**Data Structures and Representation (ZEIT2103)**

**Assignment 3 – Social Network**

**100 Marks = 17%**

**Due Date: 11:55 PM, Friday 27 May 2022**

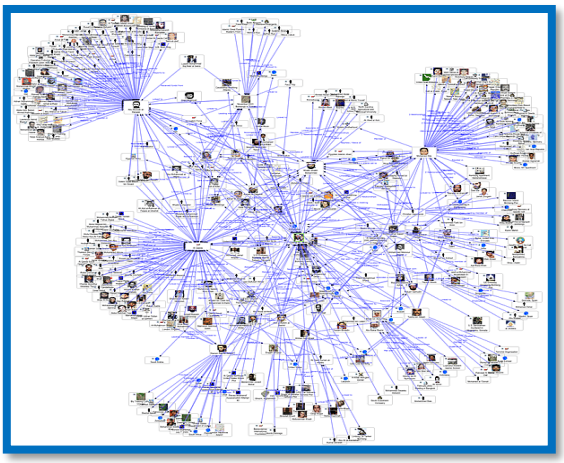
---

## Assignment Summary

In this assignment, you will investigate the use of graphs, hashing, reading files, priority queues and other data structures and deal with multiple interacting objects.

This is **a group assessment** ( I assume your team is ready now). Although a single copy of the code is needed for the submission, the amount of work each team member has done will contribute to their final mark. Also, each team member will get an individual mark for Task 6 (reflection).

## Problem Description



A social network can be represented as an undirected graph with each person represented as a node and each connection between two nodes an edge.

In this assignment, you will be given a randomly generated sample of data consisting of more than 1,000 accounts (people) (`data.txt`), with information of each, i.e., id, name, date of birth (DoB), suburb and a set of numbers representing any connection between the current person and the other people with the id provided, provided in the supplied file. In the figure below, as an example, Gillian Garnett, whose DoB is 27 November 1994 and the suburb in which he lives Majura, is connected to both Minna Whittaker and Stephen Ernest whose IDs are 1 and 3, respectively.

```
1       Minna Whittaker 1980-06-15, Majura, 2
2       Gillian Garnett 1994-11-27, Majura, 1, 3
3       Stephen Ernest  2006-11-01, Canberra Central, 2
```

You will be asked to read this data and represent it as a graph. Bearing in mind the amount of data, as hashing will be used to accelerate the insertion and search processes, you will be required to investigate the efficiency of different hashing functions. Then, you will implement your *mini social network application*, which suggests friends to a given account, finds mutual friends of two friends and presents reminders of upcoming birthdays.

**Note**: please read task 6 to get an overall idea about the expectations in terms of teamwork, for example:

- effective composition and distribution of tasks (you can use MS Teams[1], Trello or other tools),
- clear milestones are identified,
- all team members are contributing,
- regular meetings (at least once a week) - think of having an agenda for each meeting;
- managing the source code: you may use Git[2345] or any other similar tool; not mandatory, though;

# Getting Started – Resources

---

[1] https://support.microsoft.com/en-us/office/create-a-plan-with-planner-in-teams-fa65ee5c-3c9b-42da-97b3-2fcd1a1c626d

[2] https://www.youtube.com/watch?v=RzYJvSnzlMk

[3] An example of using Git in NetBeans https://netbeans.apache.org/kb/docs/ide/git.html.

[4] https://eclipsesource.com/blogs/tutorials/egit-tutorial/

[5] https://www.jetbrains.com/help/idea/using-git-integration.html

**Data Structures and Representation (ZEIT2103)**

**Assignment 3 – Social Network**

**100 Marks = 17%** **Due Date: 11:55 PM Friday 27 May 2022**

---

Some resources have been provided for this assignment in the assignment area of the course Moodle website. They consist of the following class definitions:

- `NodeInterface.java` –is a java interface that holds several methods you need to implement in the `Node.java` class.
- `Node.java` – represents a node (person) in the graph with its adjacency list of edges (friends).
- `GraphInterface.java` –is a java interface that holds the basic methods needed to implement the `Graph.java` class
- `Graph.java` that constructs an undirected graph with some basic operations
- `SocialNetworkInterface.java` contains some basic methods needed to complete the `SocialNetwork.java` class
- `HarnessClass.java` class, which you will update by implementing your test cases in it.
- `data.txt` – contains randomly generated accounts (people), with information of each, i.e., id, name, date of birth (DoB), suburb and a set of numbers representing any connection between the current person and the other people with the id provided;

# Assignment Tasks

In general, this assignment involves several kinds of tasks: implementing some classes, documenting design decisions and reflecting on your programming processes. The latter two should be included in the pdf file `ass3.pdf`, which should be structured with a heading for each task and sub-headings for the different activities. Do not forget to mention the group members in the report.

## Task 1. Implementing Node and Edge Classes (10 Marks)

- As previously mentioned, each person in a graph is represented by a node. Therefore, the `Node class` (which implements `NodeInteface`) has four main attributes: `id, name; dateOB` (of type `LocalDate`); `suburb;` and `adj,` which represents connections to friends and is structured as a `HashMap`, whereby `HashMap < Integer, Edge> adj`.

  You need to

  - define the constructor (`public Node(Integer id, String name, LocalDate dob, String suburb))` that defines the initial values of personal information using the supplied id, name, DoB and suburb. Also, remember to complete the declaration of the `adj` variable. **[1 mark]**
  - write all the getter methods required **[2 marks]**; and
  - implement the `toString()` method which returns a string representation of all data about a Node object; the string format may look like: `Node {id = 1, name = Minna Whittaker, DOB = 1980 -06-15, suburb = Majura}` **[2 marks]**

  **Expectations for documentation [2 marks]:** (1) documentation of your test cases as instructed in lectures; (2) a UML diagram;

- As the `Edge` class has one attribute `friend`, that is, (`protected Node friend;`), you should
  - implement the constructor (**`public Edge(Node friend)`**) that creates a new edge with the supplied parameter (`friend`); **[1 mark]**; and
  - write a `toString()` which returns a string representation of all data about an Edge object; the string format may look like this: `friend= {1, Minna Whittaker, 1980 -06-15, Majura}` **[1 mark]**;

**Expectations for documentation :** (1) documentation of your test cases; and (2) a UML diagram; **[1 mark]**

Note: although no exceptions are expected to be thrown in this task, you can do so if you see needed.

## Task 2. Hashing: overriding hashCode() and equals() methods in Node Class (10 Marks)

This task requires you to write a hash function designed specifically for `Node` objects, overriding the `hashCode()` method **[3 marks]** in the `Node` class. When implementing it, you can choose which attributes to use, how to normalise and weight them, etc. You may try some of the hash functions presented in lectures or conduct some research when designing yours.

**Warning 1: you should NOT use the built-in hashCode() method for any part of the calculation of your hash function. If you do, you will automatically receive a ZERO mark for this task.**

Then, to conform with the attributes you used in `hashCode()`, you need to override the `equals(Object obj)` method **[2 marks]** as discussed in the lecture.

**Expectations for documentation:** (1) at least one paragraph (and no more than half a page) describing your approach while rationalising/explaining any special weightings adopted or attributes omitted **[4 marks]**. You should reference any literature you use in `ass3.pdf`; (2) test the hashCode() method you implemented **[1 mark]** (i.e., check how many duplicates keys are generated (you can use MS excel to see if there are duplicates; or insert all values in a set and check its size).

Note: although no exceptions are expected to be thrown in this task, you can do so if you see needed.

## Task 3. Implementing Graph Class (25 Marks)

To begin forming an undirected graph representing a social network, a `Graph.java` class is provided to you, which has an empty constructor, and an attribute called `nodeList` will hold all nodes (people information) in the graph.

Then, implement the following:

- `public Node addNode(Integer id, String name, LocalDate dob, String suburb)`, which creates a new Node based on the data provided, then adds it to a graph if it does not exist, and returns the Node created; otherwise, throw an exception; **[3 marks]**
- `public void addEdge(Node from, Node to)`, which creates and adds an edge between both nodes if it does not exist; otherwise, throw an exception; remember, this is an undirected graph, so you need to add the Edge to both nodes. **[3 marks]**
- `public void removeEdge(Node from, Node to)`, which removes an edge between two nodes if it exists. No action is needed if the edge does not exist. **[3 marks]**
- `public removeNode(Node node)` appropriately removes a node from `nodeList` and any edge between this node and other nodes; **Hint**: to achieve the second part of this subtask, you may need to use `Iterator`; if a node does not exist, throw an exception; **[3 marks]**
- `public Set< Edge> getNeighbors(Node node)`, which returns a set of friends of a given node; otherwise, throw an exception if the node does not exist; **Hint:** returning the set of friends can be done with only one line of code; and **[3 marks]**

- `public String toString()`, which returns a string representation of each person's name and friends. A possible format of this string may be **[3 marks]**
  - person 1: --> friend1 friend2
  - person 2: --> friend1 friend2

```
Minna Whittaker:-->Gillian Garnett
Gillian Garnett:-->Genevieve Swinton          Stephen Ernest          Minna Whittaker
Stephen Ernest:-->Genevieve Swinton           Gillian Garnett
Bevan Kenley:-->Olive Rexley                  Genevieve Swinton
```

**Expectations for documentation :** (1) documentation of test cases for each method; (2) justification of any helper method(s) you might implement; (3) UML diagram; **[7 marks]**

Note: although no exceptions are expected to be thrown in this task, you can do so if you see needed.

## Task 4. Implementing SocialNetwork Class (20 Marks)

This task aims to read the data files provided to you, construct a graph, and conduct some types of social media operations. This class has an attribute `sn` of type `Graph` instantiated in the constructor**.** Then, implement:

- `public void processFile()` that reads the data file and builds the graph accordingly while remembering to call this method in the constructor; **[5 mark]**
- `public List<Node> suggestFriends(Node currentPerson)` which returns a list of suggested friends. This will return a `List<Node>` such that each Node in the list is a friend of a friend that lives in the same suburb as `currentPerson`; **[5 mark]** and
- `public List<String> getMutualFriends(Node x, Node y)` which returns a list of all names of mutual friends of two friends (`x` and `y`). **[4 mark]**

**Expectations for documentation :** (1) a short paragraph justifying the selection of the class you used to read `data.txt` **[1 mark];** (2) UML diagram **[1 mark];** (3) justification of your approach to addressing this task **[1 mark]**; and (4) big O discussion for the `suggestFriends` **and** `getMutualFriends` methods**[3 mark];**.

## Task 5. Implementing Birthday Reminder (no assistance given) (20 Marks)

Extend the `SocialNetwork` class by writing a `public String remindBDEvents(Node currentPerson)` method that takes a `Node` object as an input and returns a `String` that represents all the friends of the input node sorted based on the periods until their DoB; for instance, the figure below, which was generated using a different database, shows all the friends of Gillian Garnett sorted based on their next birthdays.

```
Hey Gillian Garnett:->
Genevieve Swinton has his birthday after 0 Year, 1 Months, 10 days
Minna Whittaker has his birthday after 0 Year, 4 Months, 28 days
Stephen Ernest has his birthday after 0 Year, 9 Months, 14 days
```

**[15 marks]**

**Hint: (1) use PriorityQueues; and (2) you will need to override the compareTo method in Node class; this is to help with sorting based on the period remaining until their birthday**

**Expectations for documentation :** (1) justification of your design; (2) documentation of test cases for each method; [**5 marks**]

## Task 6. Reflection on Teamwork (15 Marks)

You should reflect on

- how tasks were distributed (who did what); did the team meet the project schedule? The tool(s) you used to manage the project, how did you maintain team communication (i.e. how often did you meet and how meetings were organised), how you managed the source code (i.e., did the team use any version control tools, like GitHub or similar tools, not mandatory, though), the challenges the team faced and how you dealt with them.
- Did you encounter any technical or personal challenges resulting from teamwork? How did you deal with them? What could you do to prevent these challenges in your future teamwork projects?
- Have your java skills increased? What lessons did you learn?
- Any other points you want to highlight.

There is no exact expectation about the length of this task, but you need to answer all points.

## Marking and Submission

You are required to refer to and follow the 'Requirements and Expectations for Programming Assignments' document. If you do not, you are likely to be seriously disappointed by your final mark!

If you are aiming for a basic pass/credit result, you need to show your mastery of the basics by completing the following tasks:

- provide and discuss the required designs and specifications with appropriate justifications and references, and have them noted by your lab tutor;
- ensure that your Java code is well designed, with design decisions documented and the requested UML diagrams provided;
- correctly and efficiently use your Java code to implement the classes and methods specified in the relevant tasks;
- add suitable **Javadoc** comments to your Java code for each class, attribute and method, and include inline comments on methods where necessary to clarify the intent of the code; ***marks will be deducted if you fail to do so;***
- design your Java code in a modular way using a sensible set of methods;
- ensure that the layout and indentation of your Java code are consistent and readable; and
- discuss the design and validation of a test strategy for the code you developed.

- correctly used proper java name conventions

To aim for a higher mark, you need to complete all the specified tasks to the basic standard (above) and also discuss the extensions you tackled, including the research, design, testing involved, and the big O calculations of the approaches you used.

When you have completed your work, you need to submit the files you have written, that is, **`Edge.java`, `Node.java`, `Graph.java`, `SocialNetwork.java`, `HarnessClass.java`** and **`ass3.pdf`** to **Moodle**.

**Only** one member of each group is required to submit the source code, but **everyone** has to upload the **reflection** part (task 6). In the report, you must mention the members of the group.

Remember: the amount of work each team member has done will contribute to their final mark. Therefore, be clear about the tasks each member has completed.