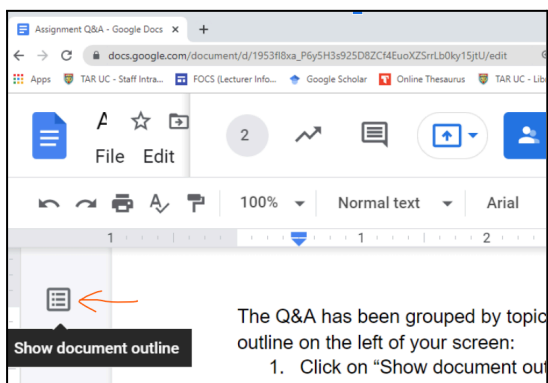


Assignment Q & A

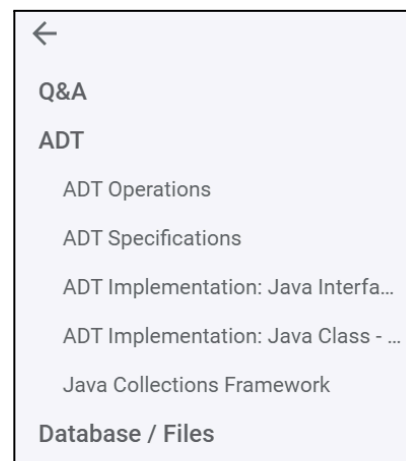
This document contains questions and answers regarding the assignment, and **will be updated with new entries as and when they arise**. Note, however, that questions for which the answers may be found in the Assignment Specification, Assignment Report Template and/or Assignment Rubrics will not need to be addressed here. In addition, to avoid redundant work, we will not be responding to questions asked or discussed in class, emails/Classroom if the information may be obtained from the Assignment Specification, Assignment Report Template and/or Assignment Rubrics.

This Q&A has been grouped by topic which you can conveniently access via the document outline on the left of your screen:

First, click on “Show document outline”:



Next, click on the link for the relevant topic:



Source code organized in correct packages: ADT, Boundary,Control, Entity, DAO, etc.

<https://docs.google.com/document/d/1JY-Jvzd37I5CpRzC2wfbNQjZhgQniCDI/edit>

Discuss entity classes involved in modules

1. Collection ADTs

1.1. General Questions

Collection ADTs are able to store objects of different types. Collection ADTs should be defined using a **generic type<T>** to represent the type of objects that will be stored in the collection.

1.1.1. Purpose

What is the point of implementing collection ADTs by ourselves when the code is simply given in the sample code: what is preventing us from just copying whatever java interface or class we need for our ADT? Is that considered plagiarism?

ANSWER:

- The point is to learn about collection ADTs (what are the different collection ADTs, what are the different ways of implementing collection ADTs, etc.).
- If you use the collection ADT implementation from the sample code or from other sources, do acknowledge that at the top of your Java interfaces and/or classes.
- The code given in the sample code is just one example of many ways to implement some of the basic collection ADTs. There are many alternative ways of implementing such collection ADTs. In addition, there are many other collection ADTs - refer to the following non-exhaustive list for some examples:

List	Stack	Queue	Set	Binary Tree	Dictionary	Etc
<ul style="list-style-type: none"> • List • Sorted List • etc. 	<ul style="list-style-type: none"> • Stack • etc. 	<ul style="list-style-type: none"> • Queue • Priority Queue • Deque • etc 	<ul style="list-style-type: none"> • Set • HashSet • Etc. 	<ul style="list-style-type: none"> • Binary Search Trees • AVL Trees • etc. 	<ul style="list-style-type: none"> • Dictionary • HashMap 	

- Create your **own collection ADTs** with the required characteristics and operations.
- If students choose to directly use the collection ADT implementation given in the sample code, their marks for assessment criteria “Collection ADT implementation - originality” would be considered to be in the “Developing” category. Refer to the Assignment Rubrics for details.

1.1.2. What does *implementation approach* mean?

“Implementation approaches” refers to the different ways of implementing the collection ADTs. For example, here is a non-exhaustive list of different implementation approaches for the List ADT:

- Linear array implementation (chapter 4,7)
- Circular array implementation (chapter 4,7)
- Linear linked implementation (Practical 5)
- Doubly linked implementation (not cover)
- Circular linked implementation (practical 5)
- Circular doubly linked implementation (not cover)
- Linked implementation with dummy

1.1.3. Can we import / use

`java.util.List, ArrayList, Arrays, Stack, Queue, Deque, priority queue, vector, LinkedList, Linkedstack, Hashmap, functions, functional, HashSet, predicate, others collection interfaces.` in our assignment?

ANSWER: **No, cannot.** because the assignment requires you to create and implement user-defined collection ADTs. Therefore, you are **not allowed** to use any predefined collection interfaces and classes from the Java Collections Framework.

1.1.4. Can import use `java.util.Comparator` or `Comparable`, **exception handling event** for sorting algorithm

ANSWER: Yes for `Comparator` / **Comparable interface (Practical 7)**

1.1.5. Can import `Collection.sort()` to do sorting?

ANSWER: **NO, `Collection.sort()` - Cannot used.**

you should implement your own sorting code like **chapter 8 Sorting**:
bubble sort, selection sort, insertion sort

1.1.6. Can we use Java's predefined classes and interfaces such as `String`, `Double`, `Character`, `timer`, `Scanner` and `GregorianCalendar`, `java.io`, `nanotime`?

ANSWER: Yes, you may use any Java interfaces and classes that are not collections.

1.1.7. Can we use `java.util.Iterator`?

ANSWER: Yes, you may use any Java interfaces and classes that are not collections.

~ This section ends here. The next section starts on a new page. ~

2. ADT Specification (Format)

2.1. Good and Bad Practices

Recall: Creating & Using an ADT

Step 1 Write the ADT specification

- Write an ADT specification which describes the characteristics of that data type and the set of operations for manipulating the data. **Should not include any implementation or usage details.**

Good and Bad Practices	
Note: Bad practices are highlighted in yellow. Do not mention Array, Linked	
✓	✗
<u>ADT List</u> A list is a linear collection of entries of a type T which allows duplicate elements. An entry may be added at a specified position or at the end of the list. The list position starts from 1.	<u>ADT ArrayList</u> An array list is a linear collection of entries of a type T which allows duplicate elements. An entry may be added at a specified position or at the end of the list. The list position starts from 1.
add(T newEntry) Description : Adds newEntry to the end of the list. Postcondition : newEntry has been added to the end of the list.	add(Customer newEntry) Description : Adds a new customer to the end of the list. Postcondition : The new customer has been added to the end of the list.
add(T newEntry) Description : Adds newEntry to the end of the list. Postcondition : newEntry has been added to the end of the list.	addCustomer(int id, String name, String mobile) Description: Add a new customer to the list. ...
<u>ADT List</u> A list is a linear collection of entries of a type T which allows duplicate elements. An entry may be added at a specified position or at the end of the list. The list position starts from 1.	ADT CustomerList This list is to manage a list of customers for an online store. ...

2.2. Operations to include

2.2.1. Need to include basic operations declared as public ?

QUESTION: If there are certain basic operations of the collection ADT that I don't use in my control or entity classes, do I need to include them in the collection ADT's specification, Java interface and Java class?

ANSWER:

- YES. Your ADT should contain all the basic operations for that ADT for completeness. Remember the principle of reusability ... your ADT may be used in future projects for different

applications, different entity objects, etc.

- However, although it is not compulsory to use all the basic collection ADT operations in your control and/or entity classes, it would be good if you can use your creativity to demonstrate meaningful use of the ADT operations where possible in your control and/or entity classes.

2.2.2. Need to include the iterator operations in the ADT specification?

QUESTION: Do we need to include an iterator into the ADT Specification if we implemented the iterator for our collection ADT?

ANSWER: Do **not include any private operations** that are not accessible by the control or entity classes. Only the operations that are available for the **control and/or entity classes** to invoke should be included in the ADT specification.

2.2.3

~ This section ends here. The next section starts on a new page. ~



3. ADT Implementation

3.1. Java Interface<T>

<u>Good and Bad Practices</u> <i>Note: Bad practices are highlighted in yellow.</i>	
	
<pre> public interface ListInterface<T> { public boolean add(T newEntry); public boolean add(int newPosition, T newEntry); public T remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, T newEntry); public T getEntry(int givenPosition); public boolean contains(T anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>	<pre> interface CustomerListInterface<T> { public boolean add(Customer newEntry); public boolean add(int newPosition, Customer newEntry); public Customer remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, Customer newEntry); public Customer getEntry(int givenPosition); public boolean contains(Customer anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>
<pre> public interface ListInterface<T> { public boolean add(T newEntry); public boolean add(int newPosition, T newEntry); public T remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, T newEntry); public T getEntry(int givenPosition); public boolean contains(T anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>	<pre> interface LinkedListInterface<T> { public boolean add(T newEntry); public boolean add(int newPosition, T newEntry); public T remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, T newEntry); public T getEntry(int givenPosition); public boolean contains(T anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>
<pre> public interface ListInterface<T> { public boolean add(T newEntry); public boolean add(int newPosition, T newEntry); public T remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, T newEntry); public T getEntry(int givenPosition); public boolean contains(T anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>	<pre> interface LinkedListInterface<T> { public boolean add(T newEntry); public boolean add(int newPosition, T newEntry); public T remove(int givenPosition); public void clear(); public boolean replace(int givenPosition, T newEntry); public T getEntry(int customerID); public T getEntry(String name); public boolean contains(T anEntry); public int getNumberOfEntries(); public boolean isEmpty(); public boolean isFull(); } </pre>

3.2. Java Class (Implementation)

3.2.1. Good and Bad Practices

<u>Good and Bad Practices</u> <i>Note: Bad practices are highlighted in yellow.</i>	
	
<pre>public class ArrayList<T> implements ListInterface<T></pre>	<pre>class CustomerArrayList<T> implements ListInterface<T></pre>

3.2.2. Sorting algorithms

QUESTION: Will I get additional marks if I use a sorting, merge algorithms ?

ANSWER: Depends on the appropriateness of implementation and use of ADT (correctness). **Marks will be awarded according to the assessment criteria and corresponding descriptors specified in the Assignment Rubrics.**

~ This section ends here. The next section starts on a new page. ~

4. Subsystems (Entity classes, Control classes)

4.1. Use of multiple collection ADTs

1. QUESTION: Will I get more marks if I use more than one collection ADT in my subsystem?

ANSWER: Use 1 ADT only. You should use a **team's collection ADT** with add on methods. Marks will be awarded according to the assessment criteria and corresponding descriptors specified in the Assignment Rubrics. If your use of multiple collection ADTs fulfills the 'Ideal' descriptors of any assessment criteria, then you will be awarded marks in the 'Ideal' range for the relevant assessment criteria. A word of caution: please read and understand the descriptors in each assessment criterion specified in the Assignment Rubrics and avoid having a one-dimensional assumption that "If I do ..., I will get higher marks in the assignment". This is as we've observed students who focus only on one aspect of the assignment and ignore other parts, resulting in the students receiving lower overall marks. Therefore, we would like to reiterate and stress that **marks will be awarded according to the assessment criteria and corresponding descriptors specified in the Assignment Rubrics.**

2. QUESTION: Create samples of hardcoded entity objects when running your module during demo.

ANSWER: use constructors to invoke ADT method to return a collection of hard-coded entity values.

Refer to ECB sample

Method to return a collection of with hard-coded entity values

```
public ListInterface<Product> initializeProducts() { .. }
```

ListInterface<Product> prodList = new ArrayList<>();

declare collection ADT object in control class , initializerData class

```
pList.add(new Product("A1001", "Book", 10));
pList.add(new Product("A1002", "Pen", 50));
pList.add(new Product("A1003", "Pencil", 30));
pList.add(new Product("A1004", "Notepad", 40));
pList.add(new Product("A1005", "Ruler", 15));
return pList; }
```

3. How many collections of ADT I use in my subsystem?

Use **ONE Collection of ADT** that is selected by Team.

Each member should contribute to the implementation of ADT methods (modified, add new) in the Implementation class and invoke the methods from the collection ADT to complete your subsystem.

4.2. Entity Classes

1. QUESTION: Should constructors, setters, getters, `toString`, and `equals` methods be included in each entity class?

ANSWER: Yes, it is a good OO programming practice to include constructors, setters, and getters in each entity class. In addition, you should also override the `toString`, and `equals` methods in each entity class. Some methods will not work correctly if you did not override the `equals` and `compareTo` methods.

2. QUESTION: Can entity classes include input statements (to obtain input from the user) and/or output statements (to display output to the user)?

ANSWER: NO. This is bad design. Entity classes should NOT include any input statements to obtain input from the user and should NOT include any output statements to display output to the user. The best practice is to make each entity class a POJO (plain old Java object) so that they can be independent of the entity classes and control classes that use them.

An entity class represents the template for objects that you will store to persistent storage (e.g. binary files, database table, etc.). Examples of entity classes are `Runner`, `Customer`, etc. Entity classes **SHOULD NOT include I/O statements** to obtain input from the user (e.g. using `Scanner`) or displaying output to the user (e.g. using `System.out.println`).

3. Question: how to use a **collection ADT in an entity class for showing 1 to Many relationships**.

ANSWER: 1 tutor → many tutorialGroup relationship: In **Tutor** class:

```
ListInterface <TutorialGroup> tutGroupList = new ArrayList<>();
```

4.3. Database / Files (DAO)

1. QUESTION: For the database, do we need to use a built-in Apache database or do we need to specially download the database software to connect to databases such as SQL Server?

ANSWER: **Not required** to use a database / file (text, dat) for this assignment. If you do, you may use any open-source database.

4.4. Miscellaneous questions

1. QUESTION: Do we need to do type validation for the user input?

ANSWER: Yes, include only validations which involve the **invocation of a method** of the **collection object**.

2. QUESTION: For the **Generate Report** functionality, do I have to write my own codes?

ANSWER: Yes, you cannot use other 3rd party software. You are required to write code in Java to generate the reports.

3. QUESTION: For “Generate relevant reports”, can I just list all the entries in the collection object?

ANSWER: No, because listing all entries in the collection object is already specified as a separate use case under the subsystem. If you want to get Ideal or Approaching marks for the criteria “*Use of ADTs - creativity and complexity*”, all the implementations of your use cases should be such that it involves creativity and complexity that demonstrates a high level of understanding and competency in using collection ADTs.

Use collections of ADT for generate Reports in correct format :

Title, printed date, customize data in meaningful, field names, number of records displayed, etc.

4. QUESTION: Why is it better to use the Java interface instead of the implementation class when declaring collection variables or parameters (see below for example)?

```
ListInterface<Employee> employeeList;
```

ANSWER: It's best practice to use the Java interface (e.g. **ListInterface**) as the type for the variable because

- a. this declaration restricts the allowed methods on the variable to only the methods declared in **ListInterface** and
- b. enforces the requirement for the implementation classes (e.g., **ArrayList**, **LinkedList**, etc) to provide the implementations of all the abstract methods in the interface.

Moreover, if we used the interface instead of the class as the type for method parameters, e.g.

```
void displayList(ListInterface<Employee> employeeList)
```

we could replace the class (e.g., **ArrayList**, **LinkedList**, etc) with another class that also implements **ListInterface** without modifying the subsequent statements in the program.

~ This section ends here. The next section starts on a new page. ~

5. Assignment Report

TEAM REPORT
1. Abstract Data Type (ADT)
1.1 ADT Specification
1.2 ADT Implementation
INDIVIDUAL REPORTS
2. Use of ADTs
Student1 Name
Student2 Name
Student3 Name
Student4 Name

1. QUESTION: For the complete source code (for collection ADT implementation, control classes, entity classes), can I embed screenshots of the source code in my report?

ANSWER: copy & paste source code listing from netbean project / use print function in netbean IDE.

Please refer to your tutor's requirements.

General answer:

Yes, as long as they can be easily and clearly read. Do ensure that:

- Use light mode instead of dark mode in your IDE.
- Ensure that there is proper indentation for your class. To automatically format, either
 - Right-click on your mouse and then select **Format**, or
 - Press <Alt><Shift><F>
- The font size is big enough (or you can zoom-in by pressing the Alt key and then using the scroll wheel on your mouse) before you capture the screenshot using either the *Snipping Tool* or *Snip & Sketch* tool that is automatically provided by Windows.

2. QUESTION: For the documentation, do we need to write the overview of the chosen application or the specific subsystem that we worked on?

ANSWER: NO.

3. QUESTION: Is there any guideline for the documentation format like: Times New Roman, 12 point font size and so on?

ANSWER: Use the font type and size in the Assignment Report Template.