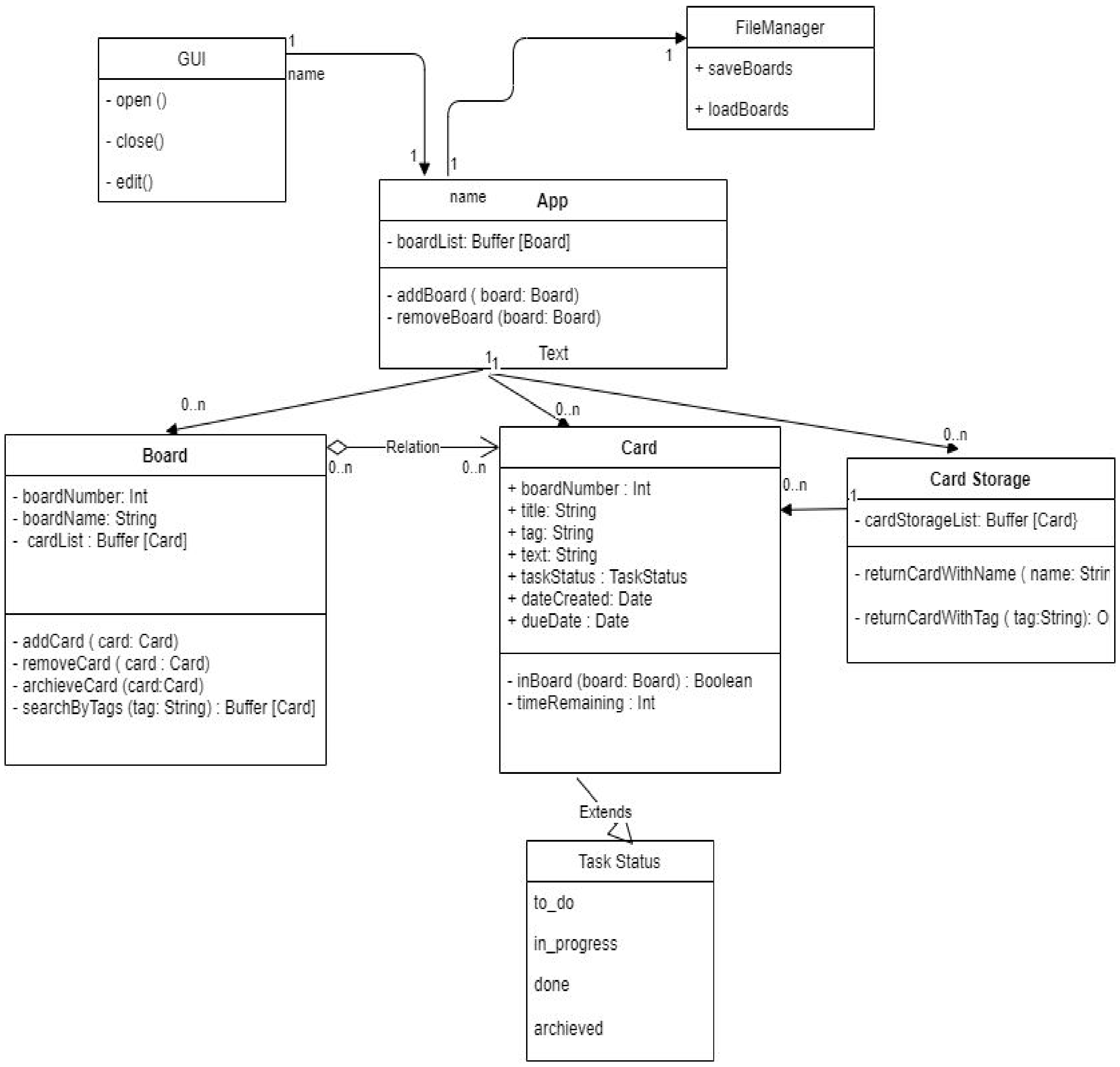
TECHNICAL PLAN

Personal Information

- Title : Kanban App
- Name: Linh Ngo
- Student Number: 906502
- Degree Program: Bachelor's Programme in Technology and Science- Data Science
- Year of studies: Year 1
- Date: 17/02/2021

1. **Class structure**

- Class FileManager allows the program to read from the text file the saved data and also writes to a file.
- Class Card store some variables that are related to object Card. Trait TaskStatus consists of 4 case objects that define the status of the card.
- Class Board stores some variables that are related to the object board and some functions related to class Card such as add, remove, update Card.
- Class App consists of boards and some functions for Board.



GUI
- open ()
- close()
- edit()
1
name
FileManager
+ saveBoards
+ loadBoards
1
1
1
name
App
- boardList: Buffer [Board]
- addBoard (board: Board)
- removeBoard (board: Board)
Text
1
1
0..n
0..n
0..n
Board
- boardNumber: Int
- boardName: String
- cardList : Buffer [Card]
- addCard (card: Card)
- removeCard (card : Card)
- archieveCard (card:Card)
- searchByTags (tag: String) : Buffer [Card]
Relation
0..n
0..n
Card
+ boardNumber : Int
+ title: String
+ tag: String
+ text: String
+ taskStatus : TaskStatus
+ dateCreated: Date
+ dueDate : Date
- inBoard (board: Board) : Boolean
- timeRemaining : Int
0..n
1
Card Storage
- cardStorageList: Buffer [Card}
- returnCardWithName (name: Strin
- returnCardWithTag (tag:String): O
Extends
Task Status
to_do
in_progress
done
archieved

2. **Use case description**

- THe users open the program. If they have already used and saved the Kanban Board before, the board will show their boards.
- The user can also add or remove the cards/boards by clicking into the icon button. Some methods in the class Board and Table will be used to make some changes to the program
- The user also can also filter by tags. There will be a search box and users can search for cards they want to find.

3. **Algorithm**

- A Map structure will be used for some filtering methods (for example: filter based on tags) as it's easier to filter based on key-value pairs. (key: Card, value: tag:String type).
- To parse the stored file, I will use some useful methods such as trim, for loop, match case. Throwing exceptions is also implemented when loading and writing from/to file to avoid some error cases.

4. **Data Structure**

- The program contains a lot of Board and Card instances, so I needed to have collections to store them. I decide to choose mutable Buffer as these collections may be modified by adding or removing board or card later on.
- The file will be stored in JSON or human readable format, but I haven't decided yet.

5. **Schedule**

Demo	24/02 - 26/02
Algorithm phase 1	26/02-10/03
Debugging	10/03 - 14/03
Algorithm phase 2	14/03 - 20/03
Debugging	21/03 - 24/03
UI & IO	24/03 - 10/04
Improvement/ Testing	11/04 - 24/04

6. **Testing plan**

- Unit testing will be tested mainly for some core methods that cause some changes in the board(add, remove, update, filter method) to check the validity of the algorithm. I will use test classes to check what the method should return and compare it with a result of a function. (assert)
- The GUI will be tested manually after creating each component. Different use cases such as dragging and dropping between columns, clicking the button, adding/removing cards to/from board, and handling errors will also be tested

in the user interface to test the functionality of some core methods and the effect of them on program objects again.

-

7. Reference

- Drag and Drop in JavaFx
- Source: https://docs.oracle.com/javafx/2/drag_drop/jfxpub-drag_drop.htm
- Drag and Drop in JavaSwing
https://zetcode.com/javaswing/draganddrop/
- Scala API
https://www.scala-lang.org/api/current/index.html
- Scala Numeric Date Formatting

Source:https://alvinalexander.com/scala/scala-number-nnumeric-date-formatting-casting-examples/

8. Appendices