SCHOOL OF COMPUTER SCIENCE
COURSEWORK ASSESSMENT PROFORMA

MODULE & LECTURER: CM1210 - DR MATT MORGAN

DATE SET: FRIDAY 2ND MARCH 2018

SUBMISSION DATE: FRIDAY 20TH APRIL 2018 at 9:30am

SUBMISSION ARRANGEMENTS: SEE "SUBMISSION INSTRUCTIONS" BELOW

TITLE: RESIT ASSESSMENT

This coursework is worth 50% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet.

INSTRUCTIONS

Answer all of the attached questions.

SUBMISSION INSTRUCTIONS

You must submit to Learning Central a zip archive (named using your student number) which contains the following files:

Description		Type	Name
Cover sheet	**Compulsory**	One PDF (.pdf) file	[Student number].pdf
Q1	**Compulsory**	One PDF (.pdf) containing screen shots showing an example of the output of each application in question 1. You should only provide ONE screen shot for each.	Q1_[Student number].pdf
	Compulsory	A zip archive containing one or more Java source file(s) (.java) containing your answers to question 1. Each source code file should contain your name and student number in a comment.	Q1_[Student number].zip
Q2	**Compulsory**	One PDF (.pdf) containing screen shots showing an example of the output of each application in question 2. You should only provide ONE screen shot for each.	Q2_[Student number].pdf
	Compulsory	A zip archive containing one or more Java source file(s) (.java) containing your answers to question 2. Each source code file should contain your name and student number in a comment.	Q2_[Student number].zip
Q3	**Compulsory**	One PDF (.pdf) containing a written justification for your design and an explanation of what makes it extensible.	Q3_[Student number].pdf
	Compulsory	A zip archive containing one or more Java source file(s) (.java) containing your answers to question 3. Each source code file should contain your name and student number in a comment.	Q3_[Student number].zip

CODE REUSE

Your solutions may make use of any classes in the Core Java API. You may also reproduce small pieces of code from:

- The CM1210 course handouts and solutions
- java.oracle.com
- any textbooks

provided:

- The section reproduced does not form the entire solution to a single question
- The source of the code is **clearly referenced** in your source code
- Your code is commented to demonstrate clearly that you understand how the reproduced code works (i.e., explain why types have been selected, why other language features have been used, etc.)

You may **NOT** reproduce code written by any other student or code downloaded from any other website.

If you are in any doubt about whether you may include a piece of code that you have not written yourself, ask the lecturer before submitting.

CRITERIA FOR ASSESSMENT

Credit will be awarded against the following criteria:

Functionality

- To what extent does the program perform the task(s) required by the question.

Design

- How well designed is the code, particularly with respect to the ease with which it may be maintained or extended. In particular, consideration will be given to:
- Use of appropriate types, program control structures and classes from the core API.
- Definition of appropriate classes, interfaces and methods.

Ease of use

- Formatting of input/output.

- Interaction with the user.
- How does the code deal with invalid user input? Will the applications crash for certain data?

Documentation and presentation

- Clear and appropriate screenshots.
- Appropriate use of comments.
- Readability of code.

[Questions on Next Page]

Questions

1. Write a **command line application** to store and retrieve student details. Each entry should consist of the following data:

Field	**Constraints**	**Examples (comma separated)**
Name	Only letters/spaces allowed, at least one character	matt, Michael Bolt
Student Number	Upper case "C" followed by 6 digits	C123456, C078925
Course Name	Only letters/spaces allowed, at least one character	Computer Science, maths
Course ID	2 upper case letters, 4 digits	CM2536, MM6846
House Number	At least one digit followed by at most one letter	3, 16, 123a
Street Name	Only letters/spaces allowed, at least one character	cardinal avenue, The Strand
Town	Only letters/spaces allowed, at least one character	bristol, Edinburgh
Postcode	2 upper case letters, 1 digit, 2 upper case letters	BM0PQ, MC9SL

NOTE: Each data field should adhere to the constraints stated before being accepted as input into the application, e.g. for postcode, valid entries are those that are made up of a sequence of 2 upper case letters followed by 1 digit followed by 2 upper case letters. Any other entries that do not match this sequence for postcodes, are invalid and should be rejected as input into the application.

(a) The application should have the ability to:

- load and save all student details to file (in binary/text format), i.e. your system should allow both binary AND text format files to be used;
- create a new (empty) file to store student details;
- add new student entries;
- display all student details to screen;
- display all student details whose course name contains a specified substring.

(12 Marks)
[Functionality: 6, Design: 3, Ease of use: 2, Presentation: 1]

(b) Add the extra functionality:

- delete student entries (specified by their position in the list);
- find all student addresses that contain a given substring in any of the data fields: House Number, Street Name, Town and Postcode;
- display a specified subset of the student details to screen (for example, entries 2 to 7).

(8 Marks)
[Functionality: 4, Design: 2, Ease of use: 1, Presentation: 1

[Please Turn Over - Question 2 on Next Page]

2. Any $n \times n$ magic square (where n is an odd integer) consists of an $n \times n$ 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 magic square for $n = 3$, 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 $n \times n$ magic square for 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 $n$ by $n$ 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).

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

(8 Marks)
[Functionality: 3, Design: 3, Ease of use: 1, 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.

[Please Turn Over - Question 2 Continues on Next Page]

- 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.

(12 Marks)
[Functionality: 5, Design: 4, Ease of use: 2, Presentation: 1

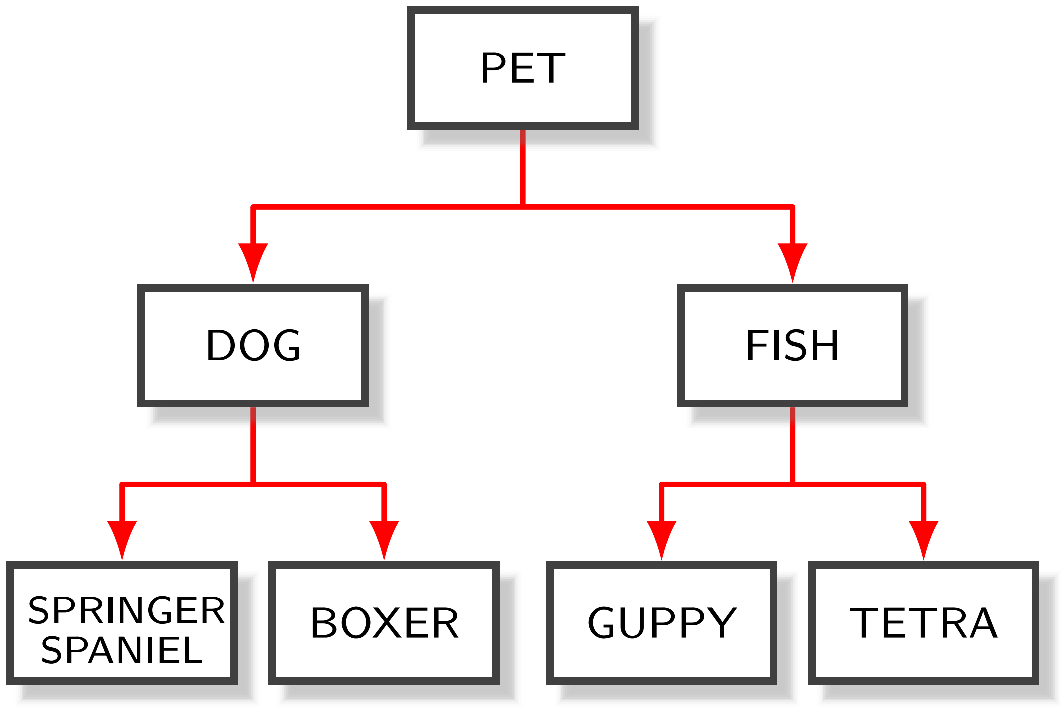
HINT: MULTIDIMENSIONAL ARRAYS

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

```java
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();
      }
   }
}
```

[Please Turn Over - Question 3 on Next Page]

3. Consider the hierarchical representation of Pets in the following diagram:



Design and implement classes for PET, DOG, FISH, SPRINGER SPANIEL, BOXER, GUPPY and TETRA, that maximise the concepts of Inheritance and Polymorphism in Java. Your classes should be designed in a such a way that new classes can be easily produced, making them extensible, e.g. add further DOGS and FISH, or further PETS, such as CATS, SNAKES, etc.. Classes should include relevant constructor, accessor and mutator methods, where appropriate.

(10 Marks)

[Functionality: 4, Design: 6, Ease of use: 0, Presentation: 0

[End of Questions]