Cardiff School of Computer Science and Informatics
Coursework Assessment Pro-forma

MODULE & LECTURER: CM1210

MODULE TITLE: OBJECT ORIENTED JAVA PROGRAMMING

LECTURER: DR MATT MORGAN

ASSESSMENT TITLE: JAVA IMPLEMENTATION SKILLS

ASSESSMENT NUMBER: 1

DATE SET: FRIDAY 28TH FEBRUARY 2020

SUBMISSION DATE and TIME: FRIDAY 27TH MARCH 2020 at 9:30am

RETURN DATE: MONDAY 20TH APRIL 2020

This assignment is worth 50% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:

https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf

INSTRUCTIONS

Answer all of the attached questions.

SUBMISSION INSTRUCTIONS

You should submit you completed coursework under the "Assessment" tab in the CM1210 module on Learning Central. Your submission should 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 a written justification for your design and an explanation of what makes it extensible.	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

Any code submitted will be run on a system equivalent to those available in the Windows laboratory and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) will result in a reduction in marks for that question part of 15%.

Staff reserve the right to invite students to a meeting to discuss coursework submissions

Assignment

1. 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*].

   (16 Marks)
   [Functionality: 6, Design: 6, Ease of use: 2, Presentation: 2

   (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 1 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.

(24 Marks)

[Functionality: 10, Design: 8, Ease of use: 4, Presentation: 2

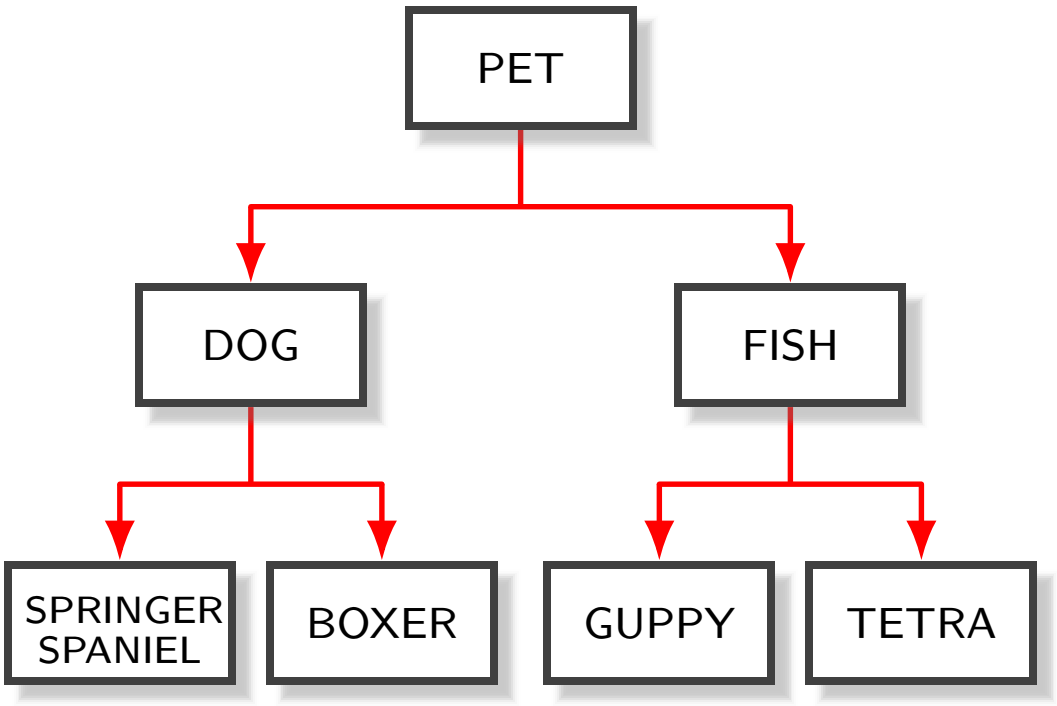
HINT: MULTIDIMENSIONAL ARRAYS

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. 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]

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.

FEEDBACK AND SUGGESTION FOR FUTURE LEARNING

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on Monday 20th April 2020 via Learning Central.