**Programming Assignment Planning Document**

**Data Structure**: Linear / Doubly ADT : List / Queue

|  |  |  |  |
| --- | --- | --- | --- |
| Member Name: | **<Name 1>** | **<Name 2>** | **<Name 3>** |
| Module:  **Console or UI** | *Emergency Management*  *(including contact lists, equipment, etc )* | *Emergency Alert*  *(including Registration of Users)* | *Workload Assignment and Task Coordination*  *(including Registration of Guard)* |
| Main / Driver program:  Number of linked object(s) created with <generic type> |  | QueueInterface<Request> AlertList = new DLinkedQueue<>();  QueueInterface<Request> servedRList = new DLinkedQueue<>(); | QueueInterface<????> TaskAssignedList = new DLinkedQueue<>(); |
| Entity Classes |  | Request, User | Guard, |
| How should the various entity classes be processed |  | Compare a new entry’s level with the existing node in list. |  |
| ADT(s) implementation required (operations) |  | addNewRequest (int level, T entry)  updateRequest(int level, T entry)  removeRequest(T entry)  displayEntry(QueueInterface requestList) |  |
| Sorting / Ranking | Usage: High  Available : | 3 condition levels:  1 low, 2 average , 3 high | Experience  Available / not available |
| Useful Reports |  | Latest served request , newly added request in list.  Summarize all requests added and waiting in queue. |  |

**Each member required ONE interface, implementation and driver program.**

**Report Hardcopy checklist:**

1. Cover page (*use the template provided*).
2. Declaration page(Sign)
3. Contents page
4. Part 1: ADT Specifications (appendix 4.1 format), Fraction Prac2Q1
5. Part 2: ADT Implementations
6. Part 3: Application Program
7. Softcopy: Under Your Name 🡪 3 folders🡪 1.adt(interface, implementation class) 2.driver program, 3.entity classes.

include the NetBeans project, or text files : - DVD/CD-ROM to be attached end of the report.

Part 1: ADT Specifications

ADT: Alert

Part 2: ADT Implementations

2.1 Data Structure:

Linked implementation using  **doubly linked queue** with **head** and **tail references**

Reason: **traverse both direction to compare the request ranking in existing linked chain, …**

2.2 Efficiency of the data structure used to represent the ADTs to implement its various operations:

1. Boolean addNewEntry (int level T new Entry)

NodetempNode;

Int Count;

ADT: Emergency Alert (member: \_\_\_\_\_\_\_\_\_\_)

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Fields** | **Operations** | **Description** | **Efficiency** |
| **tempNode**  **count** | Boolean addNewEntry (int level T new Entry)  NodetempNode;  Int Count; | How operation performed.  Declare 3 temporary points: nodeBefore, nodeAfter, tempNode  3 conditions:   1. Add new request is highest ranking   For……  2. | O(n) : when low request, ..  O(1) :when |
| startNode, lastNode,  rankRequest | Boolean arrangeNode (T new Entry, int Rank) |  |  |
|  |  |  |  |

ADT: Workload and Task Coordination (member: \_\_\_\_\_\_\_\_\_\_)

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Fields** | **Operations** | **Description** | **Efficiency** |
| startNode, lastNode,  TaskRequest | Boolean addTaskEntry (int level T new Entry) | How operation performed.  Declare 3 temporary points: nodeBefore, nodeAfter, tempNode   1. conditions: 2. Add new task request based on guard’s status…….   For | O(n) : when low request, ..  O(1) :when |
| startNode, lastNode,  rankRequest | Boolean arrangeNode (T new Entry, int Rank) |  |  |
|  |  |  |  |

**Part 3: Application Program**

Main Program:

ScreenShots for functionalities with label in figure 1, 1.1, 1.2

Descriptions of the functionalities provided by your application program with sample screenshots.

**Appendices: Source Code Listing** 🡪 print out in doubled-side

1. Interface class
2. Implementation class – line number code listing
3. Main or driver program – console : QueueInterface<Request> alertList = new DLinkedQueue<>();

GUI : Close off coding for UI component (refer print option in netbean)

File 🡪 Print 🡪 Print options

