Task : Computing cluster

Introduction

In this task you will create a program to keep track of all the components in a computing cluster. That way, a simulation that would take a month to run on a regular machine can be run on the computing cluster in a few hours instead.

computing cluster `s constituents

A computing cluster consists of one or more racks (a cabinet with rails) how many nodes can be mounted over each other. A node is an independent machine with a motherboard with single or multiple processors and memory, as well as some other things. In this task, we will only look at the number of processors (max 2) and the size of the memory. You can therefore assume that a node has one or two processors and an integer number of GB with memory.

Program Design

The program should be designed with three classes representing nodes, racks and computing cluster An object of the class computing cluster should be able to refer to one or more rack-objects, where each rack object again refers to one or more node objects. A few more requirements and tips for designing the individual classes can be found in the assignment.

Under the Tasks and Delivery section below, you will find the description of what your program should be able to do.

Important that you comment on your chosen methods/choices.

Class Node

The class should be able to initiate new objects with the desired memory size and processor number, and for the rest of provide services (methods) that needs in other parts of the program.

Class Rack

The Rack class should save the Node objects belonging to a rack in a list. We will be able to add nodes to the racket if there are fewer than the maximum number of nodes there from before. For simplicity, we assume that each rack in the computing cluster has space for as many nodes. Other instances variables and methods as needed.

Class Ccomputing cluster

The class computing cluster will keep track of a list of racks, and must offer a method that accepts a node object and places it in a rack with free space. If all the racks are full, a new Rack object will be added to the list and the node will be placed in the new rack.

Tip. You may want to include the number of nodes per rack in the constructor of computing cluster

Tasks and delivery

You must deliver the program in Devilry with one file per class and main program, as well as drawing from task a).

a) Data Structure Drawing

Note. This assignment should be answered - but the drawing will not be able to be deduced in the assessment of the assignment, and you choose how you think it is useful to illustrate. Delivered as a separate file in pdf or similar format.
Suggestion. Variable is drawn as named boxes with the value inside. Objects are drawn as squares with instances variables inside. Class names may be overwritten over the object with ":" in front.
References are drawn as arrows from the variable they are in to the object they refer to. Draw the data structure as it would look after we have inserted the following nodes into a new object of computing cluster. Leave max number of nodes per rack to be 2.

* Node 1: 16 GB memory, 1 processor
* Node 2: 16 GB memory, 1 processor
* Node 3: 128 GB of memory, 2 processors

b) Number of processors and memory requirements

Create a method of antProsessorer(self) in the Computing cluster that returns the total number of processors in the computing cluster.
Some programs require a lot of memory, typically a given number of GB with memory of each node we use. We are therefore interested in knowing how many nodes have enough memory to use them. Create a method noderMedNokMinne(self, paakrevdMinne) in computing cluster which returns the number of nodes with least required Memory GB memory.

c) Main Program

Create a main program to test that the classes are working properly. Create a computing cluster , abel, and leave space for 12 nodes in each rack. Enter 650 nodes with 64GB of memory and one processor each. Also add 16 nodes with 1024 GB of memory and two processors.
Check how many nodes have at least 32GB, 64GB and 128GB of memory. Find the total number of processors, and check how many racks are used. Print the answers in the terminal. The printout can, for example, look like this:

```
Node with at least 32 GB: 666
Nodes with at least 64 GB 666
Nodes with at least 128 GB: 16
Number of processors: 682
Number of racks: 56
```