University of Newcastle
School of Electrical Engineering and Computing

SENG6110 Programming Assignment 2 – Trimester 1, 2020

Due (electronic submission on Blackboard) by 11:59pm on Monday 27 April 2020.
Worth: 20%

Introduction
The objective of this assignment is to extend the implementation of Assignment 1 using **arrays** and **external files**.

This assignment must be completed individually.

Before you start
Carefully read the specification below. Make sure you have all the information necessary to start writing the program. If you are uncertain of something, do not make assumptions. Post your questions to the discussion board forum named "Assignment 2" and check it regularly.

Additional resources on Blackboard which you can use as a starting point:

- Three .java files. Modify these files to answer the questions below. DO NOT re-name them or create additional .java files.
- A guide to answering the assignment is provided, giving additional explanation and help for each question.

Specifications
Create a database which will hold up to 5 animal family objects. Each family may have up to 3 animals in it – see the sample data in the Appendix at the end of this document. To implement this you should use arrays, one array of Family objects each with its own array of Animals.

That is, create an array of up to 5 Family objects. Each Family object can have an array of up to 3 Animal objects as one of its attributes.

The program should have the functionalities listed below.

Question 1. A user may add a Family (15 marks).
The user will specify the Family name, continent and diet. There should be an error message if the Family name already exists, or if there are already 5 Family objects.

Question 2. A user may add an Animal object to one of the Family objects (15 marks).
The user will specify the Family name (the object to which this Animal is being added), Animal name, continent and habitat. There should be an error message if the Animal already exists, or if the Family does not exist, or the Family already has 3 Animal objects.

Question 3. A user may delete a Family (15 marks).
The user will specify the Family name. There should be an error message if the Family does not exist. This will also cause any Animal objects in that Family to be deleted.

Question 4. A user may delete an Animal (15 marks).
The user will specify the Animals name. There should be an error message if the Animal does not exist. If there exists a Family containing that Animal, the Family should be updated.

Question 5. A user may ask for a list of all objects (15 marks).
The output should list all families and animals with all attributes included. If there are no objects, the output should be: "No family objects exist". Otherwise, one line should be output for each object with one of these formats:
`FAMILY scientificName commonName diet`
`ANIMAL (Family)name name continent habitat`

Question 6. A user may ask for a list of animals that belong a family (5 marks).
The user will specify the Family name. If the Family contains no Animal objects, the output should be: "No animals contained in this family." Otherwise, one line should be output for each animal, with this format:
`ANIMAL (Family)name name continent habitat`

Question 7. A user may ask for a list of animals living on the same continent (5 marks).
The user specifies the continent. There should be an error message if there are no animals in the database living on the specified continent. Otherwise the output should be in the format:
`ANIMAL (Family)name name continent`

Question 8. A user may ask for a list of animals with the same diet as a specified animal (5 marks).
The user specifies the *animal* (not the diet specifically). If there are no other animals with that diet an error message should be shown, otherwise the output should be in the format:
`ANIMAL (Family)name name continent diet`

Question 9. A user may import family and animal information from a file (5 marks).
The user will specify the name of the file to be read from. The program should expect a file consisting of zero or more lines, each of which should have one of these forms:

`FAMILY scientificName commonName diet`
eg: `FAMILY Hominidae Ape Omnivore`

`ANIMAL (Family)name name continent habitat`
eg: `ANIMAL Ape Orang-Utan Asia Forest`

The imported information should replace any pre-existing families and animals. A FAMILY line should cause addition of the described Family object. Similarly, an ANIMAL line should cause addition of the described Animal. An error message should be output if the file is unable to be opened, or if a problem occurred during reading. This includes cases such as finding a line with an invalid format and being unable to add a family or animal due to one of the issues listed under functionalities 1 and 2 (in both cases, the program should continue to process the rest of the file). An example of a file which could be used is included as `input_data.txt`.

Question 10. A user may write family and animal information to file (5 marks).
The user will specify the name of the file to create. The created file should have the same format as that used for importing. An error message should be output if the file is unable to be created, or if a problem occurred during writing.

Question 11. Additional challenge (10 additional marks).
For functionalities 5 and 6, the output must be alphabetically sorted using the object names.

You must implement the sorting algorithm yourself (do not use an existing sort method in a Java library but rather write your own). Extra material about sorting algorithms, including lecture slides and a video, are available in Blackboard.

Program Requirements

In this assignment you should use arrays to store the Family and Animal objects, and there are also two constants specifying array sizes.

The program should consist of three classes (**and only three classes**), with the following names and instance variables:

Class Interface - provides (TIO) the user interface

- family[] - an array of Family objects managed by the program.
- MAX_FAMILIES - static final int - the max number of families managed.

Class Family - contains the attributes of each family

- Family_name - the scientific name of the family
- Common_Name – the common name of the family
- Diet - the diet type
- animal[] - an array of Animal objects contained in the object
- MAX_ANIMALS - static final int - the max number of animals in a family.

Class Animal - contains the attributes of each animal

- Animal_name - the name of the animal
- Continent - the continent where the animal is found
- Habitat - the habitat type

All instance variables of your classes should be private (this is imposed so that you apply the principles of encapsulation).

Your classes will need methods to provide the required functionalities. **The only class which should have a main() method is Interface.java**, which should create an instance of the class Interface, and call a method run(), which will display the menu to the user. This class will be the only one that takes input from and sends output to the user. It may do so using either TIO or GUI methods (your choice). A template is shown below.

```
public class Interface {
    private void run(){
        // All your code here ...
    }

        // Create an instance of this class and run the code above
        public static void main(String[] args){
        Interface infFace = new Interface();
        intFace.run();
    }
}
```

Alternatively you may simply use the .java files provided as a starting point, including the menu code provided.

Marks will be awarded for layout (including visual aspects (variable names, indentation) and structural aspects (variable scope, method usage)), documentation (comments), and the submission's ability to perform as specified. A more detailed marking schema will be made available on Blackboard.

What to submit

You should submit only the three .java files (Family.java, Animal.java and Interface.java) in a compressed .zip file, via the "Assignment 2" link on Blackboard. Do not include .class files in your submission.

Make sure that your name is in a comment block at the beginning of each .java file.

Late Penalty and adverse circumstances

Note that your mark will be reduced by 10% for each day (or part day) that the assignment is late. This applies equally to week and weekend days. You are entitled to apply for special consideration if adverse circumstances have had an impact on your performance in an assessment item. This includes applying for an extension of time to complete an assessment item. See https://www.newcastle.edu.au/current-students/learning/assessments-and-exams/adverse-circumstances for more details.

Appendix – sample data to use when testing your code

The data

Family_name	Common_Name	Diet	Animal_name	Continent	Habitat
Hominidae	Ape	Omnivore	Orang-Utan	Asia	Forest
Hominidae	Ape	Omnivore	Gorilla	Africa	Mixed_Forest
Hominidae	Ape	Omnivore	Chimpanzee	Africa	Forest
Felidae	Cat	Carnivore	Tiger	Asia	Mixed_Forest
Felidae	Cat	Carnivore	Lion	Africa	Grasslands
Felidae	Cat	Carnivore	Cheetah	Africa	Grasslands
Canidae	Dog	Carnivore	Wolf	Europe	Grasslands
Canidae	Dog	Carnivore	Fennec_fox	Asia	Desert
Canidae	Dog	Carnivore	African_wild_dog	Africa	Grasslands
Chiroptera	Bat	Vegetarian	Flying_fox	Australia	Mixed_Forest
Chiroptera	Bat	Vegetarian	Short-tailed bat	New_Zealand	Forest
Chiroptera	Bat	Vegetarian	Hog-nosed_bat	Asia	Forest
Cervidae	Deer	Vegetarian	Elk	North_America	Grasslands
Cervidae	Deer	Vegetarian	Moose	North_America	Mixed_Forest
Cervidae	Deer	Vegetarian	Musk_deer	Asia	Mixed_Forest

Family data file entries

```
FAMILY Hominidae  Ape Omnivore
FAMILY Felidae Cat Carnivore
FAMILY Canidae Dog Carnivore
FAMILY Chiroptera Bat Vegetarian
FAMILY Cervidae Deer Vegetarian
```

Animal data file entries

```
ANIMAL Hominidae Orang-Utan Asia Forest
ANIMAL Hominidae Gorilla Africa Mixed_Forest
ANIMAL Hominidae Chimpanzee Africa Forest
ANIMAL Felidae Tiger Asia Mixed_Forest
ANIMAL Felidae Lion Africa Grasslands
ANIMAL Felidae Cheetah Africa Grasslands
ANIMAL Canidae Wolf Europe Grasslands
ANIMAL Canidae Fennec_fox Asia Desert
ANIMAL Canidae African_wild_dog Africa Grasslands
ANIMAL Chiroptera Flying_fox Australia Mixed_Forest
ANIMAL Chiroptera Short-tailed bat New_Zealand Forest
ANIMAL Chiroptera Hog-nosed_bat Asia Forest
ANIMAL Cervidae Elk North_America Grasslands
ANIMAL Cervidae Moose North_America Mixed_Forest
ANIMAL Cervidae Musk_deer Asia Mixed_Forest
```

Sources of the animal information

https://en.wikipedia.org/wiki/Mammal_classification
https://en.wikipedia.org/wiki/List_of_mammal_genera