Heterogeneous structures: Circular Queue and Binary Search Tree

Purpose:
- to develop generic data structures able to store heterogeneous objects offering various services
- to produce clearly written documentation reflecting understanding of OO concepts and data structures

I. SPECIFICATION

1. *AnyClass* class

Both circular queue and binary search tree data structures have to store class *AnyClass* as an abstract class stored in the package *dataobjects*. The class has a single data field *int seqNo* with the corresponding setter and getter methods. Apart from the constructor the class is to have three polymorphic (virtual) methods:

- polymorphic method *String getData()* to return data of the object in form of the string. In case of the *AnyClass* it returns value of the integer *seqNo*
- polymorphic method *String getKey()* returning a *key* value in form of the *String* type. The *key* will be used for searching of the object. In case of the *AnyClass* it returns value of the ***seqNo*** as a *String*
- polymorphic method *void edit ()* to edit object data by entering the new values from the keyboard. In case of the *AnyClass* it is "empty" because the data field *seqNo* is not supposed to be edited.

Any actual object to be stored must extend from the *AnyClass* class.

2. Classes inheriting from the abstract *AnyClass* class

Both below specified dynamic data structures (see items 4. and 5. below) will store mixture of objects of the classes *Employee* and *PartTimer* developed during practical sessions.
The key used for operations of searching and editing will be ***surname***.
Editable value will only be value of ***pay***.
Both classes have to override the polymorphic methods from the class *AnyClass*.

3. Linear and Binary Nodes

Linear Node – class *Node*

Used by the circular queue. Its data part is an object of the class *AnyClass*. Apart from the constructor it has no methods. The class *Node* is to be placed in the package *linearnodes*.

Binary Node – class *BNode*

Used by the Binary Search Tree. Its data part is an object of the class *AnyClass*. Apart from the constructor it has no methods. The class *BNode* is to be placed in the package *binarynodes*.

4. Heterogeneous Circular Queue – class *CQueue*

The *CQueue* class is to reside in the package *linearstructures* and has to import both the *dataobjects* and *linearnodes* packages. The queue is a circular queue offering the following operations:

a) construction of a *CQueue* object with the size of 20 nodes
b) placing new object to the queue returning success only if the queue was not full:
method *boolean put (AnyClass newObj)*
c) serving the front object by removing it from the queue and returning object of *AnyClass or null* if the queue was empty: method *AnyClass serve()*
d) listing all objects currently on the queue while showing data of each object: method *void listAll()*
e) editing of a particular object identified by the *key*: method *AnyClass editObject (String key)*
The method traverses the queue to find the object to be edited. If found then returns a reference to the object just edited otherwise null.
f) The method *void changePayOfAll (int percent)* which traverses the queue and changes the pay of **all** objects stored by a percentage value *percent* (i.e. without use of the edit() method)

5. Binary Search Tree (BST) – class *BinSearchTree*

Is to reside in the package *non_linearstructures* and has to import both the *dataobjects* and *binarynodes* packages. The Binary Search Tree is a tree structure offering the following operations:

a) adding new object to the tree keeping it sorted in order of a the *key (*with its value returned by the *getKey()* method) : method *void insert(AnyClass newObj)*
b) searching for the object by the *key* value returning reference to the object if found otherwise null: method *AnyClass search (String key)*
c) listing of all objects on the tree in ascending order of the *key* while showing data of each object: method *void listInOrder()*
d) population of a BST from the queue. The method traverses the queue (implemented in item 4. above) and populates the BST by binary nodes containing objects stored in the queue nodes by calling the queue's *serve()* method. This process empties the current contents of the queue. Note that this method actually sorts the objects in ascending order of the key values.
Method: *void populateFromQueue (CQueue que)*, where *CQueue* is a class name of the circular queue

II. DELIVERABLES.

1) Working applications as per the specification. The actual objects to be stored are of the classes *Employee* and *PartTimer* which **extends** the *Employee*.
The main program class should construct both dynamic structures and then demonstrate all possible operations the structures can perform.
Suggested menu:
- Construct empty queue and binary tree
- Populate the queue by mixture of objects of Employee and *PartTimer* instantiated to values entered from the keyboard. Use static method explained in lecture and developed during practical sessions.
- List all objects currently placed on the queue
- Update payment of a person with matching *surname* and *idNo* stored on the queue, display its data and list the queue again
- Update payment of all persons stored on the queue by 10% and list the queue again
- Populate the BST by all queue objects.
- Search for a particular object with matching surname
- List all objects on the BST in ascending order of their surnames

2) Documentation
- To be submitted on the **VLE**
- Screen shots showing working of major methods
- User manual specifying **services** offered by the two dynamic structures to the end user: **packages to be imported**, **procedure to be followed** when any **other** object (rather than *Employee* and *PartTimer*) would be considered to be stored on both the Circular queue and Binary Search Tree. You can assume object of class **CEO** (Chief Executive Director) who, apart of the fixed salary, is paid also an additional bonus of 30%.
- Source code listing

III. FORMAL ASPECTS.

1. **Team work**
Team will consist of two students, each selecting a single, particular, dynamic structure out of the two structures required. The marks will be assigned separately for each student based on their work and demonstration.
2. **Documentation**
1) Front page: Unit code (CIS 1100), Title of the assignment, Names of authors and names of the dynamic structures solved by them, Year (2020/2021). Failure to follow this requirement will be penalized by lower mark
2) as per above II.2
3. **Date of Submission**
The submission deadline is NOT later than the Examination of CIS1100. There is NO possibility of extension for any reason. Project submitted after the deadline will get zero marks!
4. **Demonstration of Project:**
Working program. Duration 10 min. Dates will be announced.