# Criterion C: Development (outline format)

## Technique: Abstract Data Structure (Linked List)

Class: HomeworkStudentLinkedList.java

### Success criteria:

* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The system should be able to generate the effort grade for a student based on test scores, homework completion rate and the client’s perceived effort grade, with a 50%, 25% and 25% weighting to each respective factor

Justification: The use of linked lists is appropriate as it allows for naturally sequentially assigned homework to be stored in the order in which they were assigned.

Code sample:

Text

Description automatically generated

## Technique: Exception handling

Class: Person.java

### Success criteria:

* The system should not allow for incorrect data types, incorrect data format or duplicate data to be entered, i.e. there should be error management in case invalid input is entered

Justification: The use of try catch is to check whether the input passed by the user is valid and is utilised to handle their behaviour provided they are invalid.

Code sample:

Text

Description automatically generated

Text

Description automatically generated

## Technique: Encapsulation

Class: Person.java

### Success criteria:

* The client should be able to enter the classes they teach and assign students to each class, with the data for each student being separate from other students, as no student can be in 2 classes for the same teacher

Justification: The use of encapsulation is to protect the base class attributes from external modification.

Code sample:

Text

Description automatically generated

## Technique: Serialisation (and deserialisation, WIP)

Class: HomeworkStudentLinkedList.java

### Success criteria:

* The system should be able to save entered data to secondary storage whenever the application is closed and should load said data when it is opened, provided the directory of stored data is not changed

Justification: The use of serialisation allowed for homework details to be stored to disk when the application was closed, the computer was shut down, etc.

Code samples:



Text

Description automatically generated

## Technique: Abstract Data Structure Traversal

Class: HomeworkStudentLinkedList.java

### Success criteria:

* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The client should be able to enter the classes they teach and assign students to each class, with the data for each student being separate from other students, as no student can be in 2 classes for the same teacher
* The client should be able to search for a student and view all data corresponding to the student upon the input of a query for first name, last name or student ID
* The client should be able to view all students in a class

Justification: Using a temp node allows for the traversal of the linked list for data to be stored and or removed, achieved with the utilisation of a while loop.

Code sample:

Text

Description automatically generated

## Technique: Sorting (Quicksort)

Class: Person.java

### Success criteria:

* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The client should be able to view all students in a class

Justification: The utilisation of sorting is to allow for the information being presented to the user to be more comprehensible. This is to also enable a method, changeID, to function.

Code sample:

Text

Description automatically generated

## Technique: Abstraction

Class: Person.java, Teacher.java and Student.java

### Success criteria:

* The client must be able to enter, edit and store test scores for each student
* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The client should be able to enter the classes they teach and assign students to each class, with the data for each student being separate from other students, as no student can be in 2 classes for the same teacher

Justification: The use of abstraction allows for necessary attributes to be stored for the person, teacher and student class.

Code sample:

Text

Description automatically generated



Word count – 982 words=

## Technique: Inheritance

Class: Teacher.java, Student.java and Person.java

### Success criteria:

* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The client should be able to enter the classes they teach and assign students to each class, with the data for each student being separate from other students, as no student can be in 2 classes for the same teacher

Justification: The student and teacher classes extend the person class such that code duplication is minimised

Code sample:

Text

Description automatically generated



## Technique: Exponential Moving Average

Class: To be generated

### Success Criteria:

1. The system should be able to generate the effort grade for a student based on test scores, homework completion rate and the client’s perceived effort grade, with a 50%, 25% and 25% weighting to each respective factor
2. The system should weight semester exam scores as 70% and average quarter test scores as 30% of the 50% weighting given to the final effort grade

Justification: An exponential moving average allows for the predicted test scores and by extension, effort grades to be generated using previous student test scores.

Code sample: The code for this section is still under a work in progress

## Technique: Additional Data Structures (ArrayLists)

Classes: Teacher.java, Person.java

### Success criteria:

* The client must be able to enter the number of total homework given to students and number of homework assignments completed by each student
* The client should be able to enter the classes they teach and assign students to each class, with the data for each student being separate from other students, as no student can be in 2 classes for the same teacher
* The client should be able to search for a student and view all data corresponding to the student upon the input of a query for first name, last name or student ID

Justification: ArrayLists were utilised to store input class objects, who in turn hold subject objects. They were also used to hold IDs of students.

Code sample:

The following method involves manipulation of arraylists

Text

Description automatically generated