SCHOOL OF COMPUTER SCIENCE COURSEWORK ASSESSMENT PROFORMA

MODULE & LECTURER: CM1210 - DR MATT MORGAN

DATE SET: FRIDAY 1ST MARCH 2019

SUBMISSION DATE: FRIDAY 12TH APRIL 2019 at 9:30am

SUBMISSION ARRANGEMENTS: SEE "SUBMISSION INSTRUCTIONS" BELOW

TITLE: CM1210 ASSESSMENT 1

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 via the CM1210 module on Learning Central a zip archive (named using your student number, i.e. CXXXXXX.zip) 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.	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 your application in question 2.	Q2_[Student number].pdf
	Compulsory	A zip archive containing one or more Java source file(s) (.java) containing your solution 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.	Q3_[Student number].pdf
	Compulsory	A zip archive containing one or more Java source file(s) (.java) containing your implementation for 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 rugby player details. Each entry should consist of the following data:

Field	**Constraints**	**Examples (comma separated)**
Player Name	Only letters/spaces allowed, must include forename AND surname separated by a space	George North, jonathan sexton
Player ID	Uppercase "RFU" followed by 5 digits	RFU00005, RFU05674, RFU26948
Career Tries Scored	At least one digit	0, 33, 108
Team Name	Only letters/spaces allowed, at least one character	Cardiff Blues, Bath
Team ID	3 Uppercase letters followed by 4 digits	CDB2456, BTH0739
Home Stadium Name	Only letters/spaces allowed, at least one character	Cardiff Arms Park, Recreation Ground
Home Stadium Street	Only digits/letters/spaces allowed, at least one character	3 cardinal avenue, The Strand
Home Stadium Town	Only letters/spaces allowed, at least one character	bristol, Edinburgh
Home Stadium Postcode	1 upper case letter, 1 digit, 3 upper case letters	B0PQW, M9SLD

NOTE: Each data field should adhere to the constraints stated before being accepted as input into the application, e.g. for Home Stadium Postcode, valid entries are those that are made up of a sequence of 1 upper case letter followed by 1 digit followed by 3 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 rugby player details to file (text format ONLY, i.e. NOT Binary files);
- create a new (empty) file to store rugby player details;
- add new rugby player entries;
- display all rugby player details to screen;
- display all rugby player details whose team name contains a specified substring. **NOTE:** Your substring search should be case sensitive.

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

(b) Add the extra functionality:
- delete rugby player entries (specified by their position in the list);
- find all stadium addresses that contain a given substring in any of the data fields: Home Stadium Name, Home Stadium Street , Home Stadium Town and Home Stadium Postcode. **NOTE:** Your search should be case insensitive;
- display a specified subset of the rugby players to screen (for example, entries 2 to 7).

(10 Marks)
[Functionality: 5, Design: 3, Ease of use: 1, Presentation: 1

[Please Turn Over - Question 2 on Next Page]

2. Write a Java program `Encrypt` that reads a Binary file called `Secret`, containing multiple lines of text, and prints it on the terminal with each lowercase letter cyclically replaced by the previous one, i.e. 'b' becomes 'a', 'c' becomes 'b', ... , and 'a' becomes 'z'. Any punctuation, uppercase letters, or other non-alphabetic characters should be printed unchanged. Your program should include appropriate error handling. The following skeleton program is provided as a starting point:

```
import java.io.*;
public class Encrypt
{
    public static void main(String [] args)
    {
        if (args.length != 1)
        {
            System.err.println("One argument expected!");
            System.exit(1);
        }

        // TODO: Complete the program
    }
}
```

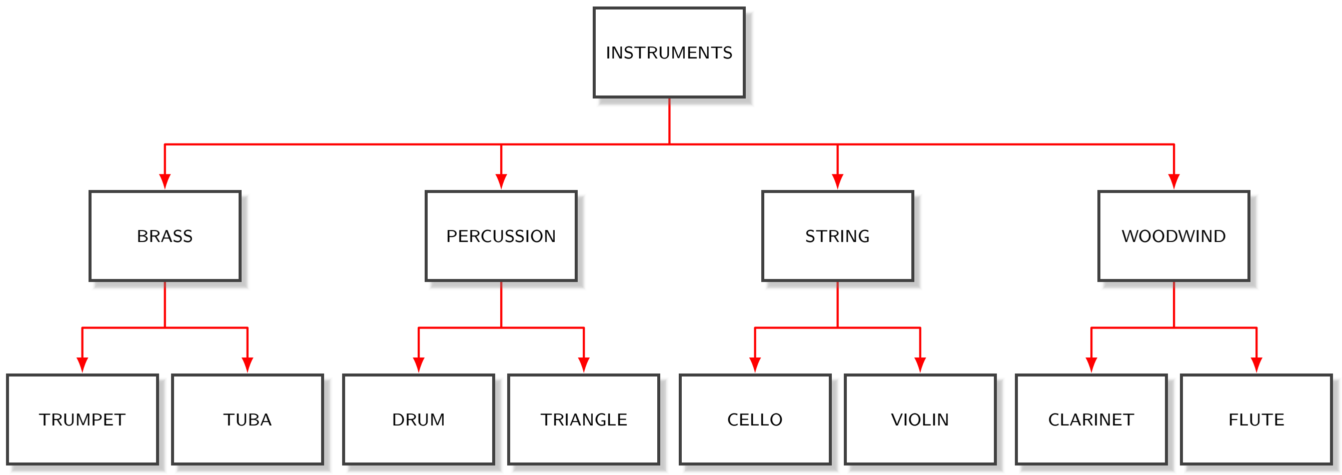
NOTE: There are a number of different classes available in `java.io` that you would use in your application. You should spend some time looking through the Java API to decide which of those classes you think would be most suitable.

(10 Marks)

[Functionality: 5, Design: 3, Ease of use: 1, Presentation: 1

[Please Turn Over - Question 3 on Next Page]

3. Consider the hierarchical representation of `INSTRUMENTS` in the following diagram:



Design and implement the `INSTRUMENTS` hierarchy shown in the above diagram. Your design/implementation should maximise the use of the concepts of Inheritance/Polymorphism in Java, to avoid duplication and reduce redundancy. Remember that separating out all the common variables and methods into the higher level classes, and leaving the specialised variables and methods in the lower level classes, allows redundancy to be greatly reduced or even eliminated. Further note that when you create your classes, if there is an existing class that includes some of the code your want, you can derive your new class from the existing one. Your classes should ideally be designed in such a way that new classes can be easily produced, making them extensible, e.g. add further BRASS instruments. Classes should ONLY include relevant constructor, accessor and mutator methods, if APPROPRIATE.

You should include with a submission a short written justification of your design, to be no more than two A4 pages in length.

(15 Marks)

[Functionality: 4, Design: 10, Ease of use: 0, Presentation: 1

[ASSESSMENT TOTAL - 50 Marks]

[End of Questions]