**Structures & Algorithms – Coursework**

**Session 2019-20**

You are advised to develop the program one step at a time and keep a copy of the version for each step which should then be included in your submission. In addition to the marks at each step, there are 25 marks for documentation, programming style, testing and an ***individual*** report.

Your ***individual*** report should justify your choice of data structure and testing of the system. You should discuss any difficulties you encountered and any parts of your solution that are not working correctly.

A computer company which has many projects being undertaken at any one time would like a system to hold information on all the current projects as well as details of the employees working on each project. An employee only works on a single project at a time.

The data to be stored for each project is:

**Project Title** e.g. Web site creation

**Expected start/end dates** e.g. 01/01/20 – 31/05/20

The data to be stored for each project team member is:

**Name** e.g. Joe Bloggs

**Employee Number** e.g. 999999

**Division** e.g. Web Development

The features required by the Company include:

* Add details of a new project
* Add details of a new member of a specified project
* Display the details for a specific project
* Display all the details for all the projects
* Remove a particular employee from a specified project
* Remove a project from the system

**Step 1: Functionality is worth 25 marks**

* create a **Company** class with suitable attributes to represent each project as well as a link to each project.
* Using an appropriate dynamic data structure, write a **CompanyTest** class that offers a simple menu to **add** a new project, **find** and **display** a specific project, **remove** a specific project and to **display** the details of all the projects currently being carried out by the company in alphabetical order of project title;
* Use appropriate exception handling.

**Step 2: Functionality is worth 15 marks**

* Create a **TeamMember** class with suitable attributes;
* Using an appropriate dynamic data structure, different from the one used above, write a **TeamMemberTest** class that offers a simple menu to **add** a new team member, **find** and **display** a specific team member, **remove** a specific team member and to **display** the details of all the team members;

**Step 3: Functionality is worth 35 marks**

* Combine the two structures above modifying the necessary methods so that the resulting data structure is a **binary search tree**; You will need:
  + An **addProject** method that adds projects to the structure;
  + an **addTeamMember** method that interacts with the user to select a **project** to add the project team member details to;
  + a **removeTeamMember** method to remove a specific team member from the system if they are no longer a member of the team;
  + a **removeProject** method to remove a project from the system;
  + a **displayTeamMembers** method which interacts with the user to select the project to display and which then displays all the team members for that project;
  + a **displayAll** method to display the information on all data for the projects including all the team member info, in alphabetical order of project;
* provide a method to display the number of members in each project team.
* add JUnit classes that test your solution.

**Marking Scheme**

Source code must compile to qualify for marks (Except items under “Individual Report”)

|  |  |  |
| --- | --- | --- |
|  |  | **Max** |
| **Step 1** |  |  |
|  | add Projects to chosen dynamic structure | 5 |
|  | checks if project already exists & displays suitable message | 2 |
|  | find a particular project & display info | 3 |
|  | suitable message if project not found | 1 |
|  | remove project from structure | 3 |
|  | display all projects in alphabetical order | 3 |
|  | suitable indentation / formatting | 2 |
|  | suitable message if no projects in the system | 1 |
|  | Error handling for user input in all menu options | 1 |
|  | exception handling | 2 |
|  | Confirmation quit message when choose quit option | 1 |
|  | invalid message if invalid menu option | 1 |
|  |  |  |
| **Step 2** |  |  |
|  | add project team members to chosen dynamic structure | 3 |
|  | checks if team member already exists | 2 |
|  | suitable message if team member already in the system | 1 |
|  | find a particular team member & display info | 2 |
|  | suitable message if team member not found | 1 |
|  | remove team member from structure | 3 |
|  | display all team members | 2 |
|  | suitable message if no team members in the system | 1 |
|  |  |  |
| **Step 3** |  |  |
|  | combine the two structures so that the resulting data structure is a binary search tree | 3 |
|  | add projects to the structure | 2 |
|  | display project added message | 1 |
|  | add project team members | 3 |
|  | when select a Project |  |
|  | detects no match for Project | 1 |
|  | suitable message displayed | 1 |
|  | only ask for team member information if project ok | 1 |
|  | displays team member added message | 1 |
|  | deletes team member | 3 |
|  | displays team member removed message | 1 |
|  | deletes a project | 2 |
|  | displays project removed message | 1 |
|  | display |  |
|  | displays team member details for a specific Project | 2 |
|  | display all the project and team member details in alphabetical order of project | 3 |
|  | layout as required | 2 |
|  | provide a method to display the number of team members assigned to each project | 3 |
|  | Add JUnit classes to that test the solution | 5 |
|  |  |  |
| **Style & report** |  |  |
|  | code layout/indentation | 2 |
|  | meaningful names | 2 |
|  | appropriate use of public/private | 1 |
|  | appropriate use of static identifier | 1 |
|  | appropriate override of Object methods | 1 |
|  | use of this to identify object components | 1 |
|  | adherence to structured programming | 1 |
|  | Source code in GitHub repository | 2 |
|  | Evidence of use of the GitHub repository | 2 |
|  | Solution design follows Model-View-Controller Pattern. | 4 |
|  | Individual report including testing & reflective assessment of work | 8 |