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
```

Task 1:

a) Extend Network card with additional property –IP. Do the necessary changes to a network device spec so that we can pass the IP address as additional argument when we add new network device. You can assume that the IP is only of type IPv4.
b) Your code should validate the IP and MAC addresses of network card.
c) Modify your program so that it can read commands from a file.
d) Command help. You should provide a help command which list all commands and short description for each command.

Task 2:

a) Your ESX simulator should be able to save its state. When you stop the ESX all the VMs and their configuration should be persisted on the file system. When you start the ESX simulator, it should read the persisted data from the file system and should restore the VMs. Each VM should be stored in a separate file and filename must be the same as vm_id. All VM files should be stored in one directory.

b) Add the following 2 commands to the supported set of operations: save-vm and read-vm. Each of these commands takes a single argument – the id of the VM to be stored or read from the file system.

- save-vm <id> - Saves the VM with the specified id on the file system.
- read-vm <id> - Reads the configuration of the VM with the specified id from the file system and replaces the current settings.

c) Extend the ESX simulator to allow reading commands from multiple files. There should be a separate thread for each input file.

Task 3:

a) Add stop command. This command should stop your program and should save the state of ESX simulator. See Task 2a.

b) Your VM should be extended to support the following new configuration – number of CPUs, which should be between 1 and 8. You should update create-vm and edit-vm commands to have one additional argument - the number of CPUs. All the existing VMs should be modified to have one CPU.

c) Extend the list of supported VM devices with new device type - **HardDisk Controller.** Add-dev command should be updated to handle the new device. There are 2 types of HardDisc Controller – **IDE**and **SCSI.**

IDE controller supports maximum 4 disks while SCSI controller supports 16 Hard Disks. A VM can have maximum 1 IDE controller and maximum 4 SCSI controllers.

A HardDisk Controller has the following properties:

- id - a character string that is unique among all devices for the VM. May only contain alphanumeric characters.
- type – one of the two values: IDE or SCSI

A device spec for a HardDisk Controller will look like this:

- <dev_id> <controller_type> - Controller_type is one of the two Strings IDE or SCSI.

Also add another device type - **Hard Disk**(also extend add-dev to support this device type). A hard disk has the following properties:

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

* size – hard disk size in bytes. Maximum size of a hard disk is 100 TB(100 * 1024 * 1024 * 1024 * 1024 bytes)

* hard disk controller id - id of the controller that the hard disk is attached to. Should be a valid id of a hard disk controller that already exists.

A device spec for a HardDisk will look like this:

- <dev_id> <size> <controller_id> -

Deleting a HardDisk Controller, deletes all the hard disks attached to it.