BACS2063 Data Structures and Algorithms

ASSIGNMENT 202305

Programme Management

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University of Management and Technology's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Student Name	Student ID	Prog / Tut.Grp	Signature
BOON YONG YEOW	22PMR09689	RDSY2S1G1	[illegible]
LIM MENG FU	22PMR05891	RDS2S1G1	[illegible]

Table of Contents

A. TEAM REPORT **4**

1. Abstract Data Type (ADT) **4**

1.1 ADT Specification 4

1.2 ADT Implementation 6

1.2.1 List Interface: 6

1.2.2 DoublyLinkedList Implementation 7

B. INDIVIDUAL REPORTS **15**

2. Use of ADTs 15

2.1 Boon Yong Yeow 15

2.1.1 Entity: Tutor 15

2.1.2 Control 18

2.1.2.1 TutorManager 18

2.1.2.2 FilterCriteriaInterface 21

2.1.2.2.1 FacultyFilterCriteria 21

2.1.2.2.2 RatingFilterCriteria 22

2.1.2.3 ReportCriteriaInterface 22

2.1.2.3.1 FacultyReportCriteria 23

2.1.2.3.2 RatingReportCriteria 24

2.1.3 Boundary: MainApp 26

2.1.4 Screenshots (Boon Yong Yeow) 39

2.2 Lim Meng Fu 43

2.2.1 Entity 44

2.2.1.1 Course 44

2.2.1.2 Programme 46

2.2.2 Control 48

2.2.2.1 CourseManager 48

2.2.3 Boundary 53

2.2.3.1 Main 53

2.2.4 Screenshots 64

A. TEAM REPORT

1. Abstract Data Type (ADT)

1.1 ADT Specification

ADT List
List data structure. It allows adding, removing, and retrieving elements, among other operations.

boolean add(T newEntry)	
Description	Adds a new entry to the list
Precondition	-
PostCondition	The new entry is added at the end of the list
Returns	true if added, false otherwise

boolean remove(T anEntry)	
Description	Removes the specified entry from the list
Precondition	-
PostCondition	The first occurrence of the specified entry is removed from the list
Returns	true if removed, false otherwise

void clear()	
Description	Clears the list
Precondition	-
PostCondition	The list is empty
Returns	-

T getEntry(int pos)	
Description	Retrieves the entry at the specified position
Precondition	pos is valid number

PostCondition	The list remains unchanged
Returns	The entry at position pos

T getLast()	
Description	Retrieves the last entry in the list
Precondition	-
PostCondition	The list remains unchanged
Returns	The last entry in the list

boolean contains(T anEntry)	
Description	Checks if the list contains the specified entry
Precondition	-
PostCondition	The list remains unchanged
Returns	true if contains, false otherwise

boolean isEmpty()	
Description	Checks if the list is empty
Precondition	-
PostCondition	The list remains unchanged
Returns	true if empty, false otherwise

int getNoOfElement()	
Description	Gets the number of elements in the list
Precondition	-
PostCondition	The list remains unchanged
Returns	The number of elements in the list

Iterator<T> getIterator()	
Description	Gets an iterator for the list

Precondition	-
PostCondition	The list remains unchanged
Returns	An iterator for the list

void amend(T oldItem, T newItem)	
Description	Replaces an old item with a new item
Precondition	oldItem exists
PostCondition	The first occurrence of oldItem is replaced by newItem
Returns	-

T find(T item)	
Description	Finds an item in the list
Precondition	-
PostCondition	The list remains unchanged
Returns	The first occurrence of the item, null if not found

1.2 ADT Implementation

1.2.1 List Interface:

```java
package ADT;

import java.util.Iterator;

public interface ListInterface<T> {
    public boolean add(T newEntry);

    public boolean remove(T anEntry);

    public void clear();

    public T getEntry(int pos);

    public T getLast();
    public boolean contains(T anEntry);

    public boolean isEmpty();

    public int getNoOfElement();

    public Iterator<T> getIterator();

    //YY
    public void amend(T oldItem, T newItem);
    public T find(T item);
}
```

1.2.2 DoublyLinkedList Implementation

```
package ADT;

import Control.FilterCriteriaInterface;
import java.util.Iterator;

public class DoublyLinkedList<T> implements
ListInterface<T>,Iterable<T> {

    // Internal Node class (Entity)
    private class Node {
        T data;
        Node next;
        Node prev;

        Node(T data) {
            this.data = data;
            this.next = null;
            this.prev = null;
        }
    }

    private Node head;
    private Node tail;
    private int size;
    // Constructor (Control)
    public DoublyLinkedList() {
        this.head = null;
        this.tail = null;
        this.size = 0;
    }

    @Override
    // Add method (Control)
    public boolean add(T item) {
        Node newNode = new Node(item);
        if (head == null) {
            head = newNode;
            tail = newNode;
            size++;
            return true;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
            size++;
            return true;
        }
    }

    @Override
    // Remove method (Control)
    public boolean remove(T item) {
        Node forward = head;
        Node backward = tail;

        while (forward != null && backward != null) {

            // Check from the start
            if (forward.data.equals(item)) {
                if (forward.prev != null) {
                    forward.prev.next = forward.next;
                } else {
                    head = forward.next;
                }
                if (forward.next != null) {
                    forward.next.prev = forward.prev;
                } else {
                    tail = forward.prev;
                }
                size--;
                return true;
            }

            // Check from the end
            if (backward.data.equals(item)) {
                if (backward.prev != null) {
                    backward.prev.next = backward.next;
                } else {
                    head = backward.next;
                }
                if (backward.next != null) {
                    backward.next.prev = backward.prev;
                } else {
                    tail = backward.prev;
                }
                size--;
                return true;
            }

            // Move the pointers
            forward = forward.next;
            backward = backward.prev;
        }
        return false;
    }

    @Override
    // Find method (Control)
    public T find(T item) {
        Node forward = head;
        Node backward = tail;

        while (forward != null && backward != null) {

            // Check the node from the start of the list
            if (forward.data.equals(item)) {
                return forward.data;
            }
            // Check the node from the end of the list
            if (backward.data.equals(item)) {
                return backward.data;
            }

            forward = forward.next;
            backward = backward.prev;
        }

        return null;
    }

    @Override
    // Amend method (Control) with double-ended search
    public void amend(T oldItem, T newItem) {
        Node forwardTemp = head;
        Node backwardTemp = tail;
        while (forwardTemp != null && backwardTemp != null) {
            // Check from the start (head)
            if (forwardTemp.data.equals(oldItem)) {
                forwardTemp.data = newItem;
                return;
            }
            // Check from the end (tail)
            if (backwardTemp.data.equals(oldItem)) {
                backwardTemp.data = newItem;
                return;
            }
            forwardTemp = forwardTemp.next;
            backwardTemp = backwardTemp.prev;
        }
}

    @Override
    public boolean isEmpty() {
        return head == null;
    }

    @Override
    public Iterator<T> iterator() {
        return new Iterator<T>() {
            private Node current = head;

            @Override
            public boolean hasNext() {
                return current != null;
            }

            @Override
            public T next() {
                if (hasNext()) {
                    T data = current.data;
                    current = current.next;
                    return data;
                }
                return null;
            }
        };
    }

    // Filter method (Control)
    public void filter(FilterCriteriaInterface<T> criteria) {
        Node temp = head;
        while (temp != null) {
            if (!criteria.filter(temp.data)) {
                remove(temp.data);
            }
            temp = temp.next;
        }
    }


    //Methods for sorting
    public int getSize() {
        return size;
    }

    public T get(int index) {
        if (index < 0 || index >= size) {
            throw new IndexOutOfBoundsException("Index out of
bounds");
        }
        Node current = head;
        int count = 0;

        while (current != null) {
            if (count == index) {
                return current.data;
            }
            count++;
            current = current.next;
        }

        return null;
    }

    public void swap(int index1, int index2) {
        if (index1 == index2) return;

        Node node1 = getNodeAt(index1);
        Node node2 = getNodeAt(index2);

        if (node1 == null || node2 == null) return;

        T temp = node1.data;
        node1.data = node2.data;
        node2.data = temp;
    }

    private Node getNodeAt(int index) {
        if (index < 0 || index >= size) return null;

        Node current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }

        return current;
    }

    public void set(int index, T element) {
        if (index < 0 || index >= size) {
            throw new IndexOutOfBoundsException("Index out of
range");
```

```
        }
        Node current = head;
        int count = 0;
        while (current != null) {
            if (count == index) {
                current.data = element;
                return;
            }
            count++;
            current = current.next;
        }
    }

    public void addLast(T element) {
        Node newNode = new Node(element);
        if (tail == null) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
    }

    public DoublyLinkedList<T> deepCopy() {
        DoublyLinkedList<T> newList = new DoublyLinkedList<>();
        Node current = head;
        while (current != null) {
            newList.addLast(current.data);
            current = current.next;
        }
        return newList;
    }

    public String toString(){
        String str = "";
        Node currentNode = head;

        while(currentNode != null){
            str +=  "\n" +currentNode.data.toString();
            currentNode = currentNode.next;
        }

        return str;
    }

    @Override
    public void clear() {
    }

    @Override
    public T getEntry(int pos) {
        return null;
    }

    @Override
    public T getLast() {
        return null;
    }

    @Override
    public boolean contains(T anEntry) {
        return false;
    }

    @Override
    public int getNoOfElement() {
        return 0 ;
    }

    @Override
    public Iterator<T> getIterator() {
        return null;
    }

}
```

B. INDIVIDUAL REPORTS

2. Use of ADTs

2.1 Boon Yong Yeow

Student Name	Student ID	Prog / Tut.Grp	Signature
BOON YONG YEOW	22PMR09689	RDSG1	[illegible]

Subsystem Chosen : **Tutor Management**

2.1.1 Entity: Tutor

```
package Entity;

public class Tutor {
    private static int idCounter = 1000;
    public int id;
    private String name;
    private int experience; // years of experience
    private String qualifications; // academic degrees or
certifications
    private String specializations; // specific areas within the
subject
    private double rating;  // feedback and testimonials
    private String faculty; // faculty the tutor teaches in
    private double salary;
    private int cancellationRate;

    public Tutor() {
        this.id = ++idCounter;
        this.name = "John Doe";
        this.experience = 0;
        this.qualifications = "[null]";
        this.specializations = "[null]";
        this.rating = 0.0;
        this.faculty = "";
        this.salary = 3000;
        this.cancellationRate = 0;
    }
    public Tutor(String name, int experience, String
qualifications, String specializations, double rating, String
faculty,double salary, int cancellationRate) {
        this.id = ++idCounter;
        this.name = name;
        this.experience = experience;
        this.qualifications = qualifications;
        this.specializations = specializations;
        this.rating = rating;
        this.faculty = faculty;
        this.salary = salary;
        this.cancellationRate = cancellationRate;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getExperience() {
        return experience;
    }

    public void setExperience(int experience) {
        this.experience = experience;
    }

    public String getQualifications() {
        return qualifications;
    }

    public void setQualifications(String qualifications) {
        this.qualifications = qualifications;
    }
```

```
public String getSpecializations() {
    return specializations;
}

public void setSpecializations(String specializations) {
    this.specializations = specializations;
}

public double getRating() {
    return rating;
}

public void setRating(double rating) {
    this.rating = rating;
}

public String getFaculty() {
    return faculty;
}

public void setFaculty(String faculty) {
    this.faculty = faculty;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public int getCancellationRate() {
    return cancellationRate;
}

public void setCancellationRate(int cancellationRate) {
    this.cancellationRate = cancellationRate;
}

@Override
```

```
    public String toString() {
        return String.format("Tutor{id=%d, name='%s', 
experience=%d, qualifications='%s', specializations='%s', 
rating=%.2f, faculty='%s', salary=%.2f, cancellation=%d}",
                                 id, name, experience, qualifications, 
specializations, rating, faculty,salary, cancellationRate);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Tutor otherTutor = (Tutor) obj;
        return this.id == otherTutor.id;
    }

    public void setId(int id) {
        this.id = id;
    }

}
```

2.1.2 Control

2.1.2.1 TutorManager

This class serves as the interaction between Tutor Entity and Doubly Linked list

```
package Control;

import Entity.Tutor;
// import adt.ListInterface;
import ADT.DoublyLinkedList;
import java.util.Iterator;

public class TutorManager {
    public  DoublyLinkedList<Tutor> tutors;
    public int cancer;

    public TutorManager() {
        this.tutors = new DoublyLinkedList<>();
    }

    public void addTutor(Tutor tutor) {
        tutors.add(tutor);
    }

    public void removeTutorById(int id) {
        if (tutors.isEmpty()) {
            System.out.println("No tutors available to remove.");
            return;
        }
        Tutor toRemove = null;
        Iterator<Tutor> iterator = tutors.iterator();
        while (iterator.hasNext()) {
            Tutor tutor = iterator.next();
            if (tutor.getId() == id) {
                toRemove = tutor;
                break;
            }
        }
        if (toRemove != null) {
            tutors.remove(toRemove);
            System.out.println("Successfully removed tutor with ID
" + id);
        } else {
            System.out.println("Tutor with ID " + id + " not
found.");
        }
    }


    public Tutor findTutorById(int id) {
        if (tutors.isEmpty()) {
            System.out.println("No tutors available to find.");
            return null;
        }
        // Create a 'prototype' Tutor object with the ID you're
searching for
        Tutor prototypeTutor = new Tutor();
        prototypeTutor.setId(id);
        // Use the find method from the DoublyLinkedList class
        return tutors.find(prototypeTutor);
    }

    public void amendTutor(int id, String newName, int
newExperience, String newQualifications, String
newSpecializations, double newRating, String newFaculty, double
newSalary, int newCancellationRate) {
        Tutor oldTutor = findTutorById(id);
        if (oldTutor != null) {
            Tutor newTutor = new Tutor(newName, newExperience,
newQualifications, newSpecializations, newRating, newFaculty,
newSalary, newCancellationRate);
            tutors.amend(oldTutor, newTutor);
        }
    }


    public Iterable<Tutor> getAllTutors() {
        return tutors;  // Assuming 'tutors' is now an Iterable
    }

    public DoublyLinkedList<Tutor>
filterTutors(FilterCriteriaInterface<Tutor> criteria) {
        DoublyLinkedList<Tutor> filteredTutors = new
DoublyLinkedList<>();
        if (tutors.isEmpty()) {
            return filteredTutors;  // return empty list
        }

        // Use for-each loop to iterate over all tutors
        for (Tutor tutor : tutors) {  // Assuming 'tutors' is now
Iterable
            if (criteria.filter(tutor)) {
                filteredTutors.add(tutor);
            }
        }

        return filteredTutors;
    }

    public String generateReport(ReportCriteriaInterface<Tutor>
criteria) {
        // Create a deep copy of the original list
        DoublyLinkedList<Tutor> copiedList = tutors.deepCopy();

        // Generate the report based on the copied list
        String report = criteria.generateReport(copiedList);

        return report;
    }

    @Override
    public String toString() {
       return tutors.toString();
    }
}
```

2.1.2.2 FilterCriteriaInterface

A filter by criteria interface was created with the purpose of easier maintainability and expandability with multiple filtering criterias.

```
package Control;

public interface FilterCriteriaInterface<T> {
    boolean filter(T item);
}
```

2.1.2.2.1 FacultyFilterCriteria

Use-Case : This filter is to filter tutors by faculty.

```
package Control;
import Entity.Tutor;

public class FacultyFilterCriteria implements
FilterCriteriaInterface<Tutor> {
    private String faculty;

    public FacultyFilterCriteria(String faculty) {
        this.faculty = faculty;
    }
    @Override
    public boolean filter(Tutor tutor) {
        return tutor.getFaculty().equalsIgnoreCase(faculty);
    }
}
```

2.1.2.2.2 RatingFilterCriteria

Use-Case: Filter Tutors that achieve a minimum threshold entered by users.

```
package Control;
import Entity.Tutor;

public class RatingFilterCriteria implements
FilterCriteriaInterface<Tutor> {
    private double minRating;

    public RatingFilterCriteria(double minRating) {
        this.minRating = minRating;
    }

    @Override
    public boolean filter(Tutor tutor) {
        return tutor.getRating() >= minRating;
    }
}
```

2.1.2.3 ReportCriteriaInterface

Purpose: Easier expandability and maintainability in case of different reports are needed.

```
package Control;
import ADT.DoublyLinkedList;

public interface ReportCriteriaInterface<T> {
    public String generateReport(DoublyLinkedList<T> list);
}
```

2.1.2.3.1 FacultyReportCriteria

Use-Case: Summarizes faculty details like Total number of tutors, average rating and average experience.

```
package Control;
import ADT.DoublyLinkedList;
import Entity.Tutor;

public class FacultyReportCriteria implements
ReportCriteriaInterface<Tutor> {

    @Override
    public String generateReport(DoublyLinkedList<Tutor> list) {
        StringBuilder report = new StringBuilder();
        DoublyLinkedList<FacultyData> facultyDataList = new
DoublyLinkedList<>();

        int n = list.getSize();
        for (int i = 0; i < n; i++) {
            Tutor tutor = list.get(i);
            String faculty = tutor.getFaculty();

            FacultyData facultyData =
findOrCreateFacultyData(facultyDataList, faculty);

            facultyData.totalTutors++;
            facultyData.totalExperience += tutor.getExperience();
            facultyData.totalRating += tutor.getRating();
            facultyData.totalSalary += tutor.getSalary();
        }

        // Generate the report data
        int facultyCount = facultyDataList.getSize();
        for (int i = 0; i < facultyCount; i++) {
            FacultyData facultyData = facultyDataList.get(i);

            double avgExperience = facultyData.totalExperience /
(double) facultyData.totalTutors;
            double avgRating = facultyData.totalRating / (double)
facultyData.totalTutors;
            double avgSalary = facultyData.totalSalary / (double)
facultyData.totalTutors;
```

```
            report.append(String.format("| %-10s | %-13d | %-18.2f
| %-14.2f | %-13.2f |\n",
                                        facultyData.faculty,
facultyData.totalTutors, avgExperience, avgRating, avgSalary));
        }
        return report.toString();
    }

    private FacultyData
findOrCreateFacultyData(DoublyLinkedList<FacultyData> list, String
faculty) {
        int size = list.getSize();
        for (int i = 0; i < size; i++) {
            FacultyData existingData = list.get(i);
            if (existingData.faculty.equals(faculty)) {
                return existingData;
            }
        }

        FacultyData newData = new FacultyData(faculty);
        list.addLast(newData);
        return newData;
    }

    private static class FacultyData {
        String faculty;
        int totalTutors = 0;
        int totalExperience = 0;
        double totalRating = 0;
        double totalSalary = 0;

        FacultyData(String faculty) {
            this.faculty = faculty;
        }
    }
}
```

2.1.2.3.2 RatingReportCriteria

Use-Case: Generate top 5 best rated tutors.

```
package Control;
import ADT.DoublyLinkedList;
import Entity.Tutor;

public class RatingReportCriteria implements
ReportCriteriaInterface<Tutor> {

    @Override
    public String generateReport(DoublyLinkedList<Tutor> list) {
        StringBuilder report = new StringBuilder();

        // Get the size of the list
        int n = list.getSize();

        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false; // To optimize, if the inner
loop didn't do any swap, then break

            for (int j = 0; j < n - i - 1; j++) {
                Tutor tutor1 = list.get(j);
                Tutor tutor2 = list.get(j + 1);

                // Sort the tutors by rating in descending order
                if (tutor1.getRating() < tutor2.getRating()) {
                    // Swap tutor1 and tutor2
                    list.set(j, tutor2);
                    list.set(j + 1, tutor1);
                    swapped = true;
                }
            }

            // If no two elements were swapped in the inner loop,
then break
            if (!swapped) {
                break;
            }
        }

        //Generate the report
        // Limit to top 5 highest rated
        int x = Math.min(5, list.getSize());
        for (int i = 0; i < x; i++) {
            Tutor tutor = list.get(i);
            report.append(String.format("| %-4d | %-15s | %-11d |
%-20s | %-20s | %-7.2f | %-10s | %-10.2f | %-18d |\n",
                                        tutor.getId(),
tutor.getName(), tutor.getExperience(),
                                        tutor.getQualifications(),
tutor.getSpecializations(),
                                        tutor.getRating(),
tutor.getFaculty(), tutor.getSalary(),
tutor.getCancellationRate()));
        }

        return report.toString();
    }
}
```

2.1.3 Boundary: MainApp

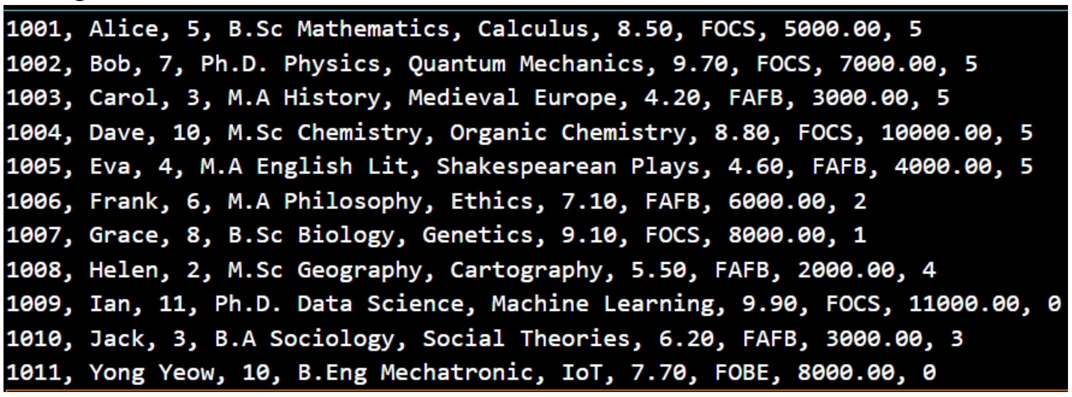
```
package  Boundary;

import  Control.*;
import  Entity.*;

import java.io.*;

import  ADT.DoublyLinkedList;

public class MainApp {
    public static TutorManager manager = new TutorManager();
    private static BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

    public static void main(String[] args) throws IOException {
        readTutorsFromFile();

        boolean continueRunning = true;
        while (continueRunning) {
            System.out.println("Tutor Management System");
            System.out.println("1. Add Tutor");
            System.out.println("2. Remove Tutor");
            System.out.println("3. Find Tutor");
            System.out.println("4. Amend Tutor");
            System.out.println("5. List All Tutors");
            System.out.println("6. Filter Tutors based on
criteria");
            System.out.println("7. Generate Report");
            System.out.println("8. Commit Tutors to File");
            System.out.println("9. Exit");
            System.out.println("0. Course Management System");
            try{
                System.out.print("Enter your choice: ");
                int choice = Integer.parseInt(reader.readLine());
                switch (choice) {
                    case 1:
                        addTutor();
                        break;
                    case 2:
                        removeTutor();
                        break;
                    case 3:
                        findTutor();
                        break;
                    case 4:
                        amendTutor();
                        break;
                    case 5:
                        listAllTutors();
                        break;
                    case 6:
                        filterSubMenu();
                        break;
                    case 7:
                        generateReportSubMenu();
                        break;
                    case 8:
                        commitTutorsToFile();
                    case 9:
                        continueRunning = false;
                    break;

                    case 0:
                        Main.mainCourse(args);
                        break;
                    default:
                        System.out.println("Invalid choice. Please 
try again.");
                }
            }catch(NumberFormatException e){
                System.out.println("Invalid choice. Please try 
again.");
            }
        }
    }

    /***********************************************R/W Methods 
Start*****************************************************/
    public static void readTutorsFromFile() {
        try (BufferedReader br = new BufferedReader(new 
FileReader("tutors.txt"))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] values = line.split(",");
                // Skip the first value (ID) and start reading 
from the second value (Name)
                // Assuming the order is ID, Name, Experience, 
Qualifications, Specializations, Rating, Faculty, Salary, 
CancellationRate
                String name = values[1].trim();
                int experience = 
Integer.parseInt(values[2].trim());
                String qualifications = values[3].trim();
                String specializations = values[4].trim();
                double rating = 
Double.parseDouble(values[5].trim());
                String faculty = values[6].trim();
                double salary = 
Double.parseDouble(values[7].trim());
                int cancellationRate = 
Integer.parseInt(values[8].trim());

                Tutor tutor = new Tutor(name, experience, 
qualifications, specializations, rating, faculty, salary, 
cancellationRate);
                manager.addTutor(tutor);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void commitTutorsToFile() {
        try (BufferedWriter bw = new BufferedWriter(new
FileWriter("tutors.txt"))) {
            Iterable<Tutor> allTutors = manager.getAllTutors();
// Assuming getAllTutors() returns an Iterable<Tutor>
            for (Tutor tutor : allTutors) {
                bw.write(String.format("%d, %s, %d, %s, %s, %.2f,
%s, %.2f, %d\n",
                                       tutor.getId(),
tutor.getName(), tutor.getExperience(),
                                       tutor.getQualifications(),
tutor.getSpecializations(),
                                       tutor.getRating(),
tutor.getFaculty(), tutor.getSalary(),
tutor.getCancellationRate()));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    /***********************************************R/W Methods
End*****************************************************/

    private static void addTutor() throws IOException {
        System.out.print("Enter Tutor Name: ");
        String name = reader.readLine();
        System.out.print("Enter Tutor Experience: ");
        int experience = Integer.parseInt(reader.readLine());
        System.out.print("Enter Tutor Qualifications: ");
        String qualifications = reader.readLine();
        System.out.print("Enter Tutor Specializations: ");
        String specializations = reader.readLine();
        System.out.print("Enter Tutor Rating: ");
        double rating = Double.parseDouble(reader.readLine());
        System.out.print("Enter Tutor Faculty: ");
        String faculty = reader.readLine();
        System.out.print("Enter the salary of the tutor: ");
        double salary = Double.parseDouble(reader.readLine());
        System.out.print("Enter the cancellation rate of the 
tutor: ");
        int cancellationRate = 
Integer.parseInt(reader.readLine());
        manager.addTutor(new Tutor(name, experience, 
qualifications, specializations, rating, faculty,salary, 
cancellationRate));
        System.out.println("Tutor added successfully.");
        System.out.println("Press Enter to 
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    private static void removeTutor() throws IOException {
        System.out.print("Enter Tutor ID to remove: ");
        int id = Integer.parseInt(reader.readLine());
        manager.removeTutorById(id);  // Call the new method
        System.out.println("Tutor removed successfully.");
        System.out.println("Press Enter to 
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    private static void findTutor() throws IOException {
        // Prompt the user to enter a Tutor ID
        System.out.print("Enter Tutor ID to find: ");
        int id = Integer.parseInt(reader.readLine());

        // Add another horizontal line for visual separation

System.out.println("--------------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Display table headers
        System.out.printf("| %-4s | %-15s | %-11s | %-20s | %-20s
| %-7s | %-10s | %-10s | %-18s |\n",
                          "ID", "Name", "Experience",
"Qualifications", "Specializations", "Rating", "Faculty",
"Salary", "Cancellation Rate");

        // Add another horizontal line for visual separation

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        Tutor tutor = manager.findTutorById(id);
        if (tutor != null) {
            // Display details of the found tutor
            System.out.printf("| %-4d | %-15s | %-11d | %-20s |
%-20s | %-7.2f | %-10s | %-10.2f | %-18d |\n",
                              tutor.getId(), tutor.getName(),
tutor.getExperience(),
                              tutor.getQualifications(),
tutor.getSpecializations(),
                              tutor.getRating(),
tutor.getFaculty(), tutor.getSalary(),
tutor.getCancellationRate());
        } else {
            System.out.println("Tutor not found.");
        }

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");
        System.out.print("Press Enter to
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    private static void amendTutor() throws IOException {
        System.out.print("Enter Tutor ID to amend: ");
        int id = Integer.parseInt(reader.readLine());
        System.out.print("Enter new name: ");
        String newName = reader.readLine();
        System.out.print("Enter new experience: ");
        int newExperience = Integer.parseInt(reader.readLine());
        System.out.print("Enter new qualifications: ");
        String newQualifications = reader.readLine();
        System.out.print("Enter new specializations: ");
        String newSpecializations = reader.readLine();
        System.out.print("Enter new rating: ");
        double newRating = Double.parseDouble(reader.readLine());
        System.out.print("Enter new faculty: ");
        String newFaculty = reader.readLine();
        System.out.print("Enter new salary: ");
        double newSalary = Double.parseDouble(reader.readLine());
        System.out.print("Enter new cancellation rate: ");
        int newCancellationRate =
Integer.parseInt(reader.readLine());

        manager.amendTutor(id, newName, newExperience,
newQualifications, newSpecializations, newRating, newFaculty,
newSalary, newCancellationRate);
        System.out.println("Tutor details amended successfully.");
        System.out.println("Press Enter to
continue...............");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    public static void listAllTutors() throws IOException {
        // Extend the line to fit the new columns

System.out.println("-----------------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Update the header to include new columns and remove
'Subject'
        System.out.printf("| %-4s | %-15s | %-11s | %-20s | %-20s
| %-7s | %-10s | %-10s | %-18s |\n",
                          "ID", "Name", "Experience",
"Qualifications", "Specializations", "Rating", "Faculty",
"Salary", "Cancellation Rate");

        // Extend the line to fit the new columns
```

```
System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Using the iterable feature of DoublyLinkedList
        for (Tutor tutor : manager.getAllTutors()) {
            // Update the format string to include new columns and
remove 'Subject'
            System.out.printf("| %-4d | %-15s | %-11d | %-20s |
%-20s | %-7.2f | %-10s | %-10.2f | %-18d |\n",
                              tutor.getId(), tutor.getName(),
tutor.getExperience(),
                              tutor.getQualifications(),
tutor.getSpecializations(),
                              tutor.getRating(),
tutor.getFaculty(), tutor.getSalary(),
tutor.getCancellationRate());
        }

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        System.out.println("Press Enter to
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    /***********************************************Filtering
Methods
Start*****************************************************/

    private static void
printFilteredTutors(DoublyLinkedList<Tutor> filteredTutors) {
        if (filteredTutors.isEmpty()) {
            System.out.println("No tutors found based on the
filter criteria.");
            return;
        }
        // Table header

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");
        System.out.printf("| %-4s | %-15s | %-11s | %-20s | %-20s
| %-7s | %-10s | %-10s | %-18s |\n",
                          "ID", "Name", "Experience",
"Qualifications", "Specializations", "Rating", "Faculty",
"Salary", "Cancellation Rate");

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        for (Tutor tutor : filteredTutors) {
            System.out.printf("| %-4d | %-15s | %-11d | %-20s |
%-20s | %-7.2f | %-10s | %-10.2f | %-18d |\n",
                              tutor.getId(), tutor.getName(),
tutor.getExperience(),
                              tutor.getQualifications(),
tutor.getSpecializations(),
                              tutor.getRating(),
tutor.getFaculty(), tutor.getSalary(),
tutor.getCancellationRate());
        }

        // Table footer

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");
    }

    private static void filterTutorsByRating() throws IOException
{
    System.out.print("\tEnter the minimum rating for tutors (e.g.
4.5): ");
    double minRating = Double.parseDouble(reader.readLine());

    FilterCriteriaInterface<Tutor> criteria = new
```

```
RatingFilterCriteria(minRating);
    DoublyLinkedList<Tutor> filteredTutors =
manager.filterTutors(criteria);

    printFilteredTutors(filteredTutors);

    System.out.println("Press Enter to continue................");
    reader.readLine();
    System.out.print("\033[H\033[2J");
}

    private static void filterTutorsByFaculty() throws IOException
{
        System.out.print("\t\tEnter the faculty to filter tutors
by (e.g. FOCS): ");
        String faculty = reader.readLine();

        FilterCriteriaInterface<Tutor> criteria = new
FacultyFilterCriteria(faculty);
        DoublyLinkedList<Tutor> filteredTutors =
manager.filterTutors(criteria);

        printFilteredTutors(filteredTutors);

        System.out.println("Press Enter to
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    private static void filterSubMenu() throws IOException {
        System.out.println("\tFilter Tutors By:");
        System.out.println("\t1. Minimum Rating");
        System.out.println("\t2. Faculty");
        try {
            System.out.print("\tPlease enter your choice: ");

            int filterChoice =
Integer.parseInt(reader.readLine());

            switch (filterChoice) {
                case 1:
                    filterTutorsByRating();
                    break;
                case 2:
                    filterTutorsByFaculty();
                    break;
                default:
                    System.out.println("Invalid choice. Please try 
again.");
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please try 
again.");
        }
    }
    /**********************************************Filtering 
Methods End***************************************************/
    private static void generateReportSubMenu() throws IOException 
{
        System.out.println("\tChoose a Report Type:");
        System.out.println("\t1. Sort Tutors By Rating");
        System.out.println("\t2. Faculty Summary");
        try{
            System.out.print("\tPlease enter your choice: ");

            int reportChoice = 
Integer.parseInt(reader.readLine());

            switch (reportChoice) {
                case 1:
                    generateSortByRatingReport();
                    break;
                case 2:
                    generateFacultySummaryReport();
                    break;
                default:
                    System.out.println("Invalid choice. Please try 
again.");
            }
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please try 
again.");
        }
    }

    private static void generateSortByRatingReport() throws
IOException {
        RatingReportCriteria sortByRatingReport = new
RatingReportCriteria();
        String report =
manager.generateReport(sortByRatingReport);

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Print the title of the report
        System.out.println("|
Top 5 Highest Rated Tutors
|");

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Update the header to include new columns
        System.out.printf("| %-4s | %-15s | %-11s | %-20s | %-20s
| %-7s | %-10s | %-10s | %-18s |\n",
                          "ID", "Name", "Experience",
"Qualifications", "Specializations", "Rating", "Faculty",
"Salary", "Cancellation Rate");

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Print the body of the table
        System.out.print(report);
        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
------------------------------------------------------------------
-------------------------------");

        // Wait for user to press Enter to continue
        System.out.println("Press Enter to
continue................");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

    private static void generateFacultySummaryReport() throws
IOException {
        FacultyReportCriteria facultyReportCriteria = new
FacultyReportCriteria();
        String report =
manager.generateReport(facultyReportCriteria);

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
---------------------------------------");
        System.out.println("|
Faculty Summary                                   |");

System.out.println("----------------------------------------------
---------------------------------------");

        // Update the header to include new columns
        System.out.printf("| %-10s | %-13s | %-18s | %-14s | %-12s
|\n",
                          "Faculty", "Total Tutors", "Average
Experience", "Average Rating", "Average Salary");

        // Extend the line to fit the new columns

System.out.println("----------------------------------------------
---------------------------------------");

        // Print the body of the table
        System.out.print(report);

        // Extend the line to fit the new columns

System.out.println("---------------------------------------------------
--------------------------------------");

        // Wait for the user to press Enter to continue
        System.out.println("Press Enter to
continue...............");
        reader.readLine();
        System.out.print("\033[H\033[2J");
    }

}
```

2.1.4 Screenshots (Boon Yong Yeow)

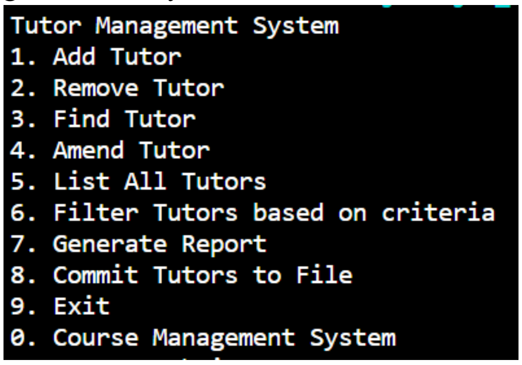
Pre-existing Tutors:



```
1001, Alice, 5, B.Sc Mathematics, Calculus, 8.50, FOCS, 5000.00, 5
1002, Bob, 7, Ph.D. Physics, Quantum Mechanics, 9.70, FOCS, 7000.00, 5
1003, Carol, 3, M.A History, Medieval Europe, 4.20, FAFB, 3000.00, 5
1004, Dave, 10, M.Sc Chemistry, Organic Chemistry, 8.80, FOCS, 10000.00, 5
1005, Eva, 4, M.A English Lit, Shakespearean Plays, 4.60, FAFB, 4000.00, 5
1006, Frank, 6, M.A Philosophy, Ethics, 7.10, FAFB, 6000.00, 2
1007, Grace, 8, B.Sc Biology, Genetics, 9.10, FOCS, 8000.00, 1
1008, Helen, 2, M.Sc Geography, Cartography, 5.50, FAFB, 2000.00, 4
1009, Ian, 11, Ph.D. Data Science, Machine Learning, 9.90, FOCS, 11000.00, 0
1010, Jack, 3, B.A Sociology, Social Theories, 6.20, FAFB, 3000.00, 3
1011, Yong Yeow, 10, B.Eng Mechatronic, IoT, 7.70, FOBE, 8000.00, 0
```

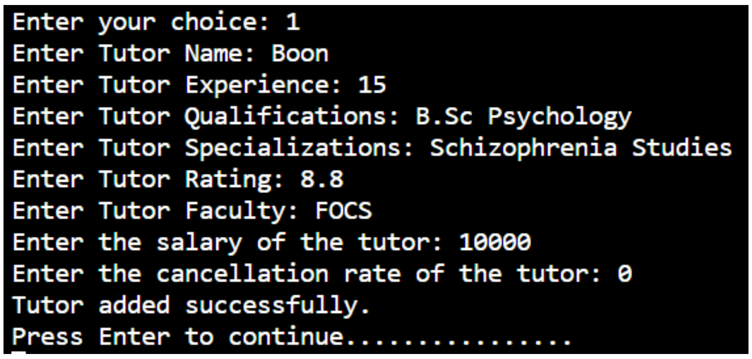
11 Tutors are hardcoded within a text file and read in at the start of the program.

Main Menu of Tutor Management Subsystem:



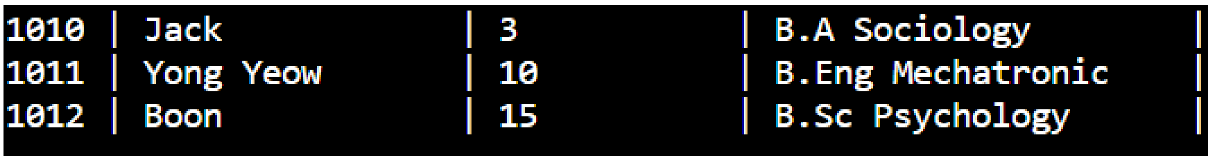
```
Tutor Management System
1. Add Tutor
2. Remove Tutor
3. Find Tutor
4. Amend Tutor
5. List All Tutors
6. Filter Tutors based on criteria
7. Generate Report
8. Commit Tutors to File
9. Exit
0. Course Management System
```

Function 1: Add Tutors



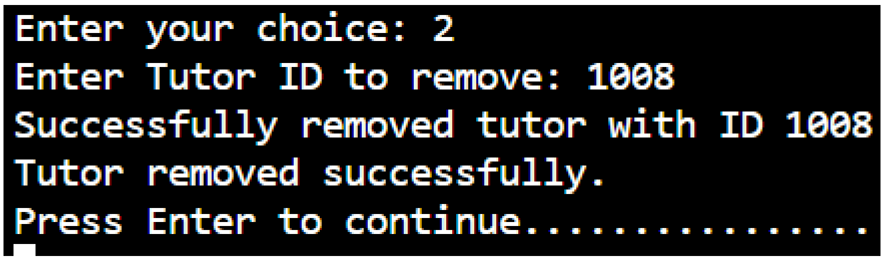
```
Enter your choice: 1
Enter Tutor Name: Boon
Enter Tutor Experience: 15
Enter Tutor Qualifications: B.Sc Psychology
Enter Tutor Specializations: Schizophrenia Studies
Enter Tutor Rating: 8.8
Enter Tutor Faculty: FOCS
Enter the salary of the tutor: 10000
Enter the cancellation rate of the tutor: 0
Tutor added successfully.
Press Enter to continue...............
```

Tutor added into collection ADT:



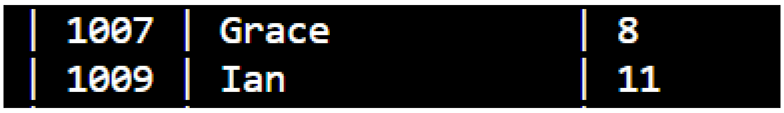
```
1010 | Jack             | 3         | B.A Sociology        |
1011 | Yong Yeow        | 10        | B.Eng Mechatronic    |
1012 | Boon             | 15        | B.Sc Psychology      |
```

Function 2: Remove Tutors



```
Enter your choice: 2
Enter Tutor ID to remove: 1008
Successfully removed tutor with ID 1008
Tutor removed successfully.
Press Enter to continue...............
```

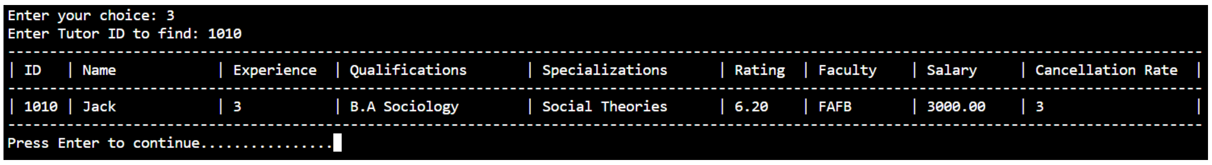
Tutor with ID = 1008 Removed:



```
| 1007 | Grace              | 8
| 1009 | Ian                | 11
```

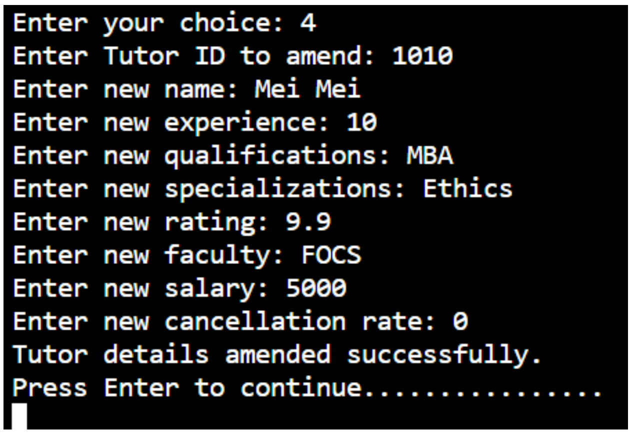
Function 3:

Tutor with ID 1010 Retrieved:



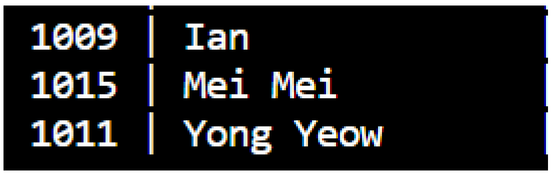
```
Enter your choice: 3
Enter Tutor ID to find: 1010
-----------------------------------------------------------------------------------------------------------------------------------------------
| ID   | Name              | Experience  | Qualifications       | Specializations      | Rating  | Faculty    | Salary     | Cancellation Rate  |
-----------------------------------------------------------------------------------------------------------------------------------------------
| 1010 | Jack              | 3           | B.A Sociology        | Social Theories      | 6.20    | FAFB       | 3000.00    | 3                  |
-----------------------------------------------------------------------------------------------------------------------------------------------
Press Enter to continue...............
```

Function 4: Amend Tutor Details



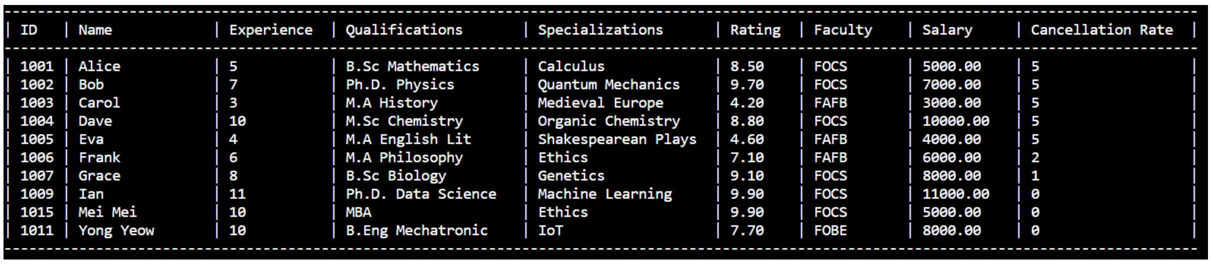
```
Enter your choice: 4
Enter Tutor ID to amend: 1010
Enter new name: Mei Mei
Enter new experience: 10
Enter new qualifications: MBA
Enter new specializations: Ethics
Enter new rating: 9.9
Enter new faculty: FOCS
Enter new salary: 5000
Enter new cancellation rate: 0
Tutor details amended successfully.
Press Enter to continue...............
```

Tutor with Tutor ID = 1010 is modified.



```
1009 | Ian
1015 | Mei Mei
1011 | Yong Yeow
```

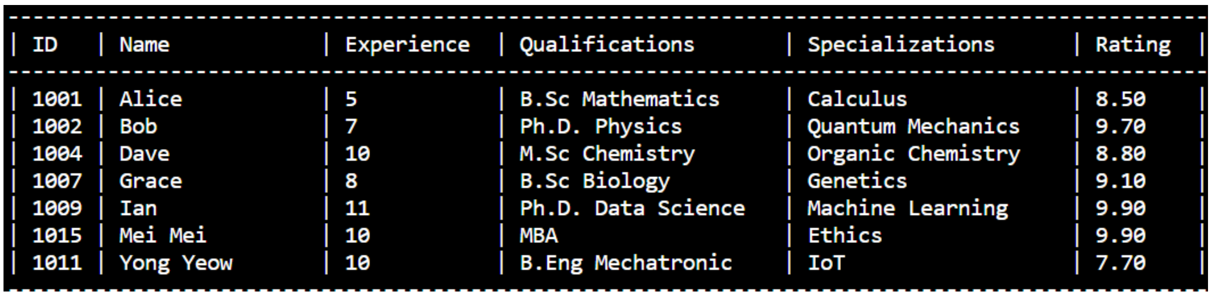
Function 5: List All Tutors



```
-------------------------------------------------------------------------------------------------------------------------------------------
| ID   | Name          | Experience | Qualifications      | Specializations     | Rating | Faculty | Salary    | Cancellation Rate |
-------------------------------------------------------------------------------------------------------------------------------------------
| 1001 | Alice         | 5          | B.Sc Mathematics    | Calculus            | 8.50   | FOCS    | 5000.00   | 5                 |
| 1002 | Bob           | 7          | Ph.D. Physics       | Quantum Mechanics   | 9.70   | FOCS    | 7000.00   | 5                 |
| 1003 | Carol         | 3          | M.A History         | Medieval Europe     | 4.20   | FAFB    | 3000.00   | 5                 |
| 1004 | Dave          | 10         | M.Sc Chemistry      | Organic Chemistry   | 8.80   | FOCS    | 10000.00  | 5                 |
| 1005 | Eva           | 4          | M.A English Lit     | Shakespearean Plays | 4.60   | FAFB    | 4000.00   | 5                 |
| 1006 | Frank         | 6          | M.A Philosophy      | Ethics              | 7.10   | FAFB    | 6000.00   | 2                 |
| 1007 | Grace         | 8          | B.Sc Biology        | Genetics            | 9.10   | FOCS    | 8000.00   | 1                 |
| 1009 | Ian           | 11         | Ph.D. Data Science  | Machine Learning    | 9.90   | FOCS    | 11000.00  | 0                 |
| 1015 | Mei Mei       | 10         | MBA                 | Ethics              | 9.90   | FOCS    | 5000.00   | 0                 |
| 1011 | Yong Yeow     | 10         | B.Eng Mechatronic   | IoT                 | 7.70   | FOBE    | 8000.00   | 0                 |
-------------------------------------------------------------------------------------------------------------------------------------------
```

Function 6: Filter by Criteria

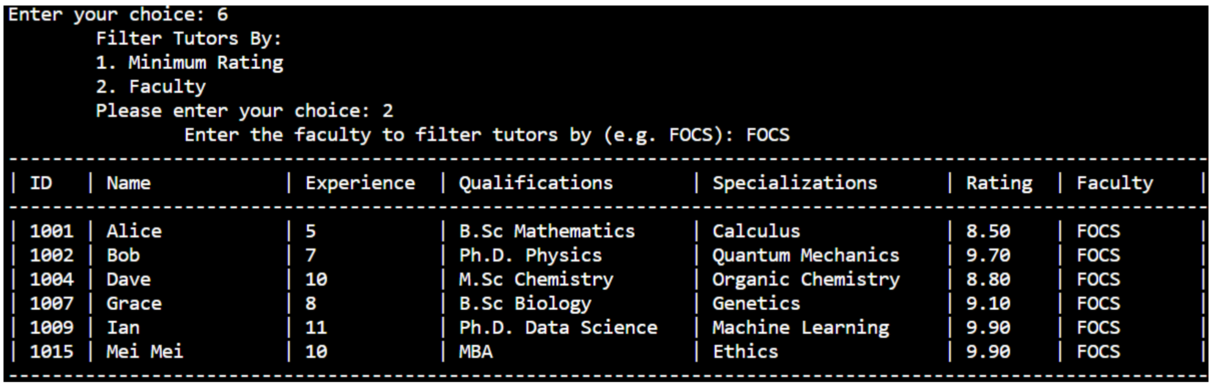
Filter by Rating:



```
-----------------------------------------------------------------------------------------------------
| ID   | Name          | Experience | Qualifications      | Specializations     | Rating |
-----------------------------------------------------------------------------------------------------
| 1001 | Alice         | 5          | B.Sc Mathematics    | Calculus            | 8.50   |
| 1002 | Bob           | 7          | Ph.D. Physics       | Quantum Mechanics   | 9.70   |
| 1004 | Dave          | 10         | M.Sc Chemistry      | Organic Chemistry   | 8.80   |
| 1007 | Grace         | 8          | B.Sc Biology        | Genetics            | 9.10   |
| 1009 | Ian           | 11         | Ph.D. Data Science  | Machine Learning    | 9.90   |
| 1015 | Mei Mei       | 10         | MBA                 | Ethics              | 9.90   |
| 1011 | Yong Yeow     | 10         | B.Eng Mechatronic   | IoT                 | 7.70   |
-----------------------------------------------------------------------------------------------------
```

- Only Tutors with a minimum rating of 7.5 are listed.

Filter By Faculty:

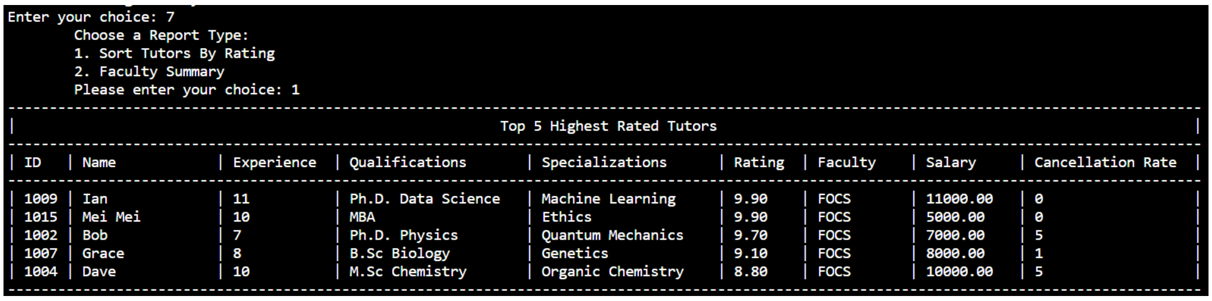


```
Enter your choice: 6
        Filter Tutors By:
        1. Minimum Rating
        2. Faculty
        Please enter your choice: 2
                Enter the faculty to filter tutors by (e.g. FOCS): FOCS
--------------------------------------------------------------------------------------------------------------
| ID   | Name              | Experience  | Qualifications        | Specializations       | Rating  | Faculty    |
--------------------------------------------------------------------------------------------------------------
| 1001 | Alice             | 5           | B.Sc Mathematics      | Calculus              | 8.50    | FOCS       |
| 1002 | Bob               | 7           | Ph.D. Physics         | Quantum Mechanics     | 9.70    | FOCS       |
| 1004 | Dave              | 10          | M.Sc Chemistry        | Organic Chemistry     | 8.80    | FOCS       |
| 1007 | Grace             | 8           | B.Sc Biology          | Genetics              | 9.10    | FOCS       |
| 1009 | Ian               | 11          | Ph.D. Data Science    | Machine Learning      | 9.90    | FOCS       |
| 1015 | Mei Mei           | 10          | MBA                   | Ethics                | 9.90    | FOCS       |
--------------------------------------------------------------------------------------------------------------
```

- Only FOCS tutors are listed.

Function 7: Report Generation

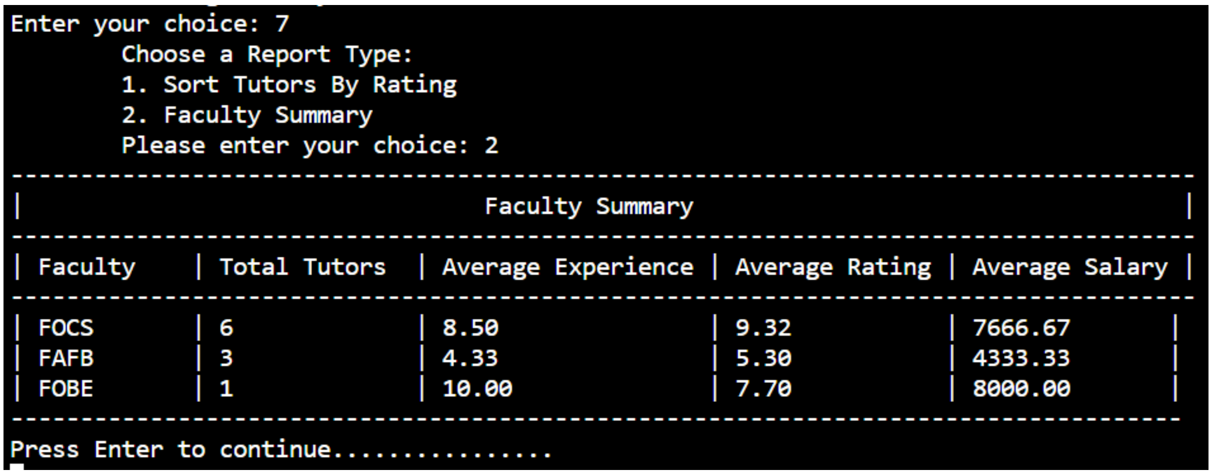
Report 1: Top 5 Highest Rated Tutors



```
Enter your choice: 7
        Choose a Report Type:
        1. Sort Tutors By Rating
        2. Faculty Summary
        Please enter your choice: 1
------------------------------------------------------------------------------------------------------------------------------------------
|                                                    Top 5 Highest Rated Tutors                                                          |
------------------------------------------------------------------------------------------------------------------------------------------
| ID   | Name              | Experience  | Qualifications        | Specializations       | Rating  | Faculty    | Salary       | Cancellation Rate  |
------------------------------------------------------------------------------------------------------------------------------------------
| 1009 | Ian               | 11          | Ph.D. Data Science    | Machine Learning      | 9.90    | FOCS       | 11000.00     | 0                  |
| 1015 | Mei Mei           | 10          | MBA                   | Ethics                | 9.90    | FOCS       | 5000.00      | 0                  |
| 1002 | Bob               | 7           | Ph.D. Physics         | Quantum Mechanics     | 9.70    | FOCS       | 7000.00      | 5                  |
| 1007 | Grace             | 8           | B.Sc Biology          | Genetics              | 9.10    | FOCS       | 8000.00      | 1                  |
| 1004 | Dave              | 10          | M.Sc Chemistry        | Organic Chemistry     | 8.80    | FOCS       | 10000.00     | 5                  |
------------------------------------------------------------------------------------------------------------------------------------------
```

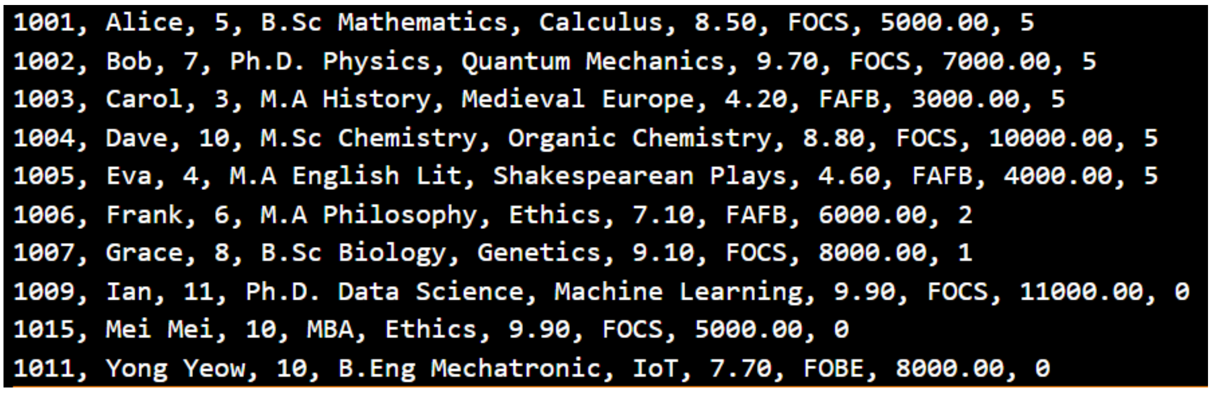
- Top 5 Highest Rated tutors report is generated.

Report 2: Faculty Summary Report



```
Enter your choice: 7
        Choose a Report Type:
        1. Sort Tutors By Rating
        2. Faculty Summary
        Please enter your choice: 2
-----------------------------------------------------------------------------------------
|                                   Faculty Summary                                     |
-----------------------------------------------------------------------------------------
| Faculty     | Total Tutors | Average Experience | Average Rating | Average Salary |
-----------------------------------------------------------------------------------------
| FOCS        | 6            | 8.50               | 9.32           | 7666.67        |
| FAFB        | 3            | 4.33               | 5.30           | 4333.33        |
| FOBE        | 1            | 10.00              | 7.70           | 8000.00        |
-----------------------------------------------------------------------------------------
Press Enter to continue................
```

Function 8: Commit Changes to file:



```
1001, Alice, 5, B.Sc Mathematics, Calculus, 8.50, FOCS, 5000.00, 5
1002, Bob, 7, Ph.D. Physics, Quantum Mechanics, 9.70, FOCS, 7000.00, 5
1003, Carol, 3, M.A History, Medieval Europe, 4.20, FAFB, 3000.00, 5
1004, Dave, 10, M.Sc Chemistry, Organic Chemistry, 8.80, FOCS, 10000.00, 5
1005, Eva, 4, M.A English Lit, Shakespearean Plays, 4.60, FAFB, 4000.00, 5
1006, Frank, 6, M.A Philosophy, Ethics, 7.10, FAFB, 6000.00, 2
1007, Grace, 8, B.Sc Biology, Genetics, 9.10, FOCS, 8000.00, 1
1009, Ian, 11, Ph.D. Data Science, Machine Learning, 9.90, FOCS, 11000.00, 0
1015, Mei Mei, 10, MBA, Ethics, 9.90, FOCS, 5000.00, 0
1011, Yong Yeow, 10, B.Eng Mechatronic, IoT, 7.70, FOBE, 8000.00, 0
```

2.2 Lim Meng Fu

Student Name	Student ID	Prog / Tut.Grp	Signature
Lim Meng Fu	22PMR05891	RDS2G1	[illegible]

Subsystem Chosen : **Course Assignment**

- Assign courses to FOCS or FAFB programmes.
- Assign tutors to a course.
- Edit some details about courses/programmes.

2.2.1 Entity

2.2.1.1 Course

```
package Entity;

import Control.*;

public class Course {

    private String courseId;
    private String courseName;
    private Tutor tutor;

    public static int courseCount = 0;

    public Course() {
        this("", "");
    }

    public Course(String courseId, String courseName) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.tutor = null;

        courseCount++;
    }


    public Course(String courseId, String courseName, Tutor tutor) {
        this.courseId = courseId;
        this.courseName = courseName;
        this.tutor = tutor;

        courseCount++;
    }

    //setter
    public void setCourseId(String courseId) {
        this.courseId = courseId;
    }

    public void setCourseName(String courseName) {
        this.courseName = courseName;
    }

    public void setTutor(Tutor tutor) {
        this.tutor = tutor;
    }



    //getter
    public String getCourseId() {
        return courseId;
    }

    public String getCourseName() {
        return courseName;
    }

    public Tutor getTutor() {
        return tutor;
    }

    //toString

    public String toString() {
        if(tutor != null)
            return courseId + " -" + courseName + ", tutor=" +
tutor.getName()+ "}\n";
        else
            return courseId + " -" + courseName + ", tutor=" + "No tutor
assigned" + "}\n";
    }
}
```

2.2.1.2 Programme

```
package Entity;

import ADT.*;
import Control.CourseManager;

public class Programme {

    private String programmeId;
    private String programmeName;
    private CourseManager course;

    public Programme() {
        this("", "", null);
    }

    public Programme(String programmeId, String programmeName,
CourseManager course) {
        this.programmeId = programmeId;
        this.programmeName = programmeName;
        this.course = course;
    }

    //setter
    public void setProgrammeId(String programmeId) {
        this.programmeId = programmeId;
    }

    public void setProgrammeName(String programmeName) {
        this.programmeName = programmeName;
    }

    public void setCourse(CourseManager course) {
        this.course = course;
    }

    //getter
    public String getProgrammeId() {
        return programmeId;
    }
    public String getProgrammeName() {
        return programmeName;
    }

    public CourseManager getCourse() {
        return course;
    }



    //toString
    @Override
    public String toString() {
        return "Programme{" + "programmeId=" + programmeId + ",
programmeName=" + programmeName + course.displayCourse() + "    " +
"}\n";
    }



}
```

2.2.2 Control

2.2.2.1 CourseManager

```
package Control;

import ADT.*;
import Entity.*;
import java.util.Iterator;

public class CourseManager {

    LinkedList<Course> courses;
    public static int totalCourse = 0;

    public CourseManager() {
        courses = new LinkedList<>();
    }

    //Get All
    // Add all courses from a source list to the courseList
    // public void addAll(ListInterface<Course> sourceList) {

    //    Iterator<Course> sourceListIte =
sourceList.getIterator();

    //     for (Course course : sourceListIte) {
    //         courses.add(sourceListIte);
    //     }
    // }

    public void addAll(ListInterface<Course> sourceList) {
        Iterator<Course> sourceListIte = sourceList.getIterator();

        while (sourceListIte.hasNext()) {
            Course course = sourceListIte.next();
            courses.add(course); // Add the individual course to
the courses list

            totalCourse++;
        }
    }


    public void addCourse(Course course){
        courses.add(course);

        totalCourse++;
    }

    public void removeCourse(Course course) {
        courses.remove(course);

        totalCourse--;
    }

    public void amendCourse2(Course course, Tutor tutor) {
        // Find the course by its ID
        Course existingCourse =
getCourseById(course.getCourseId());


        if (existingCourse != null) {
            // Assign the tutor to the course
            existingCourse.setTutor(tutor);
            System.out.println("Tutor " + tutor.getName() + " has
been assigned to " + course.getCourseName());
        } else {
            System.out.println("Course " + course.getCourseId() +
" not found. Tutor assignment failed.");
        }
    }

    public void amendCourseTutor(Course course, Tutor tutor) {
        // Find the course by its ID
        Course existingCourse =
getCourseById(course.getCourseId());


        if (existingCourse != null) {
            // Assign the tutor to the course
            existingCourse.setTutor(tutor);
            System.out.println("Tutor " + tutor.getName() + " has
been assigned to " + course.getCourseName());
        } else {
            System.out.println("Course (" + course.getCourseId() +
") " + course.getCourseName() + " not found. Tutor assignment
failed.");
        }
    }

    public String displayCourse() {
        return courses.toString();
    }

    public void ammendCourseName(Course course, String
newCourseName) {
        Course existingCourse =
getCourseById(course.getCourseId());
        if (existingCourse != null) {
            existingCourse.setCourseName(newCourseName);
            System.out.println("Course name has been changed to "
+ newCourseName);
        } else {
            System.out.println("Course " + course.getCourseId() +
" not found. Course name change failed.");
        }
    }

        public void amendCourse(Course course, Tutor tutor) {
        // Find the course by its ID
        Course existingCourse =
getCourseById(course.getCourseId());

        if (existingCourse != null) {
            // Assign the tutor to the course
            existingCourse.setTutor(tutor);
            System.out.println("Tutor " + tutor.getName() + " has
been assigned to " + course.getCourseName());
        } else {
            System.out.println("Course " + course.getCourseId() +
" not found. Tutor assignment failed.");
        }
    }
    // public void displayCourseByTutor() {

    // }

    // public void displayCourseByProgramme() {


    // }

    public Course getCourseById(String courseId) {
        // Get an iterator for the courses
        Iterator<Course> iterator = courses.getIterator();

        // Iterate through the courses using the iterator
        while (iterator.hasNext()) {
            Course course = iterator.next();
            if (course.getCourseId().equals(courseId)) {
                // If the course's ID matches the specified
courseId, return the course
                return course;
            }
        }

        // If no matching course is found, return null
        return null;
    }

    public Course getCourseByName(String courseName) {
        // Get an iterator for the courses
        Iterator<Course> iterator = courses.getIterator();

        // Iterate through the courses using the iterator
        while (iterator.hasNext()) {
            Course course = iterator.next();
            if (course.getCourseName().equals(courseName)) {
                // If the course's name matches the specified
courseName, return the course
                return course;
            }
        }

        // If no matching course is found, return null
        return null;
    }


}
```

2.2.3 Boundary

2.2.3.1 Main

```
package Boundary;

import ADT.*;
import Control.CourseManager;
import Entity.*;

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

import static Boundary.MainApp.*;

public class Main {

    public static CourseManager cmFOCS = new CourseManager();
    public static CourseManager cmFAFB = new CourseManager();

    private static Scanner input = new Scanner(System.in);
    public static void mainCourse(String[] args)  throws
IOException{

        ListInterface<Course> courseListFOCS = new LinkedList<>();
        ListInterface<Course> courseListFAFB = new LinkedList<>();
        ListInterface<Programme> progList = new LinkedList<>();

        readTutorsFromFile();
        //listAllTutors();

        //create course
        Course c1 = new Course("CS001", "Object-Oriented
Programming");
        Course c2 = new Course("CS002", "Data Structure and
Algorithm");
        Course c3 = new Course("CS003", "Database");
        Course c4 = new Course("AB004", "Economics");
        Course c5 = new Course("AB005", "Marketing");
        Course c6 = new Course("AB006", "Accounting");
        courseListFOCS.add(c1);
        courseListFOCS.add(c2);
        courseListFOCS.add(c3);

        cmFOCS.addAll(courseListFOCS);

        courseListFAFB.add(c4);
        courseListFAFB.add(c5);
        courseListFAFB.add(c6);

        cmFAFB.addAll(courseListFAFB);

        //cmRDS
        ListInterface<Course> courseListRDS = new LinkedList<>();
        courseListRDS.add(c1);
        courseListRDS.add(c2);
        courseListRDS.add(c3);

        CourseManager cmRDS = new CourseManager();
        cmRDS.addAll(courseListRDS);

        //cmRIB
        ListInterface<Course> courseListRIB = new LinkedList<>();
        courseListRIB.add(c4);
        courseListRIB.add(c5);
        courseListRIB.add(c6);

        CourseManager cmRIB = new CourseManager();
        cmRIB.addAll(courseListRIB);

        //cmFRCS
        ListInterface<Course> courseListRCS = new LinkedList<>();
        courseListRCS.add(c1);
        courseListRCS.add(c3);

        CourseManager cmRCS = new CourseManager();
        cmRCS.addAll(courseListRCS);



       // System.out.println("Poop\n"+cmFAFB.displayCourse());
```

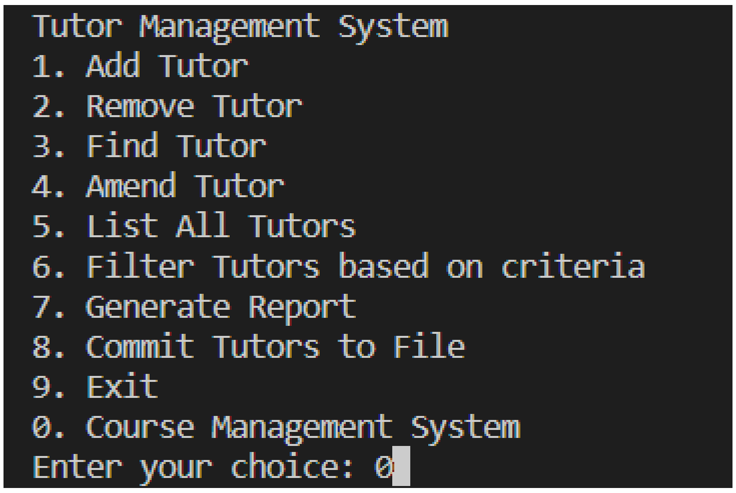
```
        //System.out.println(courseListFAFB.toString());

        //create programme
       //Programme p1 = new Programme("RDS", "Degree in Data
Science", courseListFOCS);
        //Programme p2 = new Programme("RIB", "Degree in
International Business", courseListFAFB);

        Programme p1 = new Programme("RDS", "Degree in Data
Science", cmFOCS);
        Programme p2 = new Programme("RIB", "Degree in
International Business", cmFAFB);
        Programme p3 = new Programme("RAF", "Degree in Accounting
and Finance", cmFAFB);
        Programme p4 = new Programme("RCS", "Degree in Computer
Science", cmFOCS);
        progList.add(p1);
        progList.add(p2);
        progList.add(p3);
        progList.add(p4);



        //System.out.println(MainApp.manager.toString());

        // cmFOCS.removeCourse(c3);
        // System.out.println("Removed
c3\n"+cmFOCS.displayCourse());

        // cmFOCS.addCourse(c3);
        // System.out.println("Added
c3\n"+cmFOCS.displayCourse());

        int choice = 999;

        while(choice > 0){
            System.out.print("\033[H\033[2J");
            System.out.println("Welcome to Course Management
System");
            System.out.println("1. Add Course w/o tutor
assigned");
            System.out.println("2. Assign Tutor to a Course");
            System.out.println("3. Display Course");
            System.out.println("4. Display Proggrame Courses");
            System.out.println("5. Remove Course");
            System.out.println("0. BACK TO TUTOR MANAGEMENT
SYSTEM");

            System.out.print("Enter your choice: ");
            choice = input.nextInt();

            input.nextLine(); // Consume the newline character
left in the input buffer

            switch(choice){
                    case 1:
                        System.out.print("\033[H\033[2J");
                        System.out.println("Add Course");
                        System.out.println("1. FOCS Course");
                        System.out.println("2. FAFB Course");
                        System.out.print("Enter your choice: ");
                        int choice1 = input.nextInt();
                        input.nextLine();

                        switch(choice1){
                            case 1:
                                addCourseFOCS();
                                break;
                            case 2:
                                addCourseFAFB();
                                break;
                            default:
                                System.out.println("Invalid
choice");

                                break;
                        }
                        break;

                    case 2:
                        System.out.print("\033[H\033[2J");
                        System.out.println("Assign Tutor to
Course");

                        System.out.println("1. FOCS Course");
                        System.out.println("2. FAFB Course");
                        try {
                            System.out.print("Enter your choice:
");
                            int choice2 = input.nextInt();
                            input.nextLine(); // Consume the
newline character left in the input buffer

                            switch (choice2) {
                                case 1:
                                    assignTutorCourseFOCS();
                                    System.out.println("Press
Enter to continue................");
                                    reader.readLine();

System.out.print("\033[H\033[2J");
                                    break;
                                case 2:
                                    assignTutorCourseFAFB();
                                    System.out.println("Press
Enter to continue................");
                                    reader.readLine();

System.out.print("\033[H\033[2J");
                                    break;
                                default:
                                    System.out.println("Invalid
choice");

                                    break;
                            }
                        } catch (InputMismatchException e) {
                            input.nextLine(); // Clear the invalid
input from the buffer
                            System.out.println("Invalid input.
Please enter a valid choice (1 or 2).");
                        }

                    case 3:
                        System.out.print("\033[H\033[2J");
                        System.out.println("Display Course");
                        System.out.println("1. FOCS Course");
                        System.out.println("2. FAFB Course");
                        System.out.println("3. All Course");

                        try{
                            System.out.print("Enter your choice: 
");
                            int choice3 = input.nextInt();
                            input.nextLine();

                            switch(choice3){
                                case 1:
                                    displayCourseFOCS();
                                    System.out.println("Press 
Enter to continue................");
                                    reader.readLine();
                                    
System.out.print("\033[H\033[2J");
                                    break;
                                case 2:
                                    displayCourseFAFB();
                                    System.out.println("Press 
Enter to continue................");
                                    reader.readLine();
                                    
System.out.print("\033[H\033[2J");
                                    break;

                                case 3:
                                    System.out.println("Display 
Course");
                                    System.out.println("FOCS 
Course");
                                    
System.out.println(cmFOCS.displayCourse());
                                    System.out.println("FAFB 
Course");
                                    
System.out.println(cmFAFB.displayCourse());
                                    break;

                                default:
                                    System.out.println("Invalid 
choice");
```

```
                                    break;
                            }
                            break;

                        } catch (InputMismatchException e) {
                            input.nextLine(); // Clear the invalid
input from the buffer
                            System.out.println("Invalid input.
Please enter a valid choice (1, 2 or 3).");
                        }


                    case 4:
                        System.out.print("\033[H\033[2J");
                        System.out.println("Display Programmes");
                        System.out.println(progList.toString());
                        break;

                    case 5:
                        System.out.print("\033[H\033[2J");
                        System.out.println("Remove Course");
                        System.out.println("1. FOCS Course");
                        System.out.println("2. FAFB Course");

                        try{
                            System.out.print("Enter your choice:
");

                            int choice5 = input.nextInt();
                            input.nextLine();

                            switch(choice5){
                                case 1:
                                    System.out.print("Enter Course
ID: ");

                                    String courseId =
input.nextLine().toUpperCase();


cmFOCS.removeCourse(cmFOCS.getCourseById(courseId));
                                    System.out.println("Removed
course\n"+cmFOCS.displayCourse());
                                    System.out.println("Press
Enter to continue................");
                                    reader.readLine();

System.out.print("\033[H\033[2J");
                                    break;
                                case 2:
                                    System.out.print("Enter Course
ID: ");
                                    String courseId1 =
input.nextLine().toUpperCase();
                                    //String courseId1 = "AB" +
String.format("%03d", Course.courseCount + 1);


//System.out.println(cmFAFB.getCourseById(courseId1));


cmFAFB.removeCourse(cmFAFB.getCourseById(courseId1));
                                    System.out.println("Removed
course\n"+cmFAFB.displayCourse());
                                    System.out.println("Press
Enter to continue................");
                                    reader.readLine();

System.out.print("\033[H\033[2J");
                                    break;
                                default:
                                    System.out.println("Invalid
choice");
                                    System.out.println("Press
Enter to continue................");
                                    reader.readLine();

System.out.print("\033[H\033[2J");
                                    break;
                            }
                        break;
                        }
                        catch (InputMismatchException e) {
                            input.nextLine(); // Clear the invalid
input from the buffer
                            System.out.println("Invalid input.
Please enter a valid choice (1 or 2).");
                            System.out.println("Press Enter to 
continue................");
                            reader.readLine();
                            System.out.print("\033[H\033[2J");
                        }

                    case 0:
                        try{
                            System.out.println("Exiting...");
                            TimeUnit.SECONDS.sleep(2);
                            System.out.print("\033[H\033[2J");
                            MainApp.main(args);
                            break;
                        }catch(InterruptedException e){
                            e.printStackTrace();
                        }

                    default:
                        try{
                            System.out.println("invalid Choice.");
                            TimeUnit.MILLISECONDS.sleep(100);
                            System.out.print("\033[H\033[2J");
                            break;
                        }catch(InterruptedException e){
                            e.printStackTrace();
                        }
                }

            }


        }

    private static void addCourseFOCS() throws IOException{
        System.out.print("Enter Course Name: ");
        String courseName = input.nextLine();
        String courseId = "CS" + String.format("%03d", 
Course.courseCount + 1);

        cmFOCS.addCourse(new Course(courseId, courseName));
        System.out.println("Added new 
course\n"+cmFOCS.displayCourse());

    }


    private static void addCourseFAFB() throws IOException{
        System.out.print("Enter Course Name: ");
        String courseName = input.nextLine();
        String courseId = "AB" + String.format("%03d",
Course.courseCount + 1);

        cmFAFB.addCourse(new Course(courseId, courseName));
        System.out.println("Added new
course\n"+cmFAFB.displayCourse());

    }

    private static void assignTutorCourseFAFB() {
        System.out.print("Enter Course ID: ");
        // Consume the newline character left in the input buffer
        String courseId = input.next().toUpperCase();

        //if course id is not found, print error message
        if(cmFAFB.getCourseById(courseId) == null){
            System.out.println("Course ID not found");
            return;
        }
        //print courseId
        System.out.println("Course ID: " + courseId);

        // Find the course by name
        Course course = cmFAFB.getCourseById(courseId);
        System.out.println("Course: " + course.getCourseName() + "
(" + course.getCourseId() + ")" );

        //get tutor
        System.out.println("Assign Tutor to Course");
        System.out.print("Enter Tutor ID: ");
        int tutorId = input.nextInt();
        Tutor tutor = manager.findTutorById(tutorId);

        //Assign
        cmFAFB.amendCourseTutor(course, tutor);
    }

    private static void assignTutorCourseFOCS(){
        System.out.print("Enter Course ID: ");
        // Consume the newline character left in the input buffer
        String courseId = input.next().toUpperCase();
        input.nextLine();

        //if course id is not found, print error message
        if(cmFOCS.getCourseById(courseId) == null){
            System.out.println("Course ID not found");
            return;
        }

        //print courseId
        System.out.println("Course ID: " + courseId);

        // Find the course by name
        Course course = cmFOCS.getCourseById(courseId);
        System.out.println("Course: " + course.getCourseName() + "
(" + course.getCourseId() + ")" );

        //get tutor
        System.out.println("Assign Tutor to Course");
        System.out.print("Enter Tutor ID: ");
        int tutorId = input.nextInt();
        Tutor tutor = manager.findTutorById(tutorId);

        //Assign
        cmFOCS.amendCourseTutor(course, tutor);
    }

    private static void displayCourseFOCS(){
        System.out.println("Display Course");
        System.out.println(cmFOCS.displayCourse());
    }

    private static void displayCourseFAFB(){
        System.out.println("Display Course");
        System.out.println(cmFAFB.displayCourse());
    }
```

```



}



```

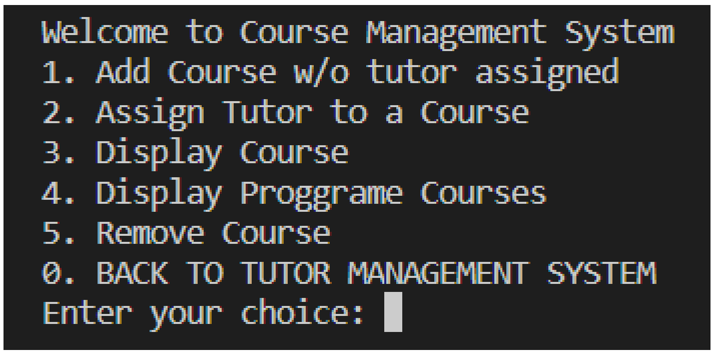
2.2.4 Screenshots

To Enter Course Management System:



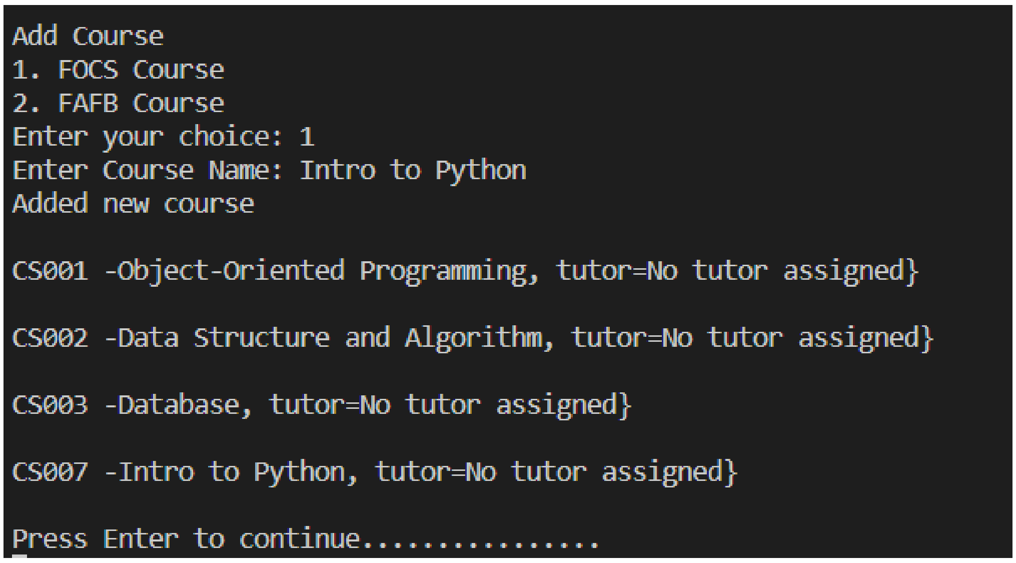
```
Tutor Management System
1. Add Tutor
2. Remove Tutor
3. Find Tutor
4. Amend Tutor
5. List All Tutors
6. Filter Tutors based on criteria
7. Generate Report
8. Commit Tutors to File
9. Exit
0. Course Management System
Enter your choice: 0
```

Menu:



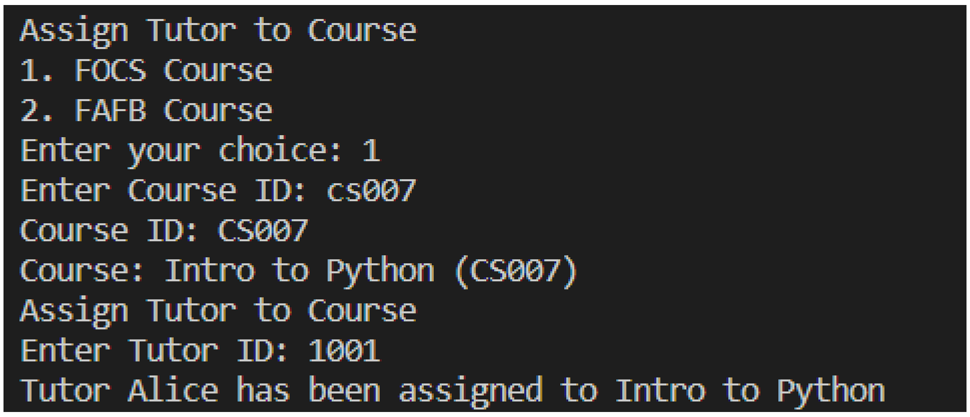
```
Welcome to Course Management System
1. Add Course w/o tutor assigned
2. Assign Tutor to a Course
3. Display Course
4. Display Proggrame Courses
5. Remove Course
0. BACK TO TUTOR MANAGEMENT SYSTEM
Enter your choice:
```

Menu Option 1: Add New Course (No tutor assigned)

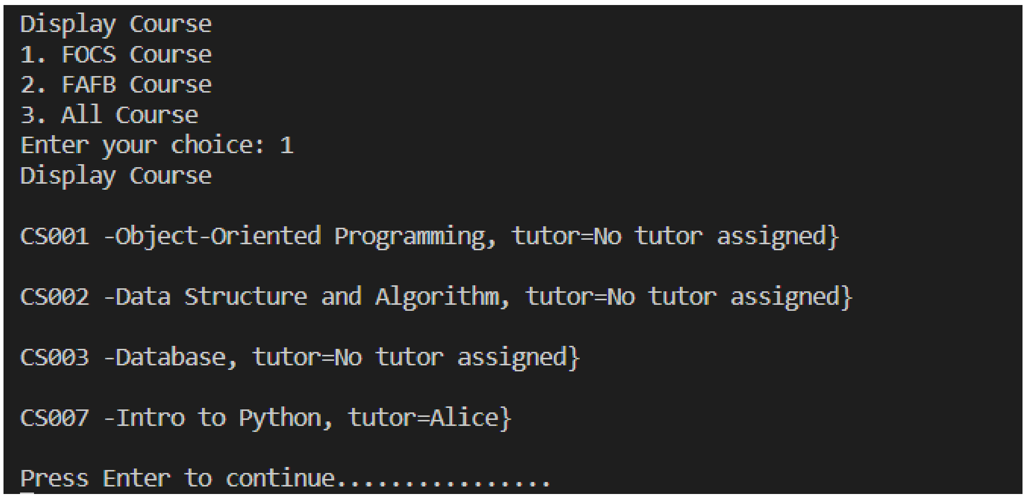


```
Add Course
1. FOCS Course
2. FAFB Course
Enter your choice: 1
Enter Course Name: Intro to Python
Added new course

CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

CS007 -Intro to Python, tutor=No tutor assigned}

Press Enter to continue...............
```

Menu Option 2: Assign Tutor To (1) Course

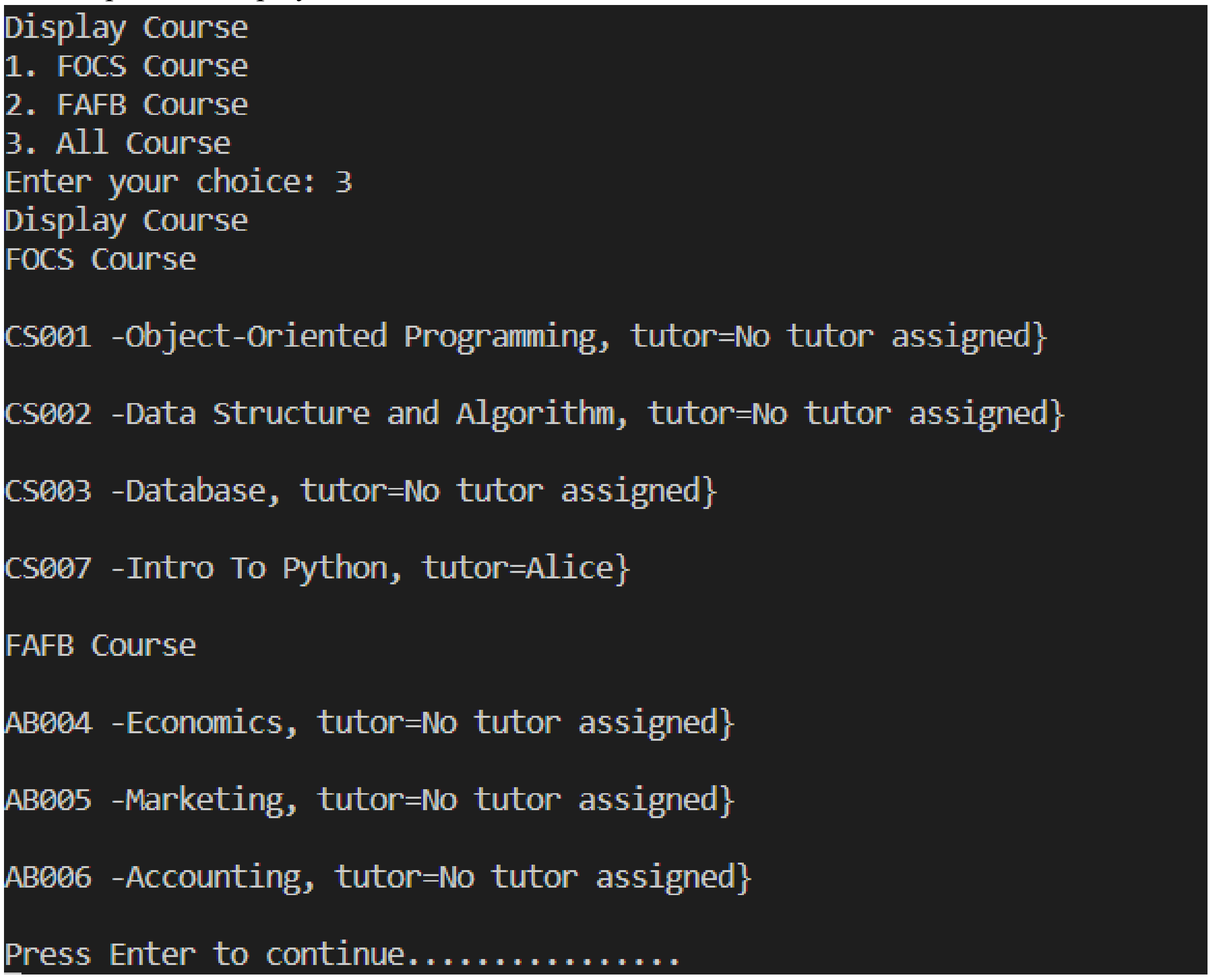


```
Assign Tutor to Course
1. FOCS Course
2. FAFB Course
Enter your choice: 1
Enter Course ID: cs007
Course ID: CS007
Course: Intro to Python (CS007)
Assign Tutor to Course
Enter Tutor ID: 1001
Tutor Alice has been assigned to Intro to Python
```



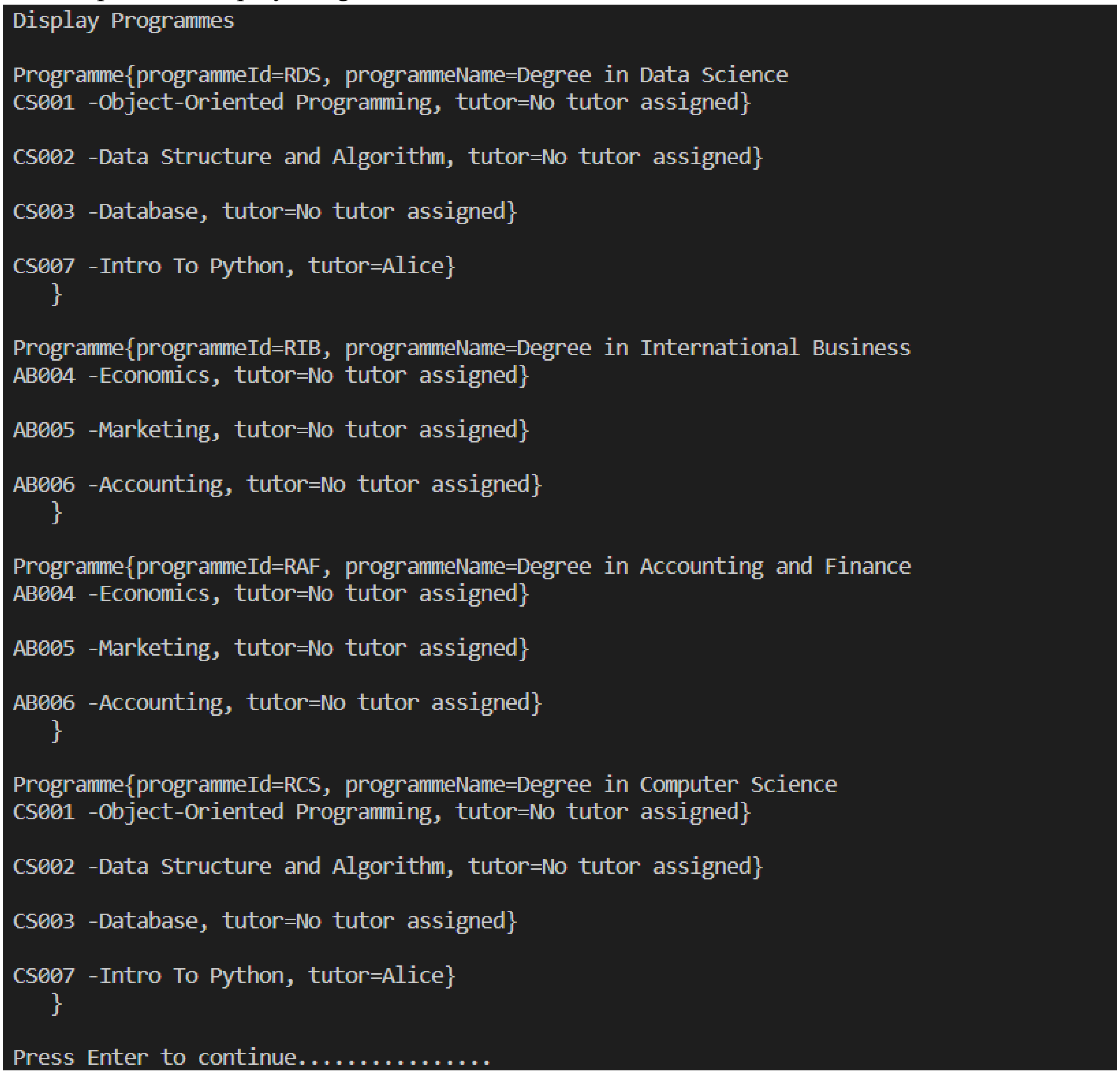
```
Display Course
1. FOCS Course
2. FAFB Course
3. All Course
Enter your choice: 1
Display Course

CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

CS007 -Intro to Python, tutor=Alice}

Press Enter to continue................
```

Menu Option 3: Display Course



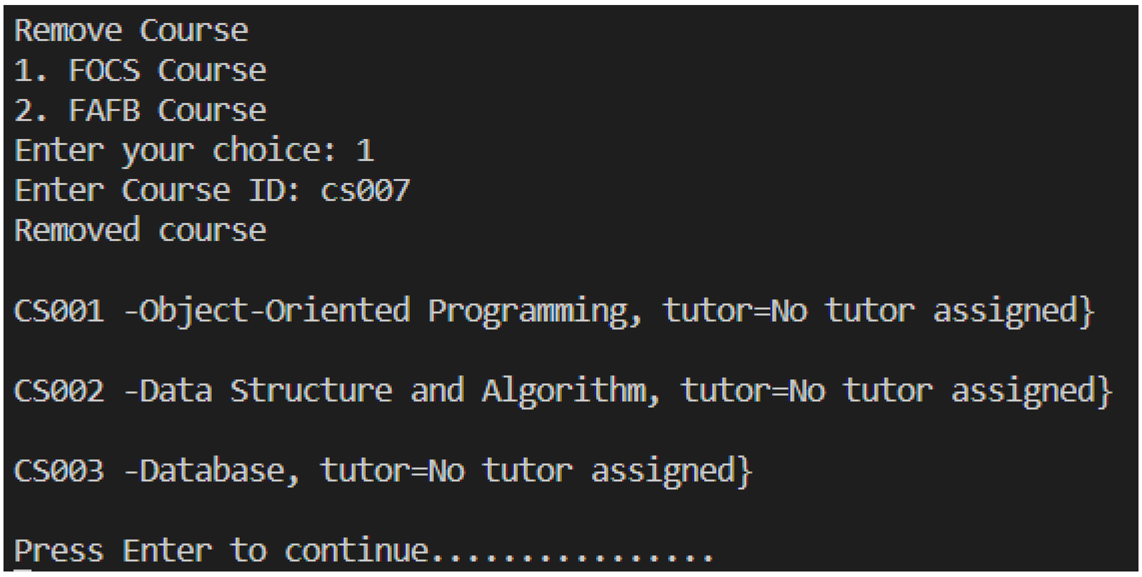
```
Display Course
1. FOCS Course
2. FAFB Course
3. All Course
Enter your choice: 3
Display Course
FOCS Course

CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

CS007 -Intro To Python, tutor=Alice}

FAFB Course

AB004 -Economics, tutor=No tutor assigned}

AB005 -Marketing, tutor=No tutor assigned}

AB006 -Accounting, tutor=No tutor assigned}

Press Enter to continue................
```

Menu Option 4: Display Programmes with their courses

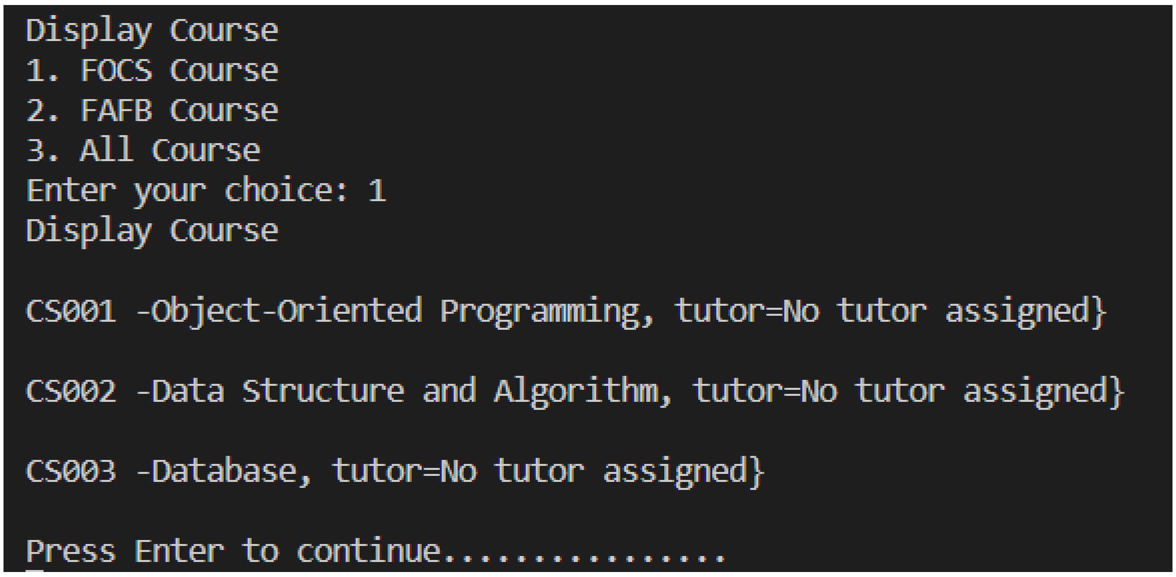


```
Display Programmes

Programme{programmeId=RDS, programmeName=Degree in Data Science
CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

CS007 -Intro To Python, tutor=Alice}
   }

Programme{programmeId=RIB, programmeName=Degree in International Business
AB004 -Economics, tutor=No tutor assigned}

AB005 -Marketing, tutor=No tutor assigned}

AB006 -Accounting, tutor=No tutor assigned}
   }

Programme{programmeId=RAF, programmeName=Degree in Accounting and Finance
AB004 -Economics, tutor=No tutor assigned}

AB005 -Marketing, tutor=No tutor assigned}

AB006 -Accounting, tutor=No tutor assigned}
   }

Programme{programmeId=RCS, programmeName=Degree in Computer Science
CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

CS007 -Intro To Python, tutor=Alice}
   }

Press Enter to continue...............
```

Menu Option 5: Remove (1) Course



```
Remove Course
1. FOCS Course
2. FAFB Course
Enter your choice: 1
Enter Course ID: cs007
Removed course

CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

Press Enter to continue................
```



```
Display Course
1. FOCS Course
2. FAFB Course
3. All Course
Enter your choice: 1
Display Course

CS001 -Object-Oriented Programming, tutor=No tutor assigned}

CS002 -Data Structure and Algorithm, tutor=No tutor assigned}

CS003 -Database, tutor=No tutor assigned}

Press Enter to continue................
```

3.0 Rubric

BACS2063 Data Structures and Algorithms - Assignment Rubrics 202305					**Team marks:**	**0**
Team members:	**Boon Yong Yeow,Lim Meng Fu**					
Team name:				**Programme /Group:**		**RDS2G1**
Assessment Criteria	**Developing 1 - 4 Marks**	**Approaching 5 - 7 Marks**	**Ideal 8 - 10 Marks**	**Marks Awarded**	**Weightage**	**Weighted Marks**
ADT specification	Includes 2 or more of the the items listed under "Approaching".	• Incomplete information OR • Unclear/inaccurate descriptions OR • Includes details of the ADT implementation or entity objects.	• Correct format AND • Complete information AND • Clear & accurate descriptions.		1	0
Collection ADT Implementation	Includes 2 or more of the the items listed under "Approaching".	• Inconsistent with ADT spec. OR • Implementation is not generic OR • Coding is not efficient or elegant.	• Consistent with ADT spec. AND • Correct use of generic types AND • Efficient & elegant coding.		1	0
Collection ADT implementation - originality	Does not demonstrate any originality of implementation & application of new knowledge.	• The ADT has not been covered in the course OR • Includes additional/modified operations that are non-trivial and meaningful.	• The ADT is not covered in the course OR • The implementation approach is different from the sample code.		1.5	0

Comments:

BACS2063 Data Structures and Algorithms - Assignment Rubrics 202305					**Total Mark**	**0**
			Student's name:	**Boon Yong Yeow**		
Team name:			**Programme/Group:**	**RDS2G1**		

Assessment Criteria	Developing 1 - 4 Marks	Approaching 5 - 7 Marks	Ideal 8 - 10 Marks	Marks Awarded	Weightage	Weighted Marks
ADT specification	Includes 2 or more of the the items listed under "Approaching".	• Incomplete information OR • Unclear/inaccurate descriptions OR • Includes details of the ADT implementation or entity objects.	• Correct format AND • Complete information AND • Clear & accurate descriptions.		1	0
Collection ADT Implementation	Includes 2 or more of the the items listed under "Approaching".	• Inconsistent with ADT spec. OR • Implementation is not generic OR • Coding is not efficient or elegant.	• Consistent with ADT spec. AND • Correct use of generic types AND • Efficient & elegant coding.		1	0
Collection ADT implementation - originality	Does not demonstrate any originality of implementation & application of new knowledge.	• The ADT has not been covered in the course OR • Includes additional/modified operations that are non-trivial and meaningful.	• The ADT is not covered in the course OR • The implementation approach is different from the sample code.		1.5	0
Use of ADTs - either in entity and/or control class(es)	• Inappropriate selection of ADT AND • Inappropriate use of ADT.	• Appropriate selection of ADT OR • Appropriate use of ADT.	• Appropriate selection of ADT AND • Appropriate use of ADT.		1.5	0
Use of ADTs - creativity & complexity	The functionalities/features in the entity and/or control classes do not demonstrate any use of ADTs.	The functionalities/features in the entity and/or control classes are creatively designed to demonstrate the use of the ADT.	The functionalities/features in the entity and/or control classes are **very** creatively designed to demonstrate the use of the ADT.		2	0
Overall solution	• There is one or more major bugs AND • Inappropriate use of packages.	• No major bugs BUT • Inappropriate use of packages.	• No major bugs AND • Appropriate use of packages AND • The code is integrated with the team's NetBeans project.		1.5	0
CLO2 Produce a software program using appropriate data structures and algorithms (P4, PLO3). Note: Students are required to obtain a pass mark (i.e., at least 40 marks) for CLO2 in order to pass the coursework.					**Subtotal**	**0**
Oral explanation on the implementation & use of ADTs.	• Unclear explanation about ADT implementation AND • Unclear explanation on the use of ADTs	• Unclear explanation about ADT implementation OR • Unclear explanation on the use of ADTs	• Clear explanation about ADT implementation AND • Clear explanation on the use of ADTs		1.5	**0**
CLO3 Explain the implementation and appropriateness of data structures for a specific scenario (A4, PLO5).						
Comments:						

BACS2063 Data Structures and Algorithms - Assignment Rubrics 202305					**Total Mark**	**0**
			Student's name:	**Lim Meng Fu**		
Team name:			**Programme/Group:**	**RDS2G1**		

Assessment Criteria	**Developing 1 - 4 Marks**	**Approaching 5 - 7 Marks**	**Ideal 8 - 10 Marks**	**Marks Awarded**	**Weightage**	**Weighted Marks**
ADT specification	Includes 2 or more of the the items listed under "Approaching".	• Incomplete information OR • Unclear/inaccurate descriptions OR • Includes details of the ADT implementation or entity objects.	• Correct format AND • Complete information AND • Clear & accurate descriptions.		1	0
Collection ADT Implementation	Includes 2 or more of the the items listed under "Approaching".	• Inconsistent with ADT spec. OR • Implementation is not generic OR • Coding is not efficient or elegant.	• Consistent with ADT spec. AND • Correct use of generic types AND • Efficient & elegant coding.		1	0
Collection ADT implementation - originality	Does not demonstrate any originality of implementation & application of new knowledge.	• The ADT has not been covered in the course OR • Includes additional/modified operations that are non-trivial and meaningful.	• The ADT is not covered in the course OR • The implementation approach is different from the sample code.		1.5	0
Use of ADTs - either in entity and/or control class(es)	• Inappropriate selection of ADT AND • Inappropriate use of ADT.	• Appropriate selection of ADT OR • Appropriate use of ADT.	• Appropriate selection of ADT AND • Appropriate use of ADT.		1.5	0
Use of ADTs - creativity & complexity	The functionalities/features in the entity and/or control classes do not demonstrate any use of ADTs.	The functionalities/features in the entity and/or control classes are creatively designed to demonstrate the use of the ADT.	The functionalities/features in the entity and/or control classes are **very** creatively designed to demonstrate the use of the ADT.		2	0

Overall solution	• There is one or more major bugs AND • Inappropriate use of packages.	• No major bugs BUT • Inappropriate use of packages.	• No major bugs AND • Appropriate use of packages AND • The code is integrated with the team's NetBeans project.		1.5	0
CLO2 Produce a software program using appropriate data structures and algorithms (P4, PLO3). Note: Students are required to obtain a pass mark (i.e., at least 40 marks) for CLO2 in order to pass the coursework.					**Subtotal**	**0**
Oral explanation on the implementation & use of ADTs.	• Unclear explanation about ADT implementation AND • Unclear explanation on the use of ADTs	• Unclear explanation about ADT implementation OR • Unclear explanation on the use of ADTs	• Clear explanation about ADT implementation AND • Clear explanation on the use of ADTs		1.5	**0**
CLO3 Explain the implementation and appropriateness of data structures for a specific scenario (A4, PLO5).						
Comments:						