Assignment 2

Due Date

The assignment is due at 11:55PM Wednesday October 2nd 2024. It is worth 24% of your mark and you should complete it as an individual.

Background

The Olympic Games, the pinnacle of world sport. Athletes from different countries compete in events with the first-placed athlete winning a Gold medal, second-placed winning Silver, and third-placed winning Bronze.

The International Olympic Committee has asked you to develop an app to manage the results for all athletes, across all events, over the 31 competitions of the Summer Olympiad. You will need to store all medallists, and then, for a specified country, you will need to determine that country's most successful year, find the most successful country for that year, and produce an athlete-based medal tally for that year.

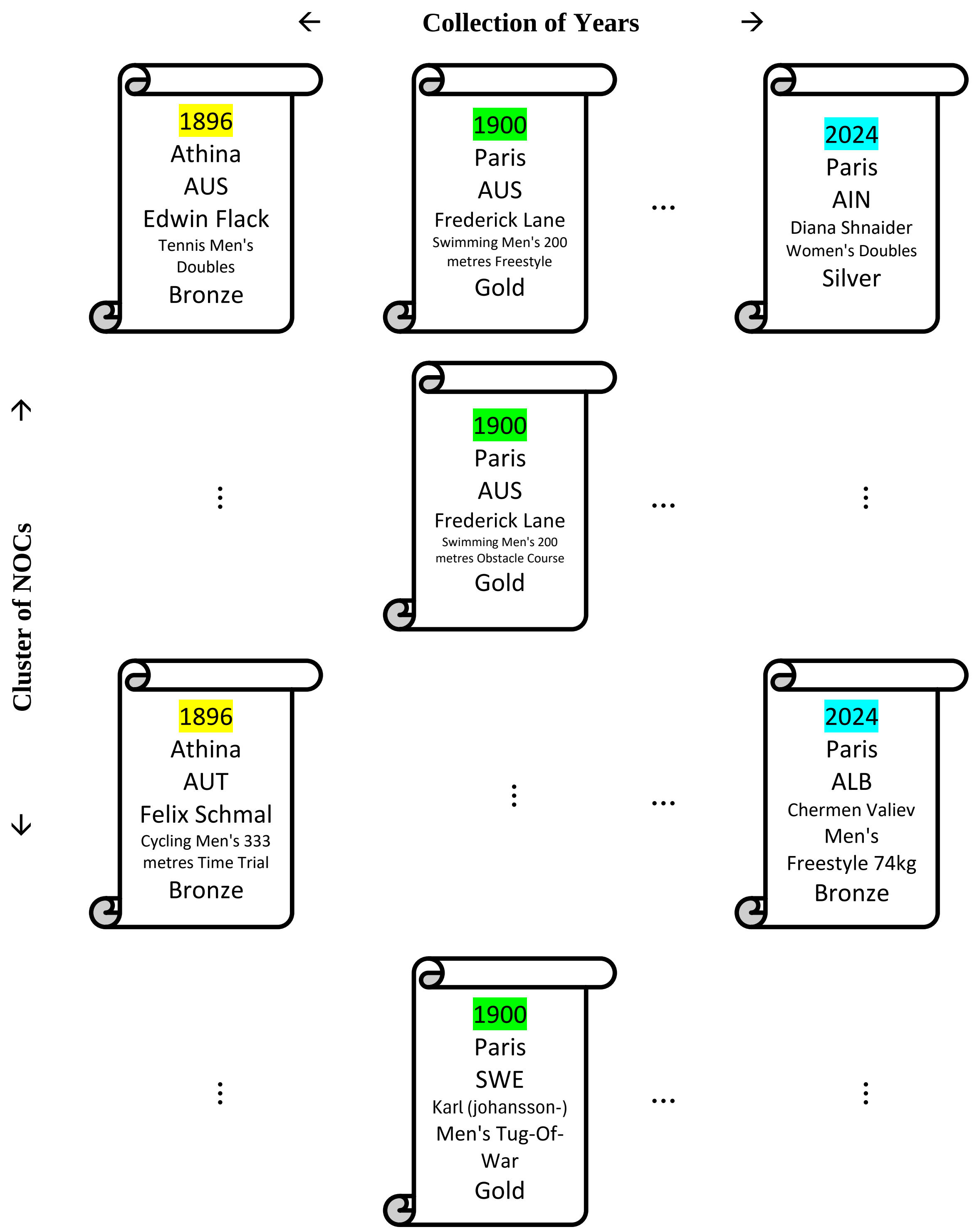
Countries, represented by a National Olympic Committee (NOC) are described by a three letter code — Australia is "AUS".

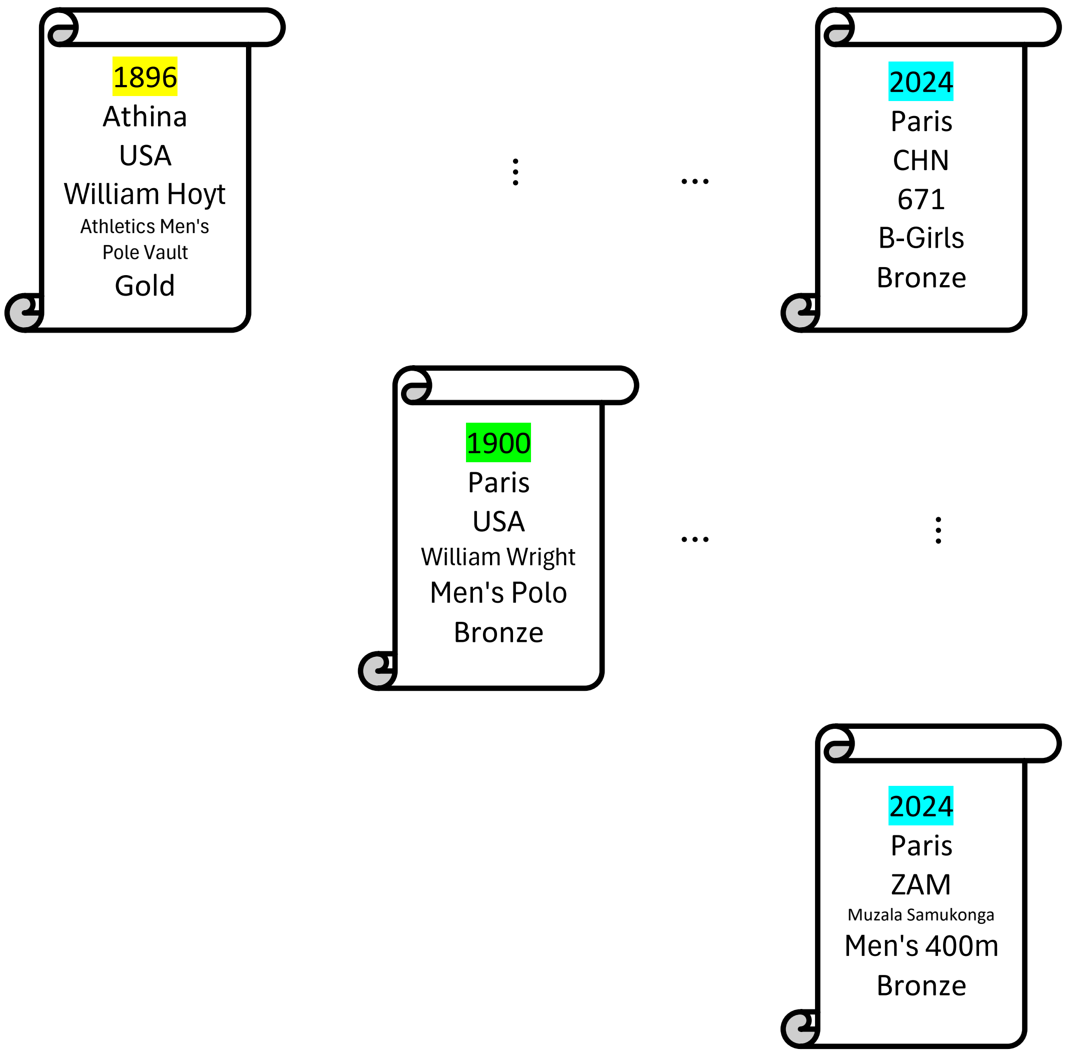
Each athlete is described by their name, NOC, sex, sport, event, and placing. Athletes should be stored in the collection if they received a Gold, Silver, or Bronze medal, and discarded otherwise.

A text file exists with the basic data of the athletes at all Olympic Games from 1896 to 2024. I have given you code to read this into your program. It is ordered alphabetically by athlete name, but you will need to re-structure the data as you read it in from the file:

- firstly, the collection must be grouped by year with years stored in increasing numerical order (i.e. earlier Games first, 2024 in Paris last); and then
- for each year, you should build the clusters of athletes belonging to that year in alphabetical order by NOC and, within that, retain the alphabetical ordering by athlete name.

For example:





You don't know which years are involved, how many NOCs there are, or how many athletes there are. All you know is what you've been told above and that athletes in a NOC are ordered as shown above and clustered within a year.

Task

Based only on the information above (and not what is in the text file or specified after this point in this document):

a Which kind of abstract data type (*binary tree, general tree, array, stack, priority queue, double-ended queue, set, list,* etc.) would you use to model the **cluster of athletes** within a year (i.e. within an Olympic Games)? In two–three sentences, justify your answer by indicating the functionality required.

b Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the **cluster of athletes** within a year (i.e. within an Olympic Games)? In two–three sentences, justify your answer.

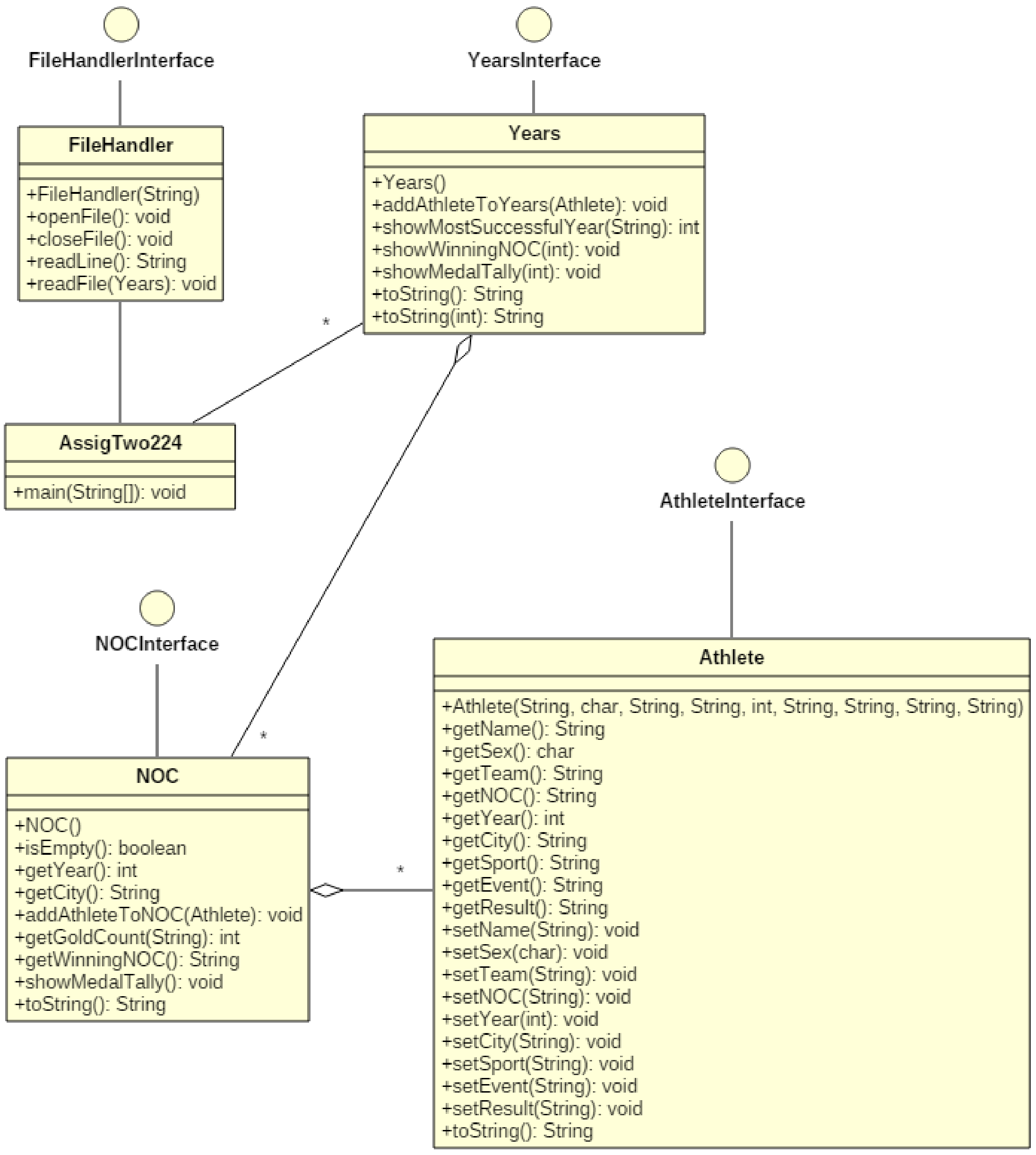
c Which kind of abstract data type (*binary tree, general tree, array, stack, priority queue, double-ended queue, set, list,* etc.) would you use to model the **collection of years**? In two–three sentences, justify your answer by indicating the functionality required.

d Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the **collection of years**? In two–three sentences, justify your answer.

e Implement your answers to (a) and (b) in `Noc.java` and your answers to (c) and (d) in `Years.java`. Read on for more detail of what is required.

Program specification

The software architecture is shown below.



There are nine program files:

- `AssigTwo224.java` — the harness class and holds the `main()` method. *This is complete.*
- `FileHandlerInterface.java` and `FileHandler.java` — contains methods for reading in the data from the text file. You may find it useful to alter the `LIMIT` final variable and to uncomment some `System.out.println()` statements during development. *These files are otherwise complete.*
- `AthleteInterface.java` and `Athlete.java` — specifies and implements the Athlete ADT, data structure, and methods. *These files are complete.*
- `NocInterface.java` and `Noc.java` — specifies and implements the ADT representing the cluster of athletes within a year (i.e. within an Olympic Games). *The interface is complete.*
- `YearsInterface.java` and `Years.java` — specifies and implements the ADT representing the collection of Years (i.e. all Summer Olympic Games). *The interface is complete.*

You must define the instance variables (i.e. the data structures) in `Noc.java` and `Years.java` according to your answers in (a) and (b), and in (c) and (d), respectively.

You must them complete the methods in `Noc.java` and `Years.java`. You are not permitted to change any code that you are given, but you may add methods beyond those present to assist you with your solution.

To implement the collection of `Years` as an array you would need to create instance data as follows:

```
    protected Noc []NOCs;
    protected int countNOC;
```

To implement the collection of `Years` as a linked list you would need to define the instance data to be:

```
    protected Node firstNOC;
```

Similarly, to implement the cluster of Athletes (the `NOC`s) as an array you would need to declare instance data as follows:

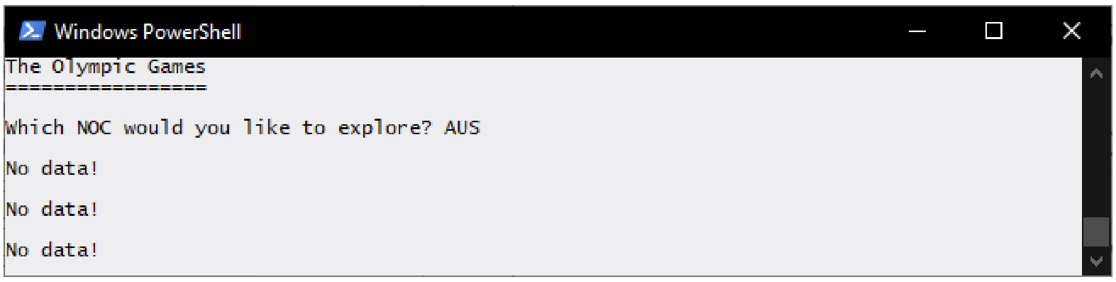
```
    protected Athlete []athletes;
    protected int countAthlete;
```

Or,to implement the cluster of Athletes (the `NOC`s) as a linked list you would declare the instance data as follows:

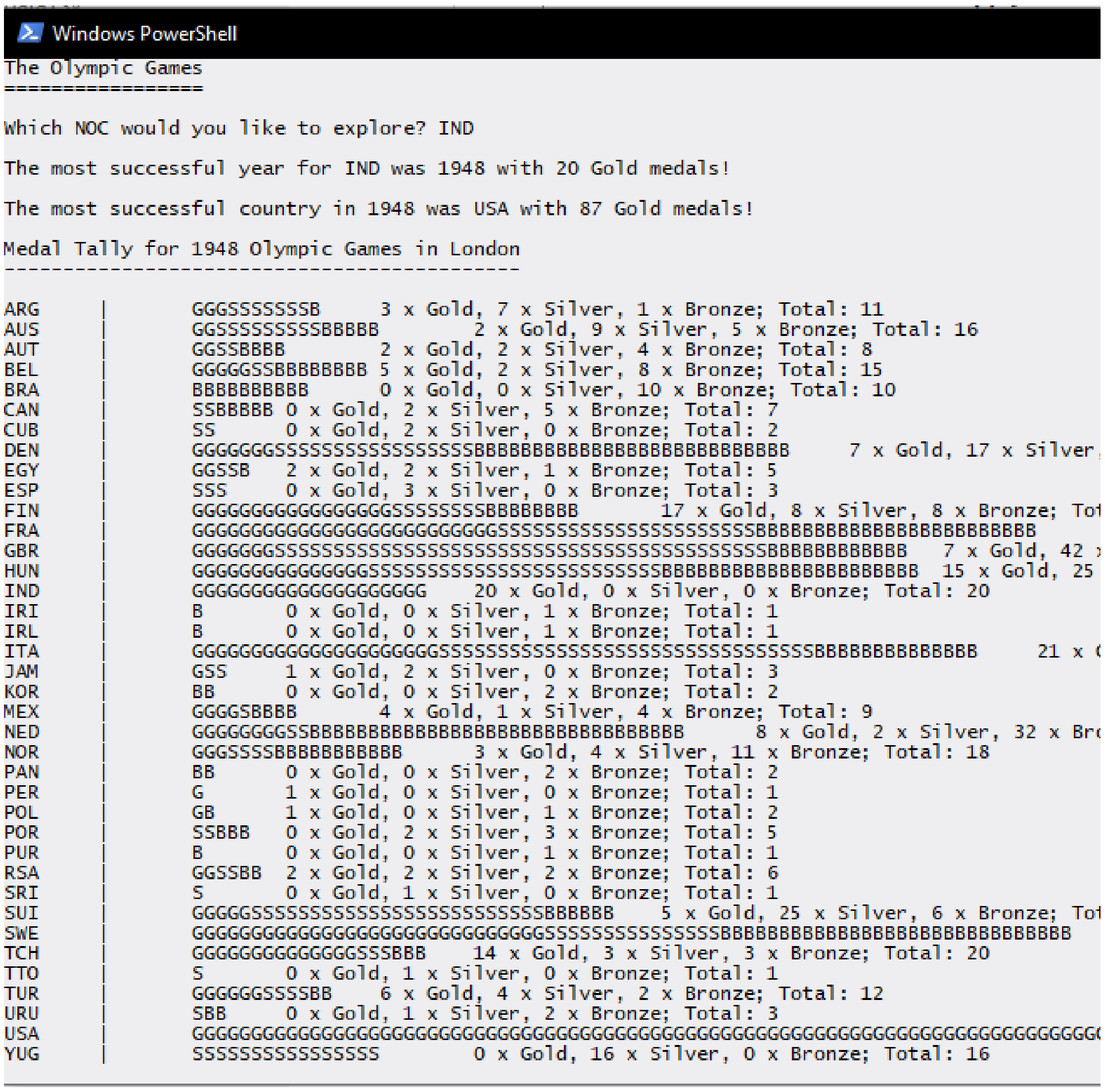
```
    protected Node firstAthlete;
```

The `NodeInterface.java` and `Node.java` files from lectures are also provided to you as complete files. Additionally, the datafile, `data.csv`, is also provided to you. You should not use its contents to assist you with answering (a)–(d), but may open the file in most text and program editors, including **VS Code**.

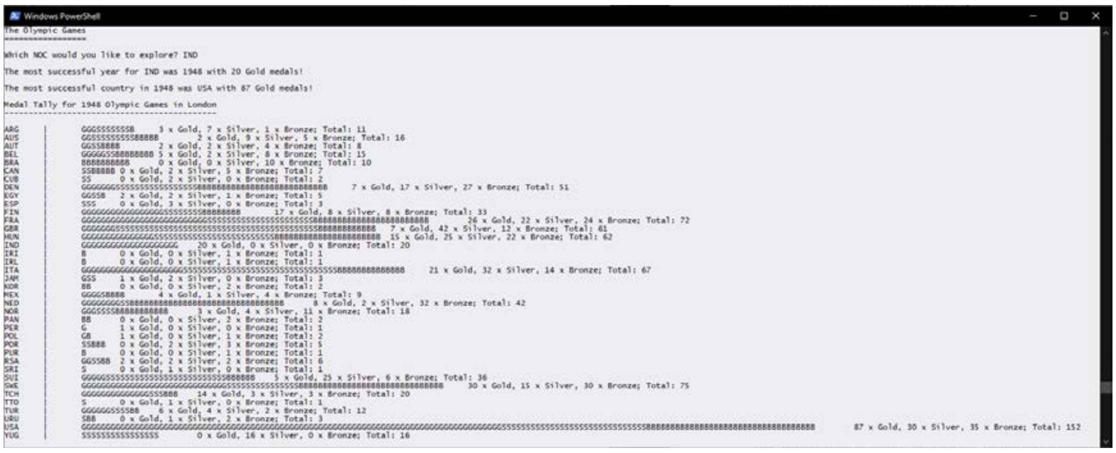
Once completed, your program should produce the following output *if there is no data* (with “AUS” entered by the user):


```
The Olympic Games
=================

Which NOC would you like to explore? AUS

No data!

No data!

No data!
```

If the full datafile is read in, and “IND” is chosen as the NOC, the following output should be produced (truncated to make it readable):


```
The Olympic Games
=================

Which NOC would you like to explore? IND

The most successful year for IND was 1948 with 20 Gold medals!

The most successful country in 1948 was USA with 87 Gold medals!

Medal Tally for 1948 Olympic Games in London
--------------------------------------------

ARG     |       GGGSSSSSSSB     3 x Gold, 7 x Silver, 1 x Bronze; Total: 11
AUS     |       GGSSSSSSSSSBBBBB        2 x Gold, 9 x Silver, 5 x Bronze; Total: 16
AUT     |       GGSSBBBB        2 x Gold, 2 x Silver, 4 x Bronze; Total: 8
BEL     |       GGGGGSSBBBBBBBB 5 x Gold, 2 x Silver, 8 x Bronze; Total: 15
BRA     |       BBBBBBBBBB      0 x Gold, 0 x Silver, 10 x Bronze; Total: 10
CAN     |       SSBBBBB 0 x Gold, 2 x Silver, 5 x Bronze; Total: 7
CUB     |       SS      0 x Gold, 2 x Silver, 0 x Bronze; Total: 2
DEN     |       GGGGGGGSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBBBBB     7 x Gold, 17 x Silver
EGY     |       GGSSB   2 x Gold, 2 x Silver, 1 x Bronze; Total: 5
ESP     |       SSS     0 x Gold, 3 x Silver, 0 x Bronze; Total: 3
FIN     |       GGGGGGGGGGGGGGGGGSSSSSSSSBBBBBBBB       17 x Gold, 8 x Silver, 8 x Bronze; Tot
FRA     |       GGGGGGGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBB
GBR     |       GGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBB   7 x Gold, 42 x
HUN     |       GGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBB  15 x Gold, 25
IND     |       GGGGGGGGGGGGGGGGGGGG    20 x Gold, 0 x Silver, 0 x Bronze; Total: 20
IRI     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
IRL     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
ITA     |       GGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBB     21 x G
JAM     |       GSS     1 x Gold, 2 x Silver, 0 x Bronze; Total: 3
KOR     |       BB      0 x Gold, 0 x Silver, 2 x Bronze; Total: 2
MEX     |       GGGGSBBBB       4 x Gold, 1 x Silver, 4 x Bronze; Total: 9
NED     |       GGGGGGGGSSBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB      8 x Gold, 2 x Silver, 32 x Bro
NOR     |       GGGSSSSBBBBBBBBBBB      3 x Gold, 4 x Silver, 11 x Bronze; Total: 18
PAN     |       BB      0 x Gold, 0 x Silver, 2 x Bronze; Total: 2
PER     |       G       1 x Gold, 0 x Silver, 0 x Bronze; Total: 1
POL     |       GB      1 x Gold, 0 x Silver, 1 x Bronze; Total: 2
POR     |       SSBBB   0 x Gold, 2 x Silver, 3 x Bronze; Total: 5
PUR     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
RSA     |       GGSSBB  2 x Gold, 2 x Silver, 2 x Bronze; Total: 6
SRI     |       S       0 x Gold, 1 x Silver, 0 x Bronze; Total: 1
SUI     |       GGGGGSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBB    5 x Gold, 25 x Silver, 6 x Bronze; Tot
SWE     |       GGGGGGGGGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
TCH     |       GGGGGGGGGGGGGGSSSBBB    14 x Gold, 3 x Silver, 3 x Bronze; Total: 20
TTO     |       S       0 x Gold, 1 x Silver, 0 x Bronze; Total: 1
TUR     |       GGGGGGSSSSBB    6 x Gold, 4 x Silver, 2 x Bronze; Total: 12
URU     |       SBB     0 x Gold, 1 x Silver, 2 x Bronze; Total: 3
USA     |       GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
YUG     |       SSSSSSSSSSSSSSSS        0 x Gold, 16 x Silver, 0 x Bronze; Total: 16
```

The full screen of output is:



```
Windows PowerShell
The Olympic Games
=================

Which NOC would you like to explore? IND

The most successful year for IND was 1948 with 20 Gold medals!

The most successful country in 1948 was USA with 87 Gold medals!

Medal Tally for 1948 Olympic Games in London
--------------------------------------------

ARG     |       GGGSSSSSSSB     3 x Gold, 7 x Silver, 1 x Bronze; Total: 11
AUS     |       GGSSSSSSSSSBBBBB        2 x Gold, 9 x Silver, 5 x Bronze; Total: 16
AUT     |       GGSSBBBB        2 x Gold, 2 x Silver, 4 x Bronze; Total: 8
BEL     |       GGGGGSSBBBBBBBB 5 x Gold, 2 x Silver, 8 x Bronze; Total: 15
BRA     |       BBBBBBBBBB      0 x Gold, 0 x Silver, 10 x Bronze; Total: 10
CAN     |       SSBBBBB 0 x Gold, 2 x Silver, 5 x Bronze; Total: 7
CUB     |       SS      0 x Gold, 2 x Silver, 0 x Bronze; Total: 2
DEN     |       GGGGGGGSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBBBBB     7 x Gold, 17 x Silver, 27 x Bronze; Total: 51
EGY     |       GGSSB   2 x Gold, 2 x Silver, 1 x Bronze; Total: 5
ESP     |       SSS     0 x Gold, 3 x Silver, 0 x Bronze; Total: 3
FIN     |       GGGGGGGGGGGGGGGGGSSSSSSSSBBBBBBBB       17 x Gold, 8 x Silver, 8 x Bronze; Total: 33
FRA     |       GGGGGGGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBB        26 x Gold, 22 x Silver, 24 x Bronze; Total: 72
GBR     |       GGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBB   7 x Gold, 42 x Silver, 12 x Bronze; Total: 61
HUN     |       GGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBB  15 x Gold, 25 x Silver, 22 x Bronze; Total: 62
IND     |       GGGGGGGGGGGGGGGGGGGG    20 x Gold, 0 x Silver, 0 x Bronze; Total: 20
IRI     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
IRL     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
ITA     |       GGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBB     21 x Gold, 32 x Silver, 14 x Bronze; Total: 67
JAM     |       GSS     1 x Gold, 2 x Silver, 0 x Bronze; Total: 3
KOR     |       BB      0 x Gold, 0 x Silver, 2 x Bronze; Total: 2
MEX     |       GGGGSBBBB       4 x Gold, 1 x Silver, 4 x Bronze; Total: 9
NED     |       GGGGGGGGSSBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB      8 x Gold, 2 x Silver, 32 x Bronze; Total: 42
NOR     |       GGGSSSSBBBBBBBBBBB      3 x Gold, 4 x Silver, 11 x Bronze; Total: 18
PAN     |       BB      0 x Gold, 0 x Silver, 2 x Bronze; Total: 2
PER     |       G       1 x Gold, 0 x Silver, 0 x Bronze; Total: 1
POL     |       GB      1 x Gold, 0 x Silver, 1 x Bronze; Total: 2
POR     |       SSBBB   0 x Gold, 2 x Silver, 3 x Bronze; Total: 5
PUR     |       B       0 x Gold, 0 x Silver, 1 x Bronze; Total: 1
RSA     |       GGSSBB  2 x Gold, 2 x Silver, 2 x Bronze; Total: 6
SRI     |       S       0 x Gold, 1 x Silver, 0 x Bronze; Total: 1
SUI     |       GGGGGSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBB    5 x Gold, 25 x Silver, 6 x Bronze; Total: 36
SWE     |       GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB     30 x Gold, 15 x Silver, 30 x Bronze; Total: 75
TCH     |       GGGGGGGGGGGGGGSSSBBB    14 x Gold, 3 x Silver, 3 x Bronze; Total: 20
TTO     |       S       0 x Gold, 1 x Silver, 0 x Bronze; Total: 1
TUR     |       GGGGGGSSSSBB    6 x Gold, 4 x Silver, 2 x Bronze; Total: 12
URU     |       SBB     0 x Gold, 1 x Silver, 2 x Bronze; Total: 3
USA     |       GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB      87 x Gold, 30 x Silver, 35 x Bronze; Total: 152
YUG     |       SSSSSSSSSSSSSSSS        0 x Gold, 16 x Silver, 0 x Bronze; Total: 16
```

The program should commence by opening the datafile and reading all the data into the program. The data must be stored in appropriate collections. At the top level there should be the Years and then for each of these there should be a collection of athletes (clustered into NOCs in alphabetical order of NOC name and then, by virtue of the order in the datafile, by athlete name).

As an athlete's data is read from the file, the details should be checked for placing. If no medal was received, the athlete's data should be discarded (i.e. not stored). If, however, the athlete is a medal winner, the data should be further examined to identify which year it is from (to determine which year it is stored in) and which NOC it is from (to determine where in the year the data belongs).

Once all the data have been read in and stored in the collections (or discarded), the user will be asked which NOC they are interested in. The data are then searched for the year (most recent if there are ties) that this NOC won the most gold medals. The year and total number of gold medals should be shown. From above (with "IND") as input:

```
The most successful year for IND was 1948 with 20 Gold medals!
```

Then, the program should investigate that year to determine which NOC received the most gold medals and display the NOC's name and the number of gold medals. From above (with "IND") as input:

```
The most successful country in 1948 was USA with 87 Gold medals!
```

Finally, the Medal Table should be built and then displayed. This can be done by traversing the NOC data. There is no need to store any results in an intermediate form other than the number of gold, silver, bronze, and total medals for the 'current' NOC. After producing a heading (which includes the year and host city), a line should be produced for each NOC in which a G is shown for every gold medal, an S for every silver medal, and a B for every bronze medal. Finally, the line should end with a summary. The information can be separated by tab (`'\t'`) characters. An example from 1948 is as follows:

```
AUS  |  GGSSSSSSSSSBBBBB   2 x Gold, 9 x Silver, 5 x Bronze; Total: 16
```

I suggest a staged approach:

1. cut back the collection to just 15–40 athletes (by changing `LIMIT` in `FileHandler.java`'s `readFile()` method)…
2. build the collection by year (with only one athlete for each year), printing it as you go using `Years`', `Noc`'s, and `Athlete`'s `toString()` methods. I.e. implement `Noc()`, `isEmpty()`, `getYear()`, `toString()`, and the basics of `addAthleteToNOC()` from the `Noc` class, and `Years()`, `isEmpty()`, `addAthleteToYears()`, and `toString()` from the `Years` class. When you are happy that this works — i.e. when you are seeing the years printed in increasing order and without duplicates…
3. build the cluster for each year adding all athletes for that year, printing it as you go. I.e. complete `addAthleteToNOC()` from the `Noc` class. When you are happy that this works — i.e. when you are seeing the athletes printed within the year in alphabetical order of NOC (and athlete name)…
4. write `getGoldCount()` from the `Noc` class and `showMostSuccessfulYear()` from the `Years` class…
5. write `getWinningNOC()` from the `Noc` class and `showWinningNOC()` from the `Years` class…
6. write `getCity()` and `showMedalTally()` from the `Noc` class and `showMedalTally()` from the `Years` class, and then…
7. Reinstate `LIMIT` to `Integer.MAX_VALUE` and you're done!

You may find the following Java methods from the `String` class useful: `compareTo()`, `equals()`, `equalsIgnoreCase()`, and `length()`.

For reference, my uncommented `Noc.java` is about 200 lines in length and my uncommented `Years.java` is about 170 lines.

Program Style

No method that you write should be longer than a screen-full of code. *You cannot change any distributed code.*

Your program should follow the following coding conventions:

- `final` variable identifiers should be used as much as possible, should be written all in upper case and should be declared before all other variables in a method;
- identifiers should be meaningful and variable and method names should be expressed in `camelCase`;
- instance variables should be used sparingly with parameter lists used to pass information in and out of methods;
- local variables should only be defined at the beginning of methods and their identifiers should start with a lower case letter;
- every `if` and `if-else` statement should have a block of code (i.e. collections of lines surrounded by `{` and `}`) for both the `if` part and the `else` part (if used);
- every `do`, `for`, and `while` loop should have a block of code (i.e. `{}`s);
- the keyword `continue` should not be used;
- the keyword `break` should only be used as part of a `switch` statement;
- opening and closing braces of a block should be vertically aligned;

- all code within a block should be aligned and indented 1 tab stop (or 4 spaces) from the braces marking this block;
- commenting:
    - There should be a block of header comment at the top of each file which includes at least
        - file name
        - student name
        - student identity number
        - a statement of the purpose of the program
        - date
    - Each variable declaration should be commented;
    - There should be a comment identifying groups of statements that do various parts of the task; and
    - Comments should describe the strategy (algorithm) of the code and should not simply translate the Java into English

Marking scheme

Task/Topic	**Maximum mark**
Design	
ADTs chosen wisely	3
Data structures correct and justified	3
Program operates as specified	
`Noc.java`	
Constructor (`Noc()`)	1
`isEmpty()`	½
`getYear()`	1
`getCity()`	1
`addAthleteToNOC()`	
• First athlete for the NOC	1
• In alphabetical order of NOC (and retaining alphabetical order of athlete)	2
`getGoldCount()`	
• Searches all athletes in the year	1
• Calculates correct result	1
`getWinningNoc()`	
• Searches all athletes in the year	1
• Calculates correct result	1
`showMedalTally()`	
• Calculates count correctly for each NOC	1 ½
• Displays histogram row correctly including ‘stars’ and summary	2
`toString()`	1
`Years.java`	
Constructor (`Years()`)	1
`isEmpty()`	½
`addAthleteToYears()`	
• First athlete for the collection	1
• First athlete for the year	1
• In increasing numerical order of years	2
• Discarding non-medallist athletes	1
`showMostSuccessfulYear()`	
• Behaves correctly for empty dataset	½

• Processes all years	1
• Calculates and displays correct result	1
`showWinningNOC()`	
• Behaves correctly for empty dataset	½
• Processes correct year, showing result	1
`showMedalTally()`	
• Behaves correctly for empty dataset	½
• Processes correct year	1
• Shows correct title including ‘balanced underlining’	1
`toString()`	1
`toString(int)`	1
Program Style	
Does not unnecessarily repeat tests or have other redundant/confusing code	4
Uses correctly the Java naming conventions	4
Alignment of code and use of white space makes code readable	4
Always uses blocks in branch and loop constructs	4
Meaningful identifiers	4
Variables defined at the start of methods only	4
Header comments for the program (author, date etc)	4
Each variable declaration is commented	4
Comments within the code indicate the purpose of sections of code (but DO NOT just duplicate what the code says)	4

The program will be marked out of 72.

What and how to submit

You should submit `Years.java` and `Noc.java`. *You should not submit any other files.*

You may provide your answers to tasks (a)–(d) as notes as part of your submission, or you may include your answers as comments in `Years.java` and/or `Noc.java`.

You should submit the files to the “Assignment 2” assignment drop-box on *KIT107*’s MyLO site.

Plagiarism and Cheating:

Practical assignments are used in the School of ICT for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is substantially your own.

Working with others

One effective way to grasp principles and concepts is to discuss the issues with your peers and/or friends. You are encouraged to do this. We also encourage you to discuss aspects of practical assignments with others. However, once you have clarified the principles, you must express them in writing or electronically entirely by yourself. In other words: you must develop the algorithm to solve the problem and write the program which implements this algorithm alone and with the help of no one else (other than staff). *You can discuss the problem with other students, but not how to solve it.*

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else (including online systems).
- Cheating is an offence under the Ordinance of Student Academic Integrity within the University. Furthermore, the ICT profession has ethical standards in which cheating has no place.
- Cheating can involve two or more parties.
    - If you allow written work, computer listings, or electronic version of your code to be borrowed or copied by another student you are an equal partner in the act of cheating.
    - You should be careful to ensure that your work is not left in a situation where it may be stolen by others.
- Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved. The case will be referred to the Head of the School of ICT for consideration of further action.

Generative AI

Under the University's Assessment and Results Procedure, all work that you submit as your own must be your own work. You can use generative artificial intelligence (AI) to learn, just like you would study with a classmate or ask a friend for advice — i.e. to learn concepts and grasp principles. You are *not permitted* to consult generative AI tools (such as ChatGPT) to learn how to complete assessment tasks, or to provide partial/full solutions to assessment tasks. In particular, you are not permitted to present the output of generative AI as your own work for your assignments or other assessment tasks. This constitutes an academic integrity breach.

Generative AI use
is not permitted.

Julian Dermoudy, September 7th 2024.