Virtual Machine management system – Simple ESX simulator.

Source code must be submitted in perforce server. Your task will be assessed in following categories: Correctness (validation of input parameters, proper exception handling), Coding Style (java docs, formatting, readability and naming conventions), Design (Interfaces, separations of concerns, proper model and patterns), Algorithms, Unit tests.

The main goal is to implement a simple **ESX** simulator. **ESX** is a system that can manage **Virtual Machines**. Manage means that our ESX system should be able to Create, Delete, Edit and List (display) Virtual Machines. A **Virtual Machine or VM** is an entity similar to physical computer. For the purpose of this task we will use simplified VM, which has only unique ID, name, memory and a set of devices.

Virtual Machine detailed description:

* name - a character string. There may be more than one VM with a given name. Name is a string, which contains alphanumeric characters and space.
* id - a character string. The id is unique for each VM and is used to identify it. One should never have more than one VM with a given id. VM id contains alphanumeric characters only.
* memory - the size of the RAM of the VM. This will be measured in bytes (1KB = 1024 bytes) and should be stored in bytes.
* devices - Collection of devices. Assume that a VM can have more than 1000 devices.

A device can be one of the following two things:

* **Video Card**
* **Network card**

A video card has the following properties:

* id - a character string that is unique through all devices for the VM. May only contain alphanumeric characters.
* Video RAM - measured in KB and its maximum value is 4 GB
* Number Of Displays – number

A network card has the following properties:

* id - a character string that is unique through all devices for the VM. May only contain alphanumeric characters.
* MAC address

The ESX system sequentially reads **operations** (commands) from the command line and executes them. Operations are one of the following:

* create-vm <id> '<name>' <memory> - creates a new VM with the specified id, name and memory and with no devices. Memory parameter is in bytes.
* delete-vm <id> - deletes the VM with the given id.
* edit-vm <id> '<new\_name>' <new\_memory> - edit the VM with the given id. Even when the name or the memory of the VM remains unchanged, they are provided.
* add-dev <vm\_id> <dev\_type> <device\_spec> - adds a new device to the virtual machine with id - **vm\_id**. Each device has a unique id among all devices in this VM. Device id is specified in device\_spec.

**dev\_type** will be VIDEO\_CARD or NETWORK\_CARD for video card or network card respectively. **device\_spec** will be a device specification for a device of the type specified (see below the device specification for video card and network card).

* delete-dev <vm\_id> <dev\_id> - deletes the device with the specified 'dev\_id' from the virtual machine with id 'vm\_id'.
* print-vms - prints a human-friendly summary of all the current VMs on the standard output for the program.

A <device spec> for a video card will look like this:

* <dev\_id> <videoRam> <numberOfDisplays> - the values are separated by at least one whitespace character. <videoRam> and <numberOfDisplays> are both non-negative integers. <videoRam> is measured in KB and its maximum value is 4 GB. Maximum allowed value of <numberOfDisplays> is 2.

A <device spec> for a network card will look like this:

* <dev\_id> <mac\_address> - the format of <mac\_address> is “HH-HH-HH-HH-HH-HH”, where “H” is a hexadecimal digit

Command Example:

> create-vm vmid1 ‘My UbutnuVm’ 536870912

> delete vmid5

VM with id:vmid5 not found!

>add-dev vmid1 VIDEO\_CARD vd1 1024 1

>printvms

Id Name Memory

vmid1 My UbuntuVm 512MB