TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BACS2063 Data Structures and Algorithms

ASSIGNMENT SPECIFICATION 202501

Introduction

Abstract Data Types (ADTs) are very important as they serve as programming tools that enable component reuse and encapsulation. This assignment requires students to create, implement and use ***collection ADTs*** in an application. You may do this assignment on an individual basis or in a team of up to 4 members from the same practical class. Team assignments will normally provide a greater degree of appreciation of how various collection ADTs may be used within an application as well as the interdependence of various functionalities and features on the collection ADTs, if all members do their respective parts in a timely and responsible manner.

Learning Outcomes Assessed

CLO	Description	% to Coursework
CLO2	Produce a software program using appropriate data structures and algorithms (P4, PLO3).	85%*
CLO3	Explain the implementation and appropriateness of data structures for a specific scenario (A4, PLO5).	15%

Problem Statement

TAR UMT is developing an **Internship Application Program** to assist students with their internship applications. This program allows students to register, create a profile, and apply for internship positions that match their qualifications. Companies can use the application to post available job positions and search for suitable candidates.

Your task is to develop an **Internship Application Program** that effectively models the **one-to-many relationships** between students, companies, and job postings using **collection ADTs (Abstract Data Types)**. You may make reasonable assumptions as needed. Marks will be awarded based on the creativity and level of competence demonstrated in your implementation.

Modules and Functionalities:

- **Job Management (Employers) Module:**
    - Create a job posting with relevant attributes.
    - Allows updating the details of an existing job posting.
    - Removes a job posting from the system.
    - Allows filtering jobs based on various criteria (e.g., location, company, job type, salary range). This can use different search algorithms based on the criteria.
- **Applicant Management (Job seekers) Module:**
    - Create an applicant with relevant attributes.
    - Allows updating applicant information.
    - Removes an applicant from the system.
    - Filters applications based on criteria (e.g., location, desired job type, skills).
- **Matching Engine Module:**
    - Matching Applicants with the Jobs.
        - Implement a more sophisticated matching algorithm. Consider:
            - Skill matching with proficiency levels
            - Weighting different skills based on importance
            - Handling experience levels and matching them to job requirements.
            - Considering location preferences.
- **Arrange an Interview with Schedule Module:**
    - Implement time scheduling for matched Applicants to attend an interview.
    - Filter successful Applicants for recruitment. Ranks the highest results.
- **Reporting Module** (all members must have)**:**
    - Generates reports on job postings, possibly filtered by criteria.
    - Generates reports on applicants, filtered by criteria.
    - Generates reports on matches (candidates), potentially including a score for each match, indicating the strength of the match.
    - Generates reports on Interview schedule, successful candidates.
- **Search Module:**
    - Implement more advanced search features with criteria, such as:
        - Keyword search across multiple fields (e.g., searching for "software engineer" should find jobs with that in the title, description, or skills).
        - Fuzzy matching (finding close matches to keywords, even if there are typos).
        - Ranking feature based on different criteria and result.

*** Sorting and Searching Algorithms:**

- You are encouraged to use appropriate sorting algorithms (e.g., Selection Sort, Insertion Sort, Quick Sort) when sorting jobs or applicants based on different criteria.
- For searching, encourage the use of more efficient algorithms like binary search if applicable (after sorting).

Scope of Work

The assignment consists of TWO components:

A. Team Component

Each team is required to submit the ADT specification and implementation of **ONE** collection ADT.

Note: Each student in the team should discuss the appropriateness of a collection ADT used and each team is required to select **ONLY ONE** collection ADT to be submitted and assessed.

B. Individual Component

1. Each student in the team is required to develop a prototype of ONE module for the application to demonstrate your appreciation and competency in the use of the team's collection ADT. There should be no duplication of modules among members.
    - The user interface for your application may be console-based, GUI-based or web-based. Note that no marks will be awarded for the user interface. However, the user interface should be easy to understand.
    - Since this is a prototype, the team does not need to develop the complete system/application (No need login, password). However, the subsystems' code for the team should be integrated in the same NetBeans project. For the collection objects that are required for your subsystem but are not included in any team members' scopes, you may populate the collection objects by reading from a file or using a method which adds hard-coded entity values.
    - Only validations that invoke the methods of collection ADTs are required.
2. You are NOT allowed to use any collection interfaces and classes from the Java Collections Framework.

Coding Standards

Teams are required to adhere to the following requirements:

1. Apply the Entity-Control-Boundary (ECB) architectural pattern to structure the classes in your NetBeans project for this assignment. The descriptions and constraints for each type of class are provided below.

    Descriptions of types of classes

    - **Entity classes** represent the data in the system/application. Examples of entity classes are Student, Team, and Course.
    - **Boundary classes** interact with the actors of the system. Examples of boundary classes are MaintainCourseUI, RegisterTeamUI, and GenerateReportUI.
    - **Control classes** implement the business logic for use cases. They orchestrate the execution of commands coming from boundary objects by interacting with entity and boundary objects. Examples of control classes are MaintainCourse, MaintainAssTeam and GenerateReports.
    - **Utility classes** (if any) contain common methods that are used by other classes to perform repetitive general tasks. A utility class contains only static methods and static variables. Thus, its methods and variables are accessed using the class name.

    Constraints for each type of class

    - Actors may only interact with boundary objects.
    - Boundary objects may communicate with actors and control objects only.
    - Control objects may communicate with boundary objects and entity objects, as well as other control objects.
    - Entity objects may only know about other entity objects

    You may refer to the ***ECBDemo*** NetBeans project (download the entire folder) which provides a minimal example of how to implement the ECB pattern in NetBeans.

1. All code must be written using the standard Java naming convention (i.e. Camel Case).

2. Include your name as a comment at the beginning of each class that you authored.
3. If the code for the collection ADT was not written by you or was adapted from a source, acknowledge the source at the beginning of the Java interface/ implementation class.

Assessment Criteria

Refer to the Assignment Rubrics for the assessment criteria and allocation of marks.

Assignment Deliverables and Deadlines

Prepare the team and individual assignment report using the *Google Doc template* provided.

Date/Week	**Deliverable/Activity**
Week 2 by Friday, 11.59pm	**Assignment Team Registration** One member to register your assignment team with your tutor.
Week 10 by Friday , 18/04/2025,11.59pm. Submit to your tutor's Google Classroom.	**Team Assignment Report (GDoc & PDF) & NetBeans project** ● One member to submit to Google Classroom**. ● The NetBeans project should include any data files (e.g. text files/binary files if needed) and a ReadMe.txt file to explain how to run your application.
Week 11,12	**Assignment Demo** ● Each student must demo his/her own work.

**Refer to your tutor for details and follow the naming convention required by your tutor.

Academic Integrity and Plagiarism

There must be originality in your work, i.e. do not copy or refer to other teams. You may only work with your team member(s) to produce the solution of this assignment. You must not share with or refer to any part of the assignment (including the code) of anyone else except your team member(s) and your tutor.

Before submitting your assignment, ensure that you have complied with TAR UMT's plagiarism policy. Any cheating, attempt to cheat, plagiarism, collusion and any other attempts to gain an unfair advantage in assessment will cause the students concerned to be penalised. Students found to be dishonest are liable to disciplinary actions.

Late Submission

Students are required to submit the Late Submission of Coursework Form The together with their assignment.
Refer to TAR UMT's Procedure Relating to Coursework Submission for more details.