DESCRIPTION

You're running your infrastructure in the cloud. It consists of multiple services and applications that together provide valuable service(s) to your customers. To manage your infrastructure you need to be able to create and boot new instances to deploy new software and upgrade existing systems. You also need to be able to monitor and query the state of your instances and shut down those that are no longer needed to minimize your costs. Since your infrastructure is highly available and fault tolerant you run multiple instances of each service/application and load balance the access. In this scenario it's often more convenient to manage hosts in groups. Finally, host groups of a certain type (specific application or service) often have dependencies on host groups of other types. For example, your customer portal application depends on your authentication service, which authenticates customers logging in. Your task is to implement necessary classes/data structures that model such a system and a simple console based shell demonstrating it in action.

REQUIREMENTS

At a very minimum you need to implement the following classes/data structures:

- Host type -- represents the meta-data describing an instance of a particular type. A host type should have the following attributes:
    - ID -- randomly generated unique ID; read only, assigned at creation time
    - Description -- arbitrary string describing host type; can be changed at any time
    - Type of service or application attached to the host -- e.g., "customer-portal"
    - Size of the host -- can be SMALL, MEDIUM, or LARGE

  It's assumed that your infrastructure has pre-defined set of host types, which is stored in the database and represents your services and applications running in the cloud. You can use a file to store this information. Here is an example of such file in JSON format:

```
{ "host_types":
    [ { "type": "customer-portal,
        "id": "HT12345aec7890",
        "description": "Customer self service web site",
        "size": "large"
      },
      { "type": "auth-service",
        "id": "HT67890edf1234",
        "description": "Account authentication service",
        "size": "medium"
      } ]
}
```

Feel free to change JSON format to your liking or choose completely different format. However, regardless of the format you should implement methods/functions that allow managing host types, e.g. reading the information from disk, adding, deleting, and updating host types, as well as storing new data to the disk.

- Host -- represents your instance in the cloud; there should be methods/functions defined to create, boot, activate (bring into the load balancer), deactivate, shutdown, and delete a host. The methods should perform appropriate state transitions and print them out on a console. You should also be able to query host's current state. A host should have the following attributes:
    - ID -- randomly generated unique ID; read only, assigned at creation time
    - Description -- arbitrary string describing the host; can be changed at any time
    - Type of service or application attached to the host -- e.g., "customer-portal"
    - Size of the host -- can be SMALL, MEDIUM, or LARGE
    - State of the host; cannot be changed directly but only via accompanying methods/functions. At a very minimum the following states should exist:
        - CREATED -- when new host is instantiated
        - BOOTING -- when a command is issued to boot the host
        - RUNNING -- when host is booted and application is running
        - SHUTTING DOWN -- when a command is issued to shutdown the host
        - SHUTDOWN -- when the host finished shutting down
    - Flag indicating whether a host is in the load balancer (active) or not
    - Host group ID -- the group host belongs to; can be NULL if host is standalone

- Host group type -- similarly to the host type, represents meta-data describing a group of hosts of the particular type. A host group type should have the following attributes:
    - ID -- randomly generated unique ID; read only, assigned at creation time
    - Description -- arbitrary string describing the type; can be changed at any time
    - Type of service or application attached to the group -- e.g., "auth-service"
    - Dependencies -- host group types (IDs) this group type depends on
    - Size of the host -- can be SMALL, MEDIUM, or LARGE
    - Number of hosts in a group
    - A flag indicating whether hosts in the group should be brought into the load balancer (activated) immediately once they are booted or not

  Similarly to host types, host group types are stored on the disk and read upon your management service initialization. You should define methods/functions that allow managing host group types, e.g. reading the information from disk, adding, deleting, and updating host group types, as well as storing new data to the disk.

- Host group -- represents a group of instances in the cloud that share the same type; there should be methods/functions defined to create, boot, shutdown, and delete host groups. The methods should perform appropriate state transitions and print them out

on a console. You should also be able to query its current state. A host group should have the following attributes:

- ID -- randomly generated unique ID; read only, assigned at creation time
- Description -- arbitrary string describing host group; can be changed at any time
- Type of service or application attached to the host -- e.g., “customer-portal”
- State of the host group; cannot be changed directly but only via accompanying methods/functions. At a very minimum the following states should exist:
  - CREATED -- when new host is instantiated
  - BOOTING -- when a command is issued to boot the host
  - RUNNING -- when host is booted and application is running
  - SHUTTING DOWN -- when a command is issued to shutdown the host
  - SHUTDOWN -- when the host finished shutting down
- Hosts -- references to the hosts that comprise the group

Finally, you need to implement a simple command line shell that can demonstrate all this in action. Your shell should have the following commands/features:

- Help -- provide information about available commands and usage
- Loading of configuration files -- load information about available host types and group types from the file system; this can be done with command line options when the shell is started or as separate commands later on
- Host management -- host creation, booting, activation/deactivation, and shutting down
- Host group management -- host group creation, booting, and shutting down
- Querying -- get the status of individual hosts and host groups

CONDITIONS

- You cannot take/copy/google an existing implementation of the classes/structures above

- Other than that you can use any 3rd party libraries to simplify your job; here is the list of suggestions:
  - C/C++
    - JSON-C
    - Mini-shell
    - Generic state machine
  - Java
    - Jackson JSON processor
    - Java FSM
    - Java Command Line Shell
  - Ruby
    -

- You can use man pages, user guides, and google information about specific implementation details, such as proper syntax and references for standard libraries:
    - C/C++
        - C++ reference
        - Man pages
    - Java
        - Java API documentation
    - Ruby
        - Ruby core API
        - Ruby standard library documentation

- Whatever code you produce, at the end of this exercise it should compile and run even if functionality is not complete