**COEN 359 Design Patterns Summer 2015**

**Assignment 3 (50 pts) Due: 3rd Aug (10 PM)**

**Question 1 (40 pts)**

Given below is a description of computer lab management in East-West University.

Individual workstations and individual printers are considered **LabResources**. A number of **workstation**s and **printer**s can be grouped into **ComputerCluster**s, where each ComputerCluster is also considered a **LabResource**.

**Note**: *In the description below, a LabResource is also referred to as a Resource.*

The following rules apply to LabResource classes.

1. Individual **Workstations** and **Printers** have a resourceId (String), modelyear (int), status (active or inactive), supervisor (String) and cost of maintenance per year (double).
2. A **ComputerCluster** may contain individual workstations, printers and/or other ComputerCluster instances. A ComputeCluster has a resourceId (String), supervisor (String) and cost of maintenance per year (double). Cost of maintenance per year is the total of cost of maintenance of the individual resources in the cluster.
3. Each ComputerCluster will have a ClusterId (string) and a single Supervisor for the entire cluster. When a workstation or printer is added to a cluster, the supervisor of each workstation in that cluster should be set to the cluster supervisor.

**You must provide methods to do the following:**

1. Write all the needed **constructors**.
2. **Adding a resource to a cluster**: An **add** () to add a Workstation (or a printer) to a Cluster.
3. **Removing a resource from a cluster**: To remove resource from a cluster, you must set the status of that resource to **inactive**. It is not required to physically remove them from the cluster.
4. **costOfmaintenance**: Compute and show the cost of maintenance per year.
5. **oldModel**: For a WorkStation and Printer, check if the modelYear is older than the year passed as a parameter. For example, oldModel(2004) should return a boolean value depending on the modelYear of the resource.
6. **showOlderModels**: For a ComputerCluster, show the list of resources in the cluster, that are older than the year passed in as a parameter. For example, showOlderModels(2004) should show the resourceIds of the resources in the cluster that have modelYear older than 2004.
7. **showResourceDetails**: For individual Resource types - A method to display the resourceId, supervisor, modelYear and status. For a ComputerCluster, display the resourceId of the cluster, supervisor and information of each individual resource.
8. **setInactive**  - For a single resource, sets the status to inactive.
9. **setInactive(resourceId)** : For a ComputerCluster, searches for the resourceId of the resources in the cluster and sets its status to inactive.

**Implement** the above scenario using the **Composite** design pattern.

1. Draw the UML class diagram for the classes in your implementation.
2. Implement the classes using the description given above.
3. Discuss if you have chosen type-safety or transparency in implementing the child management operations of adding a resource to a cluster.
4. You must create the following:
   * Two clusters (referred to as A and B in this description), where clusterA consists of two individual workstations and one printer; clusterB consists of clusterA and one other printer.
5. Show a testcase (with output) for all the operations xa to i on clusterB. Capture a screen shot of the output and include it in the submission.

**Question 2 (10 pts)**

**See the description of the classes and interfaces given below.**

The clients want to use the JobQueue and its public methods. To implement the JobQueue, there are two existing classes ArrayQ and LinkedQ, both implementing the interface Queue. The methods in JobQueue are essentially the same methods (but with different names) in a Queue interface.

Class **JobQueue** maintains a queue of JobIds (strings) and has the following methods:

**Methods**

* addJobIdToQ (String jobId)
* removeJobIdFromQ()
* noMoreJobsOnQ()
* size()

***interface* Queue**

* addToQ (String id)
* removeFromQ()
* isEmpty()
* size()

A Queue interface can be an Array Implementation (class ArrayQ) or a LinkedList Implementation (class LinkedQ).

1. What design pattern(s) would you use to implement a JobQueue providing a flexibility to use either an array implementation or a LinkedList implementation. Explain and show the UML class diagram. The UML class diagram should be saved as an image file (a jpeg or a png file) and included in the discussion. *Please do not include the UML tool generated files.*

Implementation is not required.