time, and (c) the feedback return date for all assessments.

Learning Outcomes

The learning outcomes for this assessment are as follows:

1. Develop a Object-Oriented program that has input and output functionality and that is event driven.
2. Show fluency in selecting and using basic components in the Java language.

Submission Instructions

The coversheet can be found under ‘Assessment & Feedback’ in the COMSC-ORG-SCHOOL organisation on Learning Central.

All files should be submitted via Learning Central. The submission page can be found under ‘Assessment & Feedback’ in the CM1210 module on Learning Central. Your submission should consist of multiple files:

Description		**Type**	**Name**
Coversheet	Compulsory	One PDF (.pdf) file	Coversheet.pdf
Q1a	**Compulsory**	One ZIP (.zip) archive containing JAVA (.java) source code file(s)	Q1a_[student number].zip
Q1b	**Compulsory**	One ZIP (.zip) archive containing JAVA (.java) source code file(s)	Q1b_[student number].zip
Q2	**Compulsory**	One PDF (.pdf) containing a report, no more than 2 pages in length.	Q2_[student number].pdf

If you are unable to submit your work due to technical difficulties, please submit your work via e-mail to comsc-submissions@cardiff.ac.uk and notify the module leader.

Assessment Description

1. Any n x n magic square (where n is an odd integer) consists of an n x n matrix whose elements contain the numbers 1,2,3,….,n^2 such that the sum of each row, column and diagonal is equal to $\frac{n(n^2+1)}{2}$. For example, the following magix square for $\mathrm{n} = 3$, with the sum of each row, column and diagonal being $\frac{3(3^2+1)}{2} = 15$:

6	1	8
7	5	3
2	9	4

a) An algorithm for generating an n x n magic square for an odd n is as follows:

NOTE: Assume the rows and columns wrap around (i.e. moving one column left from the first column gives the last column)

```
Create a 2-dimensional array of size n x n and set all values to be 0
Set $x = 1$, $y = \frac{n+1}{2}$ (row 1 and column $\frac{n+1}{2}$)
Insert 1 at $x, y$
for $i = 2$ to $n^2$ do
    if element $x - 1$,$y - 1$ is empty (i.e. $= 0$) then
        $x = x - 1, y = y - 1$
    else
        $x = x + 1, y = y$
    end if
    Insert $i$ at $x, y$
end for
```

Write a **command line application** that prompts the user for an **odd** integer and displays a magic square of that size to standard output (i.e. the command line).

[**HINTS:** *Recall that Java arrays start at the element 0; it may help to define a class to store a square matrix*].

(12 Marks)

[Requirements Fulfilment: 4, Design and Structure: 5, Usability and Error Handling: 2, Presentation: 1]

b) Write a **command line** game with the following functionality:

- The application should prompt the user for an odd integer and create a magic square of that size.
- The magic square should then be shuffled by repeatedly (for n^2 times) choosing a random element and swapping it with a random neighbour (**not** including diagonals).
- The shuffled square should be displayed to the user, who must attempt to reconstruct a magic square.
- The user makes moves by giving input of the form:

$$i\ j\ direction$$

where i and j specify the row and column of an element to be swapped, and $direction$ (either U, D ,L ,R representing *up*, *down*, *left* and *right*) specifies which direction it should be swapped with. For example, the move 2 1 D applied to the square above would give:

6	1	8
2	5	3
7	9	4

On completion, the game should report the number of moves made.

(18 Marks)

[Requirements Fulfilment: 6, Design and Structure: 6, Usability and Error Handling: 5, Presentation: 1]

HINT: MULTIDIMENSIONAL ARRAYS

The code below gives an example of using a 2-dimensional array to store values:

```
public class MultiArrayTest {
   public static void main( String[] args )
   {
      int a[][] = {{1,2,3}, {4,5,6}};
      System.out.println( "length of a is " + a.length );
      for (int i = 0; i < a.length; i++)
      {
         for (int j = 0; j < a[i].length; j++)
         {
            System.out.print( a[i][j] + " " );
         }
         System.out.println(); }
      }
   }
}
```

2. Write a **report** that explains what you've done and how you did it. Your report should have **four sections**:
   a) a brief overview of the assignment and your goals in completing it.
   b) a description of your solution design, including any assumptions made, algorithms used, etc.
   c) a discussion of your completed solution to the assignment, including the scope of solution, quality of solution, interesting results, difficulties overcome, enhancements delivered, etc.,
   d) a discussion of your software test methodology. How did you ensure that your solution does what it's meant to do.

Your report does not have to be long. On the contrary, you should strive for conciseness and clarity in your writing. However, details of page limits are given in the "Submission Instructions" section at the end of this document.

(20 Marks)

[For mark guidance see "Criteria for Assessment" below]

[Total Marks: 50]

Assessment Criteria

	Requirement Fulfilment	Design and Structure	Usability and Error Handling	Presentation	Report
High 1st 80%+	Initiative shown- creativity. All of tasks completed.	Application shows superb adherence to structure: well thought through use of variables types, methods, classes, and inheritance, use of advanced OOP concepts such as Inheritance, Interfaces, Abstract classes, where appropriate. The design demonstrates an excellent understanding of OOP.	Error handling is complete. The application provides a fantastic user experience.	All outputs are presented and formatted well.	Superb written report, contains all sections and gives superb insight for all sections.
1st 70-79%	Completion of all tasks	Application shows excellent adherence to structure: well thought through use of variables types, methods, classes, and inheritance, use of advanced OOP concepts such as Inheritance, Interfaces, Abstract classes, where appropriate. The type of modifier is appropriate for all methods and variables. Written description points out very clearly how the code exhibits these qualities.	Error handling is complete. The application provides an excellent user experience.	All outputs are presented and formatted well.	Excellent written report, contains all sections and gives excellent insight for all sections.
2.i 60-69%	Some completion : more than a third of the tasks.	Application shows very good adherence to structure: well thought through use of variables types, methods, classes, and inheritance, use of advanced OOP concepts such as Inheritance, Interfaces, Abstract classes, where appropriate. The type of modifier is appropriate for all methods and variables. Written description points out very clearly how the code exhibits these qualities.	Error handling is partially complete. The application provides a very good user experience.	Shows a lot of thought given to presentation of outputs.	Very well written report, contains all sections and gives good insight for some to most sections.
2.ii 50-59%	Some completion : more than a half of the tasks.	Project shows good adherence to structure: correct use of variables types, methods and classes, tentative exploration of advanced concepts of OOP such as Inheritance ,Interfaces and Abstract classes, where appropriate. The type of modifier is appropriate for most of the methods and variables. Good written description of how code exhibits these qualities included.	Error handling is partially complete. The application provides a good user experience.	Shows some thought to presentation of outputs.	Well written report, contains all sections and gives good insight for some to most sections.

3rd 40-49%	Some completion: more than a third of the tasks.	Project shows some adherence to structure: correct use of variable types, methods and classes. Some written description of how code exhibits these qualities included. The type of modifier is appropriate for some of the methods and variables.	Error handling is poor or does not exist. The application provides a weak user experience.	Shows some thought to presentation of outputs, but lots of scope of improvement.	Generally well written report, contains most sections and gives some insight.
Marginal Fail 30-39%	Limited task completion or requirements not met.	Project shows very little adherence to structure: very poor use of variable types, methods and classes. No written description of how code exhibits these qualities included. The selection of inappropriate visibility of the classes or class members. The type of modifier is not appropriate.	Error handling does not exist. The application provides an extremely weak user experience.	Presentation of outputs have not been considered.	Poorly written, contains a few sections, lacks information requested.
Fail 0-29%	No task completion or requirements not met.	Application shows no adherence to structure: extremely poor use of variable types, methods and classes. No written description of how code exhibits these qualities included. The selection of inappropriate visibility of the classes or class members. The type of modifier is not appropriate.	There is no Error handling present. The application provides an extremely weak user experience.	Presentation of outputs have not been considered.	Extremely poorly written or no report submitted

Help and Support

Help and support will be available at the weekly CM1210 lab sessions.

Feedback

Feedback on your coursework will address the above assessment criteria. Feedback and marks will be returned by Thursday 17th April 2025 via Learning Central. Further feedback will be available through 1-1 or group feedback sessions, details of which will be made available after the coursework return date.

Feedback from this assignment will be useful for future modules and assessments.