

Spring 2015

**CMPE 283**

Project 1

**Disaster Recovery Manager**

Submitted to:

**Prof. Simon Shim**

(Dept. of Computer Engineering)

 Submitted by:

Swetha Rathna Kempahanumaiah (009343477)

Table of Contents

[INTRODUCTION 2](#_Toc416502531)

[Goals: 2](#_Toc416502532)

[BACKGROUND 3](#_Toc416502533)

[REQUIREMENTS 3](#_Toc416502534)

[Functional Requirements: 3](#_Toc416502535)

[Non-Functional Requirements: 3](#_Toc416502536)

[DESIGN 3](#_Toc416502537)

[Architecture Class diagram: 3](#_Toc416502538)

[Class Description: 4](#_Toc416502539)

[VmWare Components Used: 5](#_Toc416502540)

[Key Workflows: 5](#_Toc416502541)

[IMPLEMENTATION 6](#_Toc416502542)

[Environment 6](#_Toc416502543)

[Tools: 6](#_Toc416502544)

[Screenshots: 6](#_Toc416502545)

[QUESTIONS 14](#_Toc416502546)

[DISCUSSION POINTS 15](#_Toc416502547)

[CONCLUSION 16](#_Toc416502548)

[REFERENCES 16](#_Toc416502549)

# INTRODUCTION

With the increases in the number of virtual machines running in a system, the maintenance of the same becomes very difficult. In such scenarios, there is always a risk of failures of any virtual machine which can lead to unavailability to the end users. The system administrator will have to automate the scenarios of monitoring the system for any failures and seamlessly recover or backup the data to another system. As part of this project, an availability manager for monitoring the virtual machines and a disaster recovery system is designed that can be used for administration purpose.

## Goals:

The goals of this project are:

* Support for backup cache which updates every 10 minutes.
* Periodic monitoring of the virtual machines for any failures.
* Recovery system for the failed virtual machines and vHosts.
* Support of alarm notifications on vHosts manual power off.
* Understanding of VMWare VI Java APIs.

# BACKGROUND

As part of course lab environment, there are many vCenter created. The vCenter has ESXi operating system installed on it and is used for creating datacenter, adding hosts and creating virtual machines. These vCenter’s have multiple virtual machines created under each vHost. As part of this project, we need to get a deep understanding of the VMWare architecture and the hierarchy structure of virtual machines, vHosts and vCenter. Using the VMWare VI Java Apis, an application that monitors the virtual machines and a recovery system has to be implemented.

# REQUIREMENTS

## Functional Requirements:

* Snapshot of the VM and vHost should be taken every 10 minutes.
* A VM is considered alive if its responds to ping, else dead if its stop responding for a configurable amount of time
* If the ping to VM fails, then the corresponding VHost is checked. If the VHost is alive, then the VM is provisioned using the snapshot of that VM.
* If the VHost is dead as well, then the VHost is provisioned using the VHost snapshot.
* A new vHost is created if there exists only one VHost and that VHost is not alive.
* An alarm should be created and triggered when the VM is manually powered off. In this scenario, the solution should prevent failover from occurring.

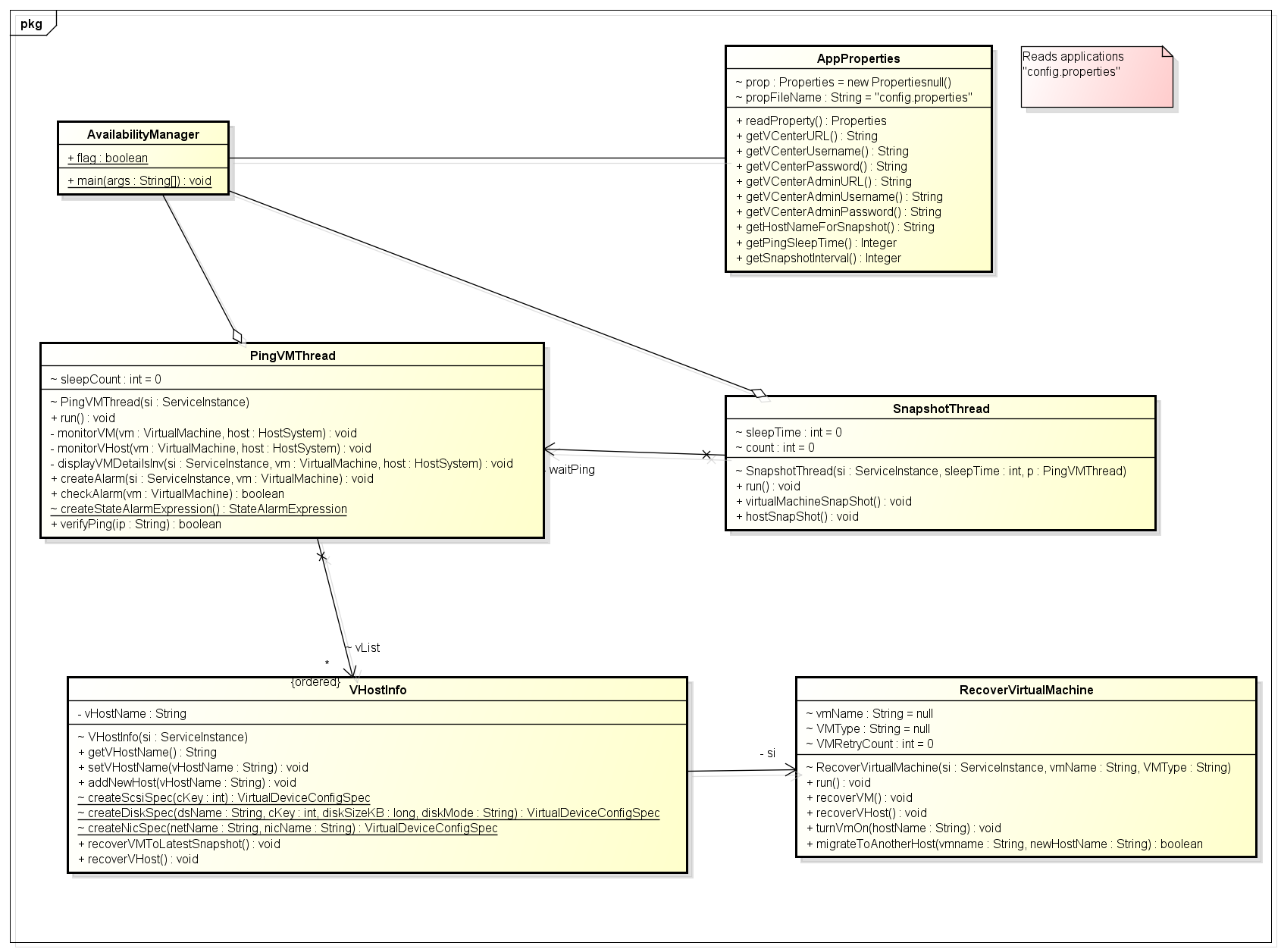
## Non-Functional Requirements:

* Snapshot should be taken only when the virtual machine are on.
* Ping to virtual machine should be after every periodic interval.
* Synchronization across multiple threads should be taken care.

# DESIGN

## Architecture Class diagram:

Class diagram of the solution architecture is shown below:



## Class Description:

**AvailabilityManager class:**

This is the main program which starts all the threads that are defined. It reads the VCenter URL from the user input and passes that url to all the thread instantiate methods. There are 3 threads that are implemented.

* The first thread is the PingVM thread, used for monitoring the VM/VHost alive status every periodic interval which is configurable.
* The second thread is the Snapshot thread, used for creating snapshots of the alive VMs and vHost every periodic interval of 10 minutes.
* The thread thread is the RecoveryVirtualMachine thread, used for recovering the failed VM or VHost from the latest snapshot.

**SnapshotThread class:**

This is a java thread class which performs the operation of maintaining snapshots of all the live virtual machines and the vHosts. This runs in a loop taking snapshot every 10 mins and the previous snapshot is deleted periodically. This is done using the VI api: createSnapshot\_Task and removeAllSnapshots\_Task method.

**PingVMThread class:**

This is a very important thread class which performs the task of retrieving all the VHosts and virtual machine under each vHost and checking their status. This thread performs ping operation on each of the VMs. Depending on the VM status, a recovery methods are invoked for recovering the VM or its vHost. This also maintains a list of active vHosts and its corresponding VMs using the VHostInfo object type. For each of the VM, an alarm is created which is used to verify if the VM is manually powered off.

**RecoverVirtualMachine class:**

This thread class performs the VM/VHost recovery operation using the snapshot revert operation. The snapshots created by the snapshot thread is used to revert the failed VM or VHost and made alive. This uses the VI apis: revertToCurrentSnapshot\_Task method.If bringing up the vHost fails, then a cold migration is performed for all the VMs under it.

**VHostInfo class:**

This class implements the interfaces needed for operation related to vHost. Methods like: adding a new vHost, migrating VM from one host to another and invoking recovery thread for VM/VHost is implemented as part of this class.

**AppProperties class:**

This is a helper class which defines a Java property methods to retrieve the user configurable values like vCenter URL, username, password, ping interval time and snapshot interval time. These methods are used by other classes for getting the input configuration values.

## VmWare Components Used:

* Access to VCenters 130.65.132.19 and 130.65.132.114
* Vhosts with ESXi installed.
* Virtual Machine running on Ubuntu 32bit and VMware tools installed.

## Key Workflows:

* AvailabilityManager starts all the application threads: Snapshot Thread and PingVM thread.
* Snapshot thread creates snapshot of all the alive VMs and VHost for a given periodic interval.
* PingVM thread gets the list of all VHosts and perform ping for each virtual machines under each VHost. If there’s an alarm raised for a VM, then no action is taken. If ping to VM is successful, then no action is taken.
* If ping to VM fails, then its respective vHost is retrieved. Check if its vHost is reachable through ping. If ping to vHost fails, then revert the vHost from its snapshot and all its VMs too. If ping to vHost is successful, then revert only the VM from its snapshot.
* If reconnect the new vHost fails, then a cold migration of all the VMs is to be performed to another host.
* Maintain a list of active vHost and its VM using the VHostInfo object type.

# IMPLEMENTATION

Environment**:**

* Class VCenter - 130.65.132.19 and Team VCenter - 130.65.132.114
* 3 VHosts (130.65.133.21, 130.65.133.22 & 130.65.133.23) with ESXi installed.
* Virtual Machine running on Ubuntu 32bit and VMware tools installed.
* VSphere Management Client on Windows.

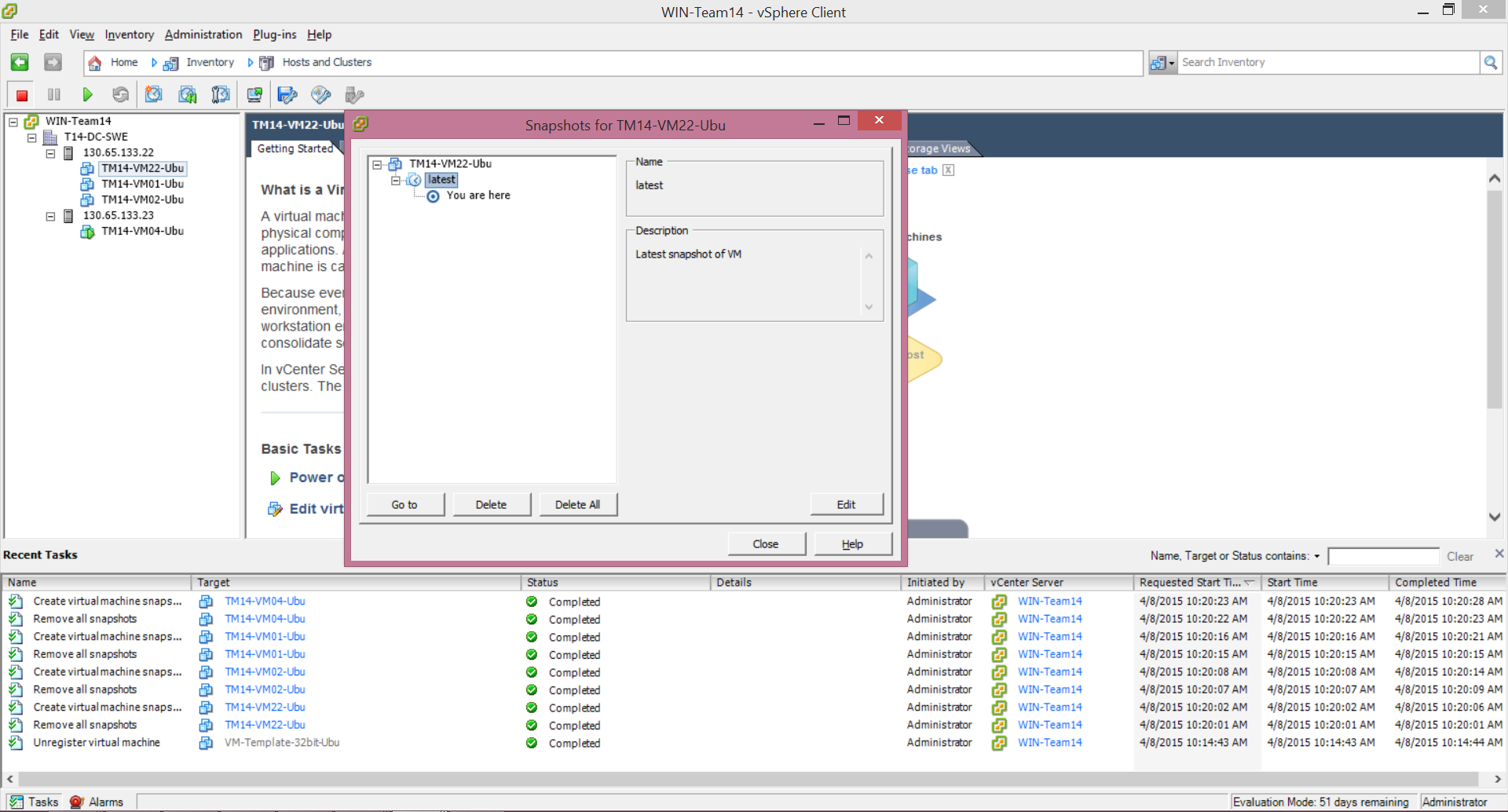
## Tools:

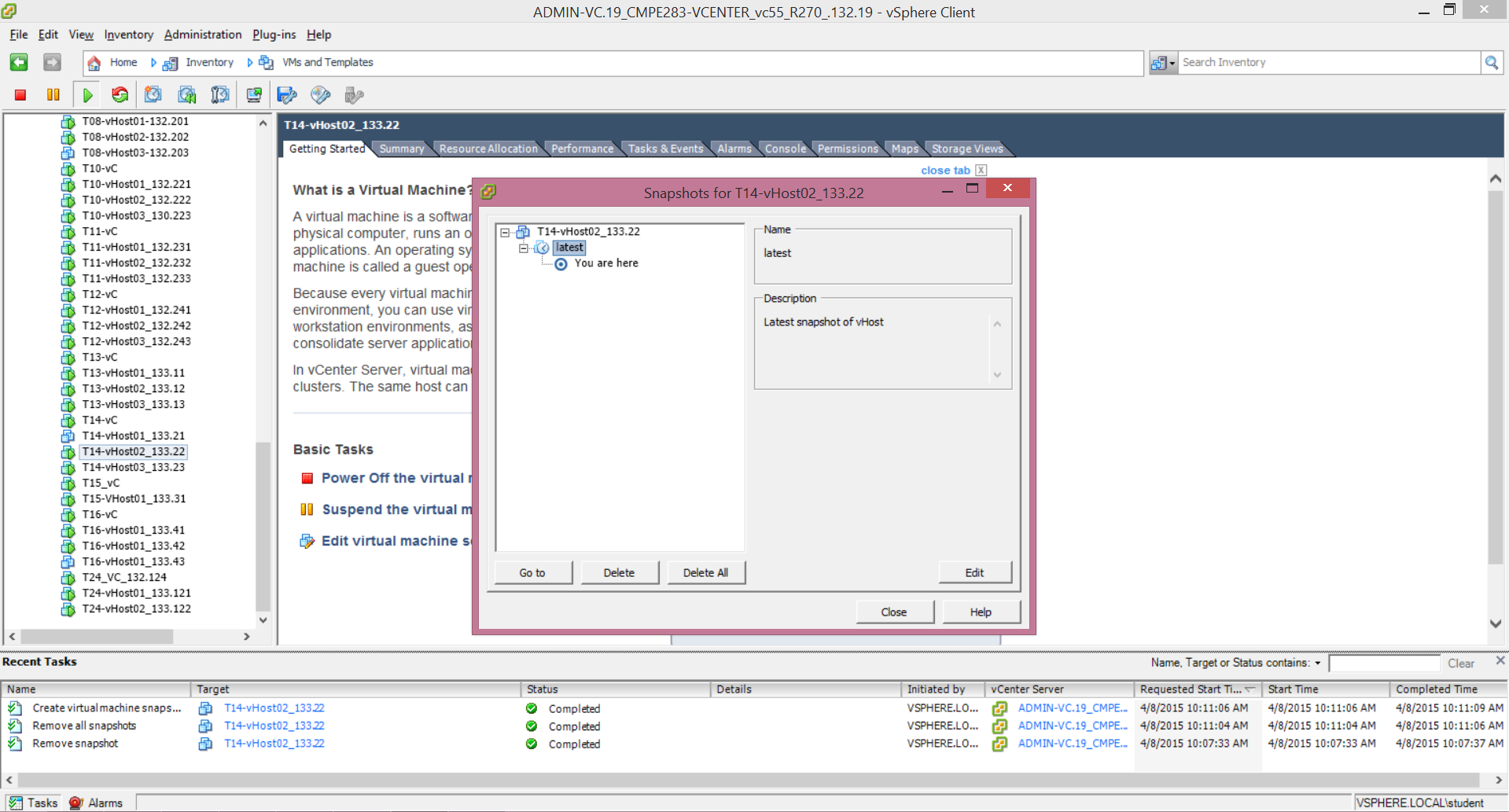
* VSphere client
* Eclipse IDE and VI APIs.
* Windows Net Beans IDE

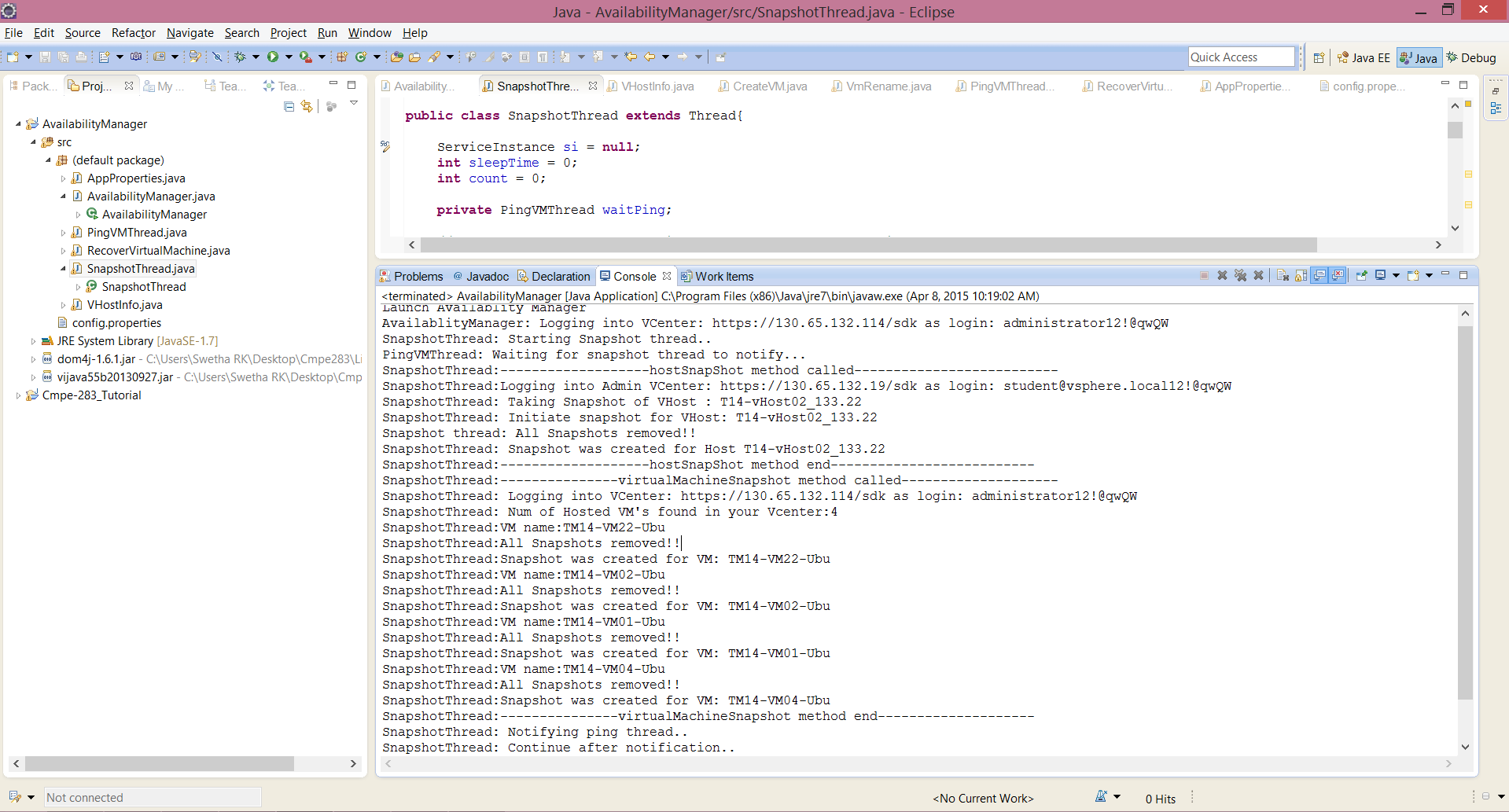
## Screenshots:

Below are the screenshots for important scenarios:

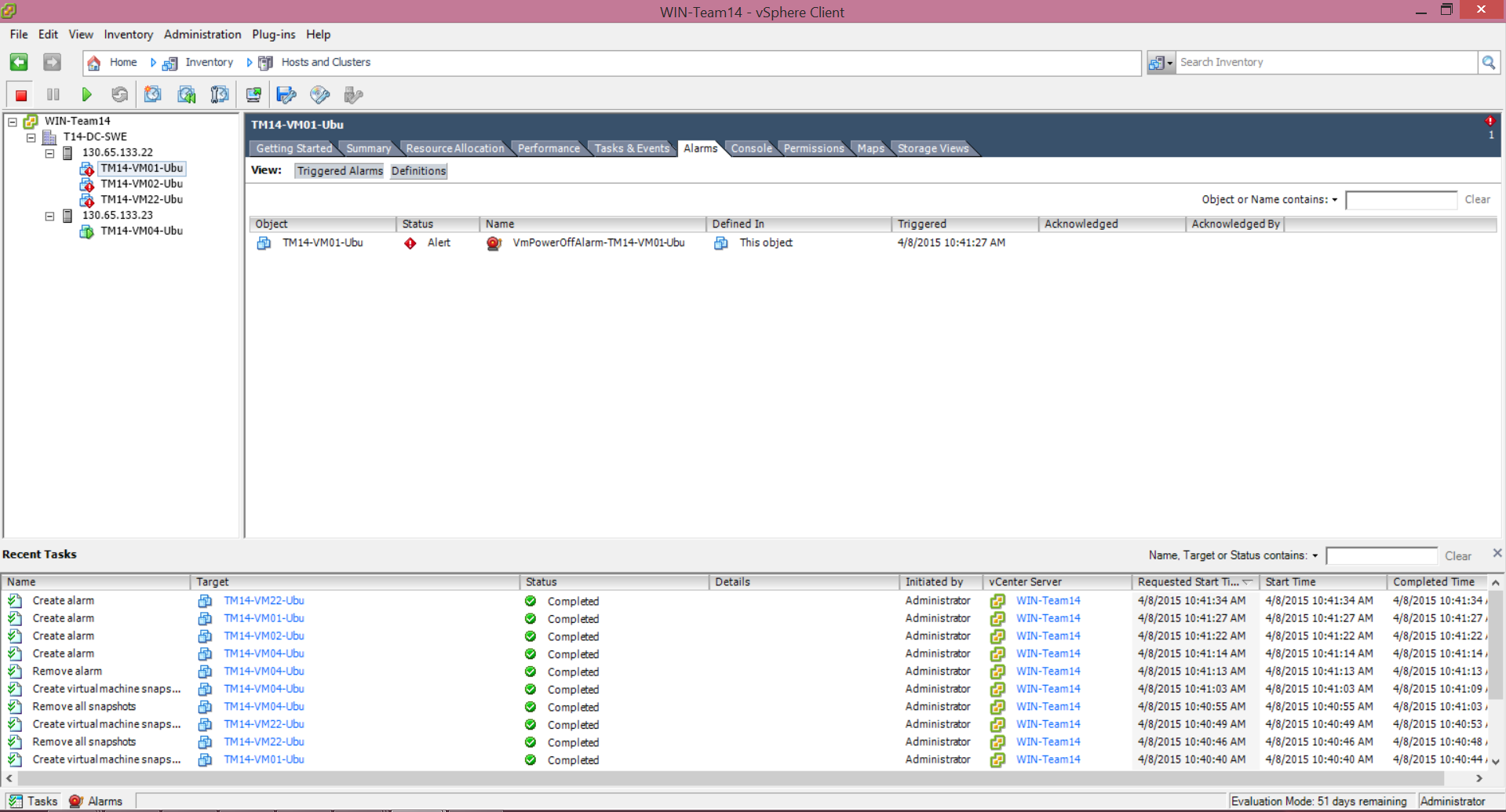
**1. Snapshot creation of VMs and VHost every 10 mins.**

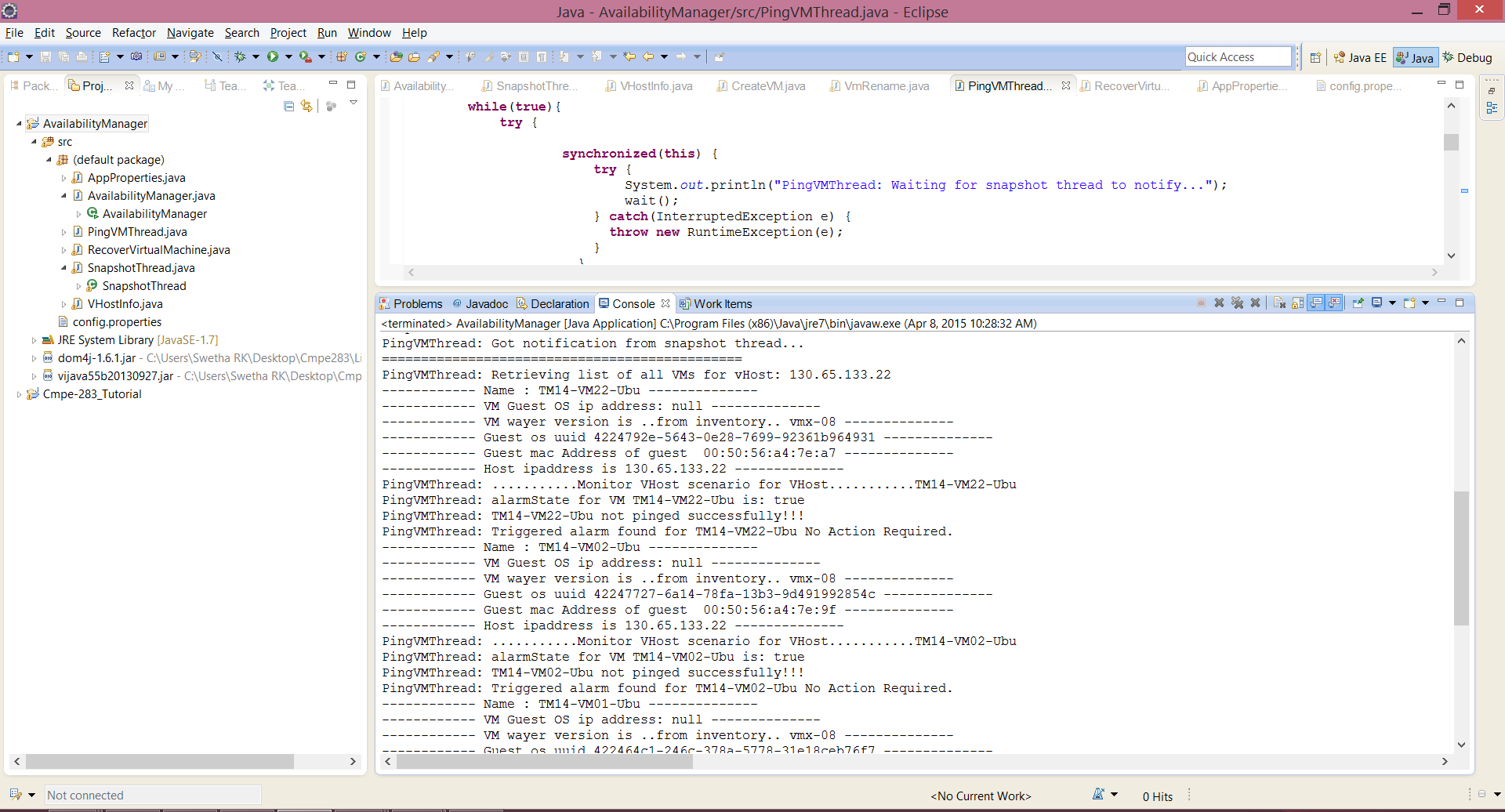
****

****

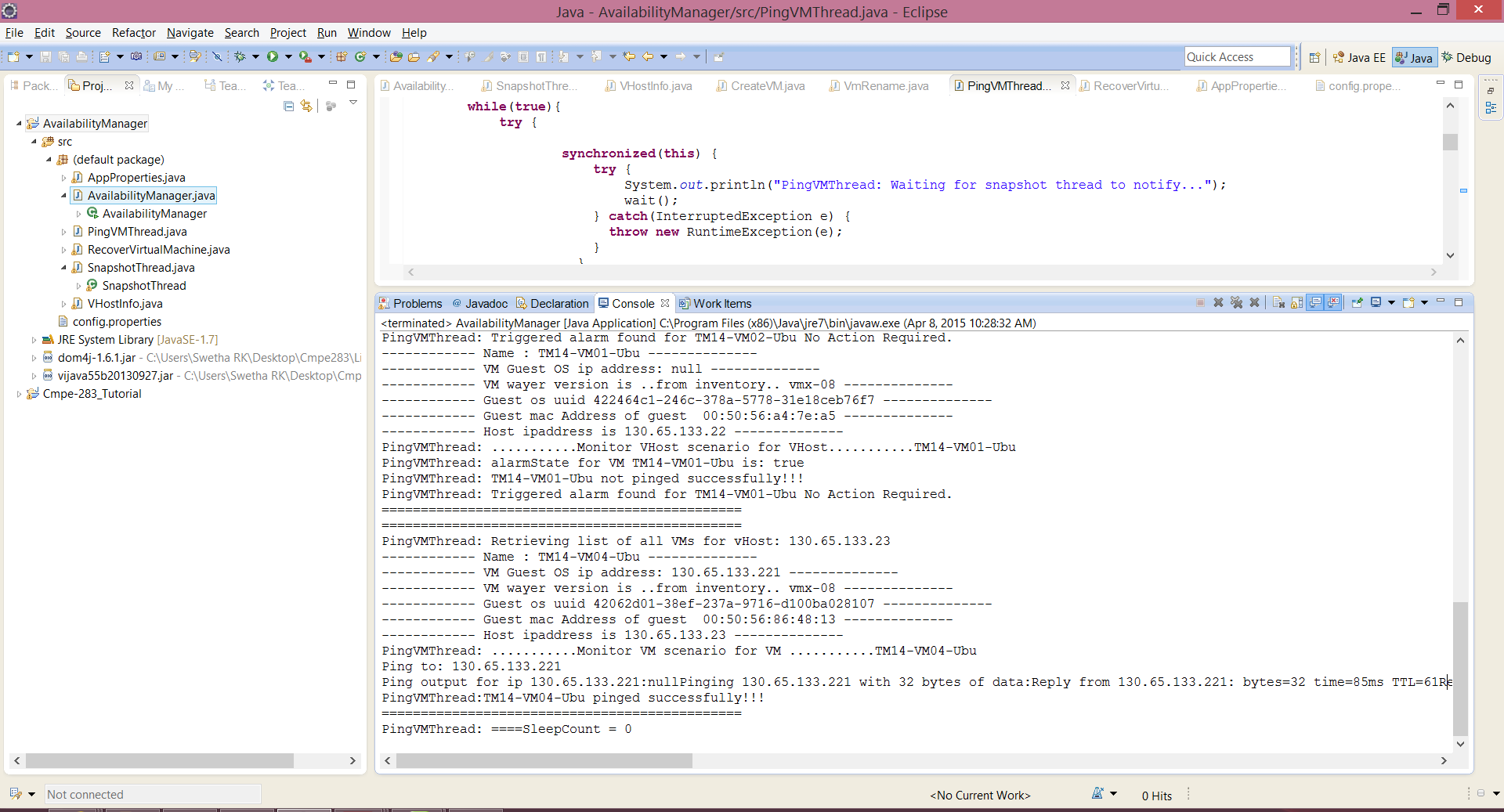
****

**2. Alarm creation for VM power off and no action taken.**

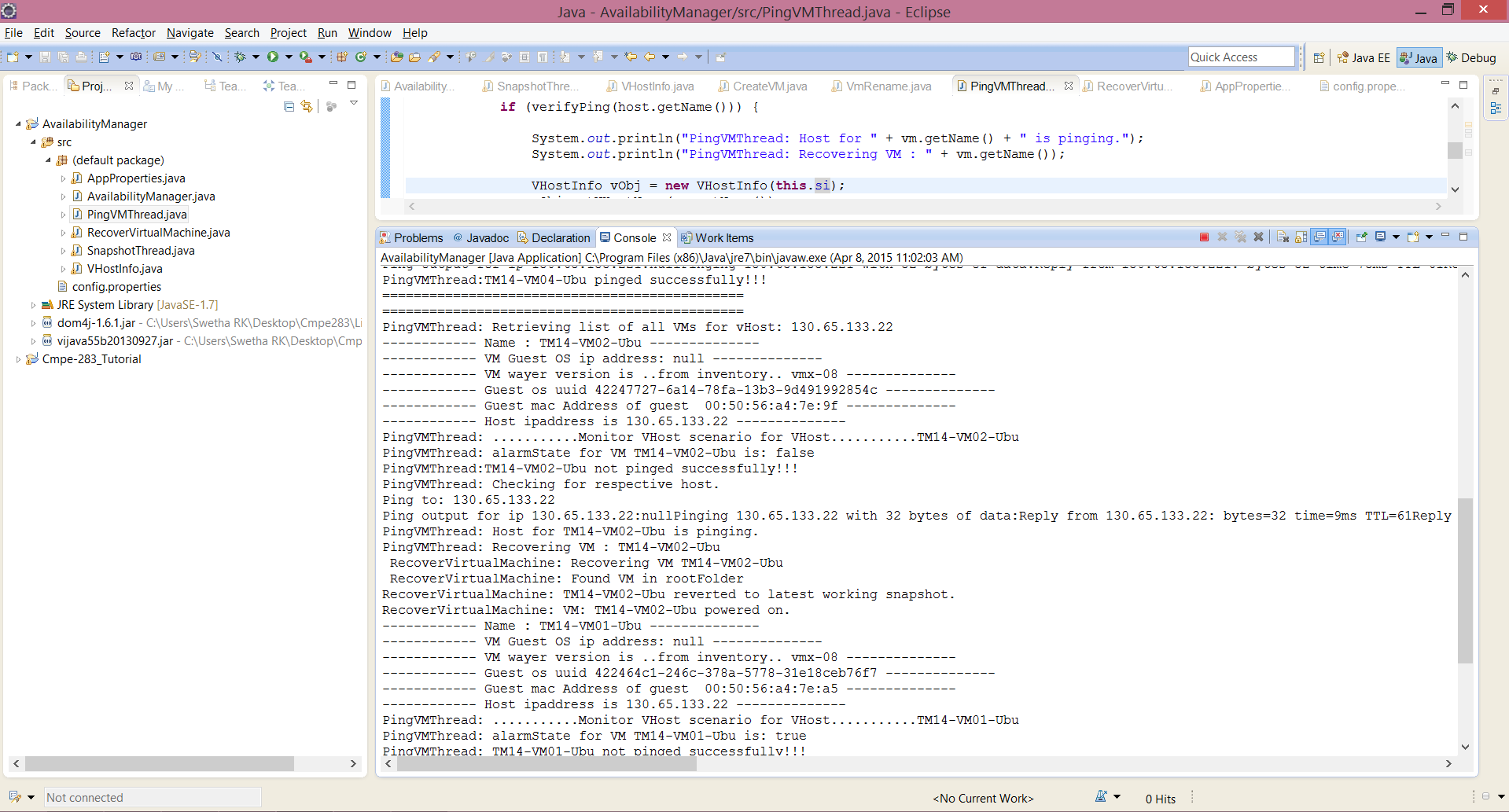
****

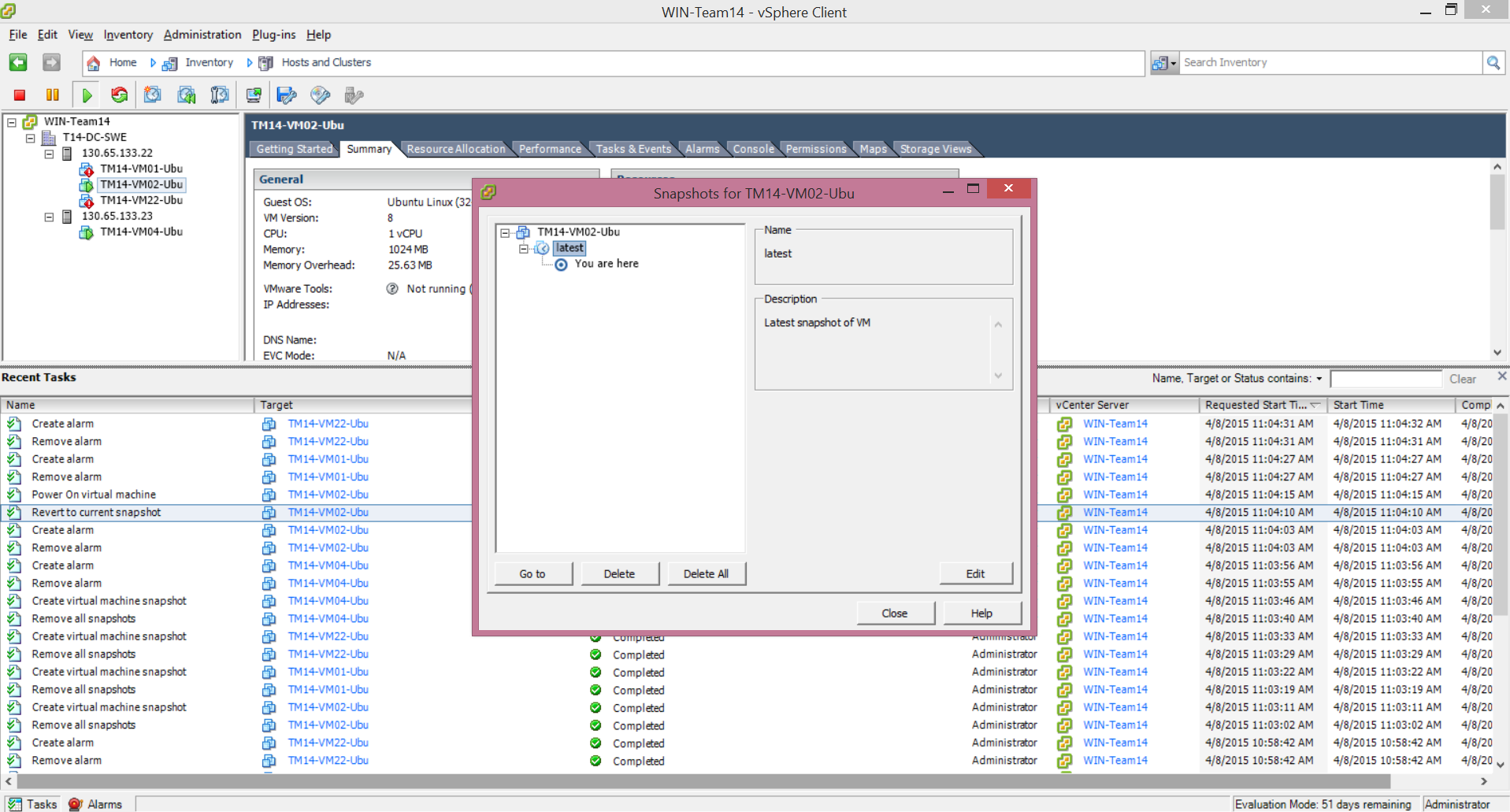
****

**3. Ping to an Alive VM and no action is required.**

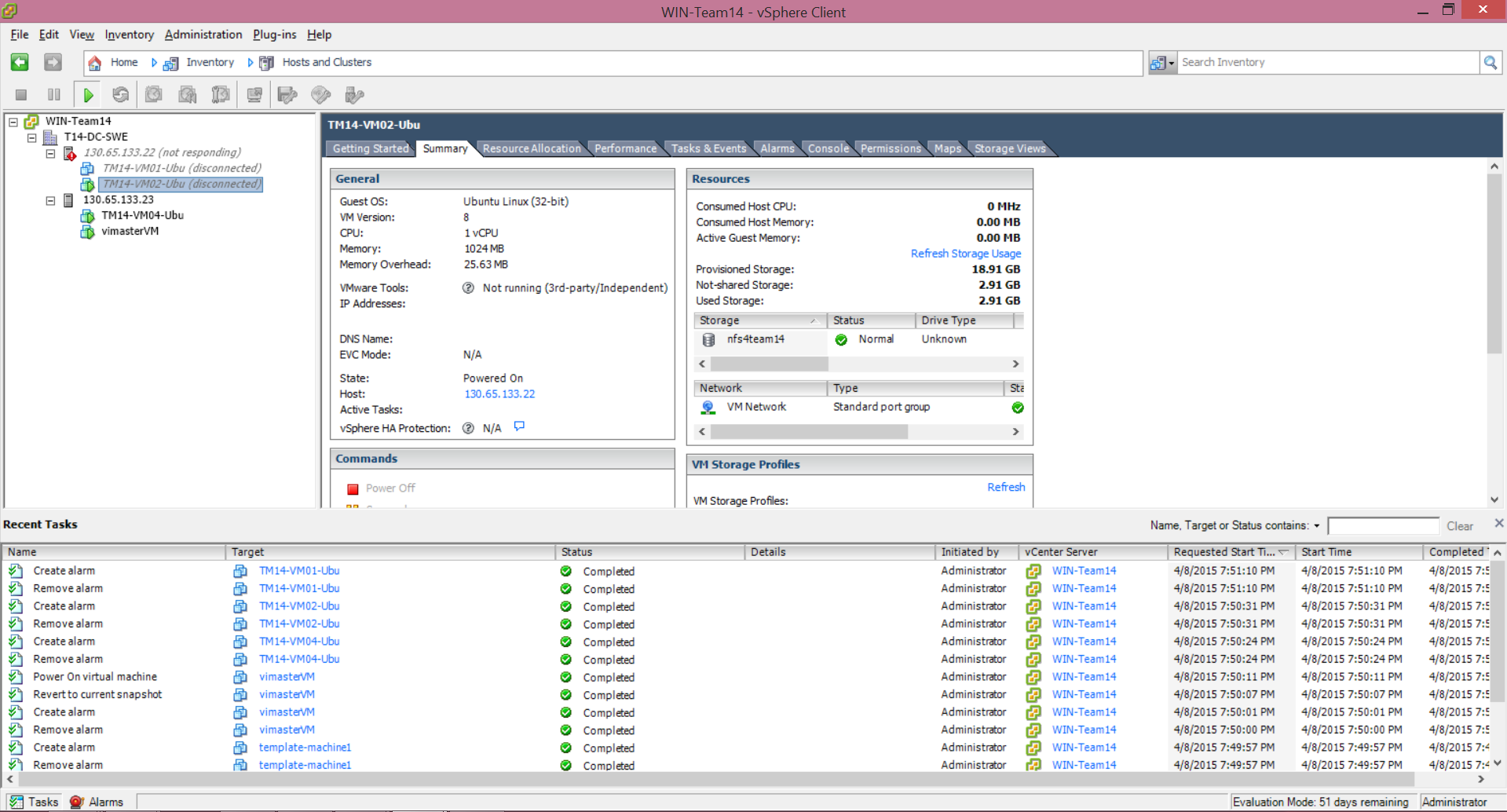
****

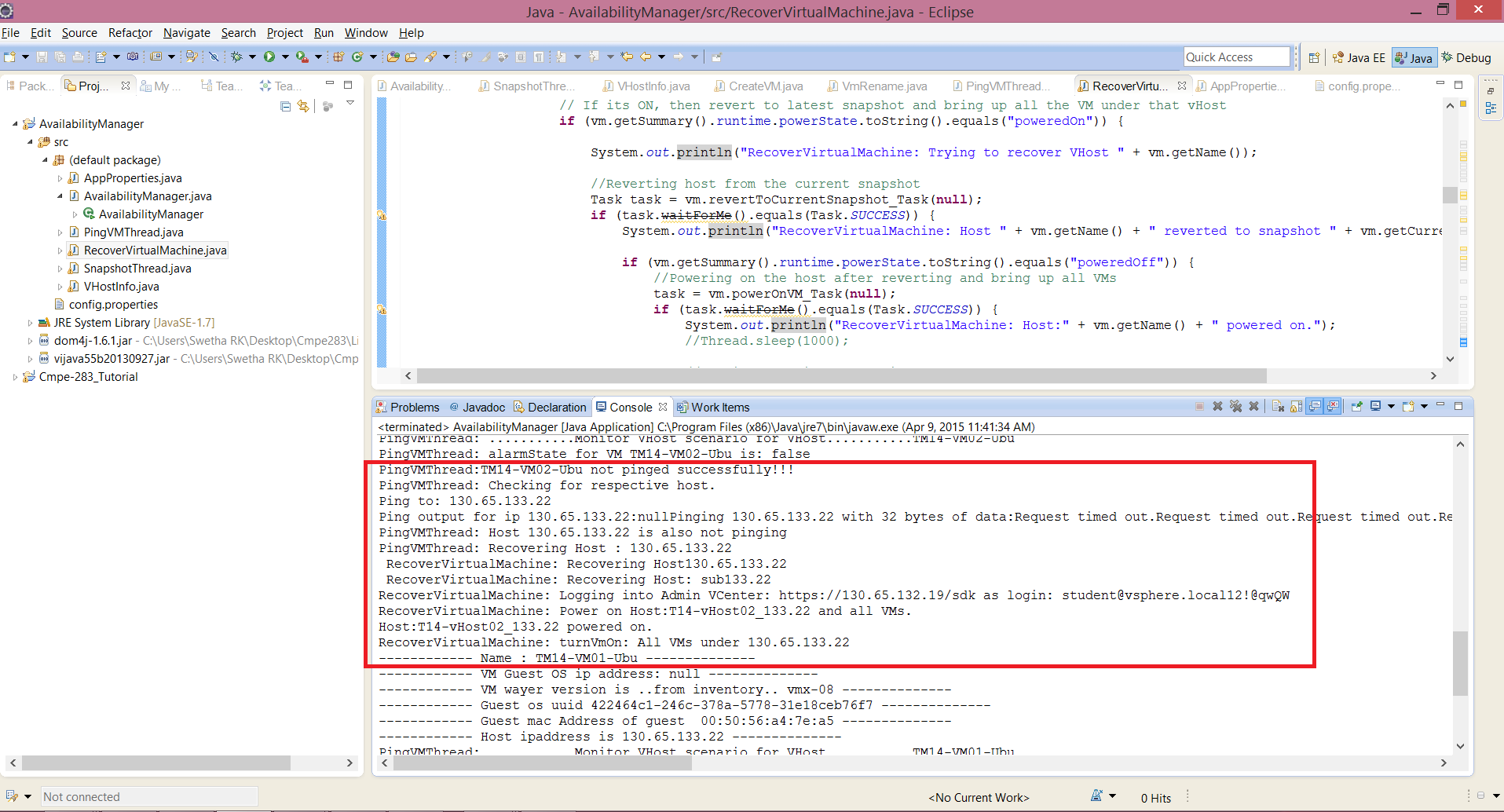
**4. Ping scenario when VM is down and its VHost is up. Then, recover VM from Snapshot**

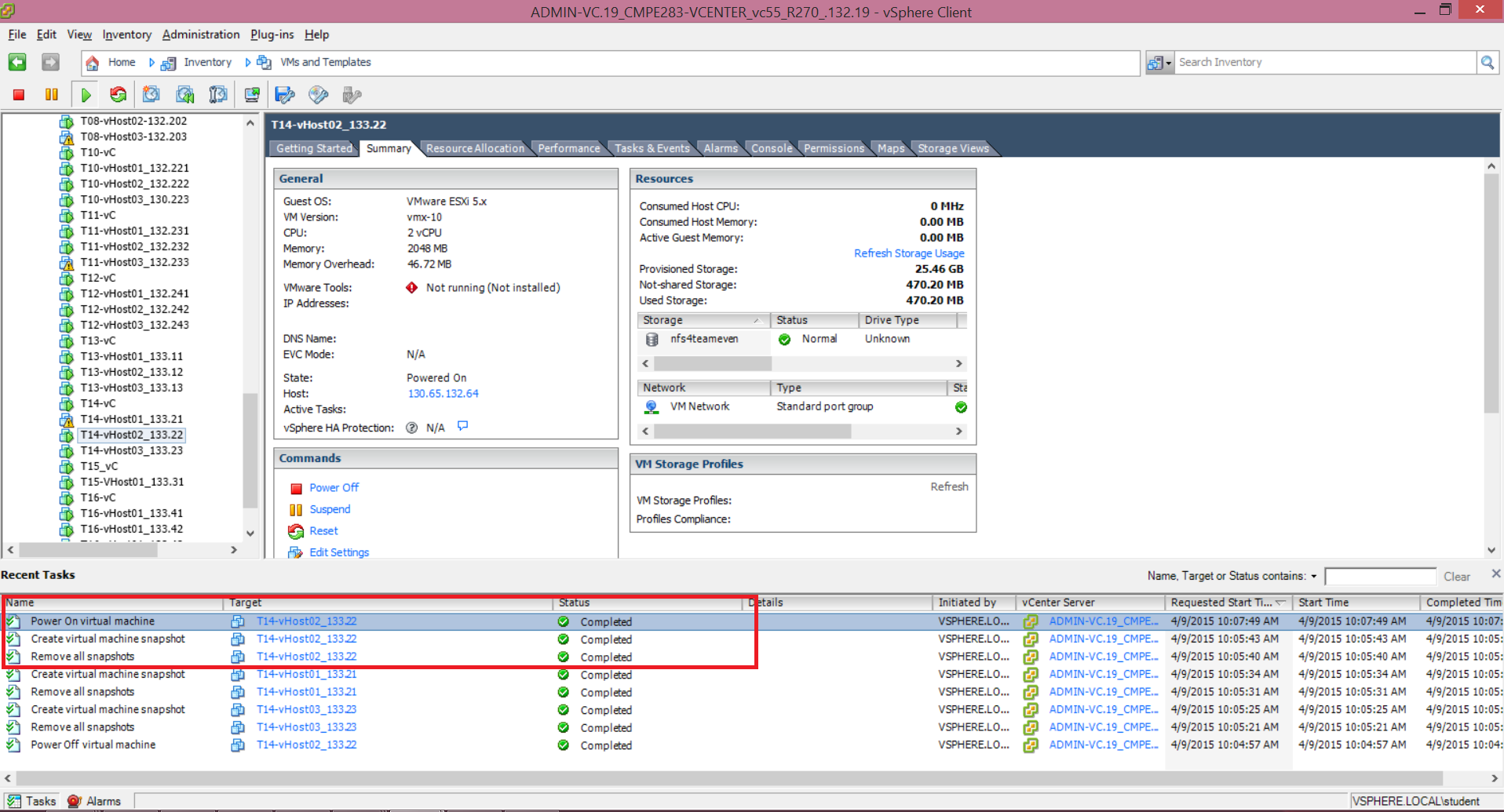
****

****

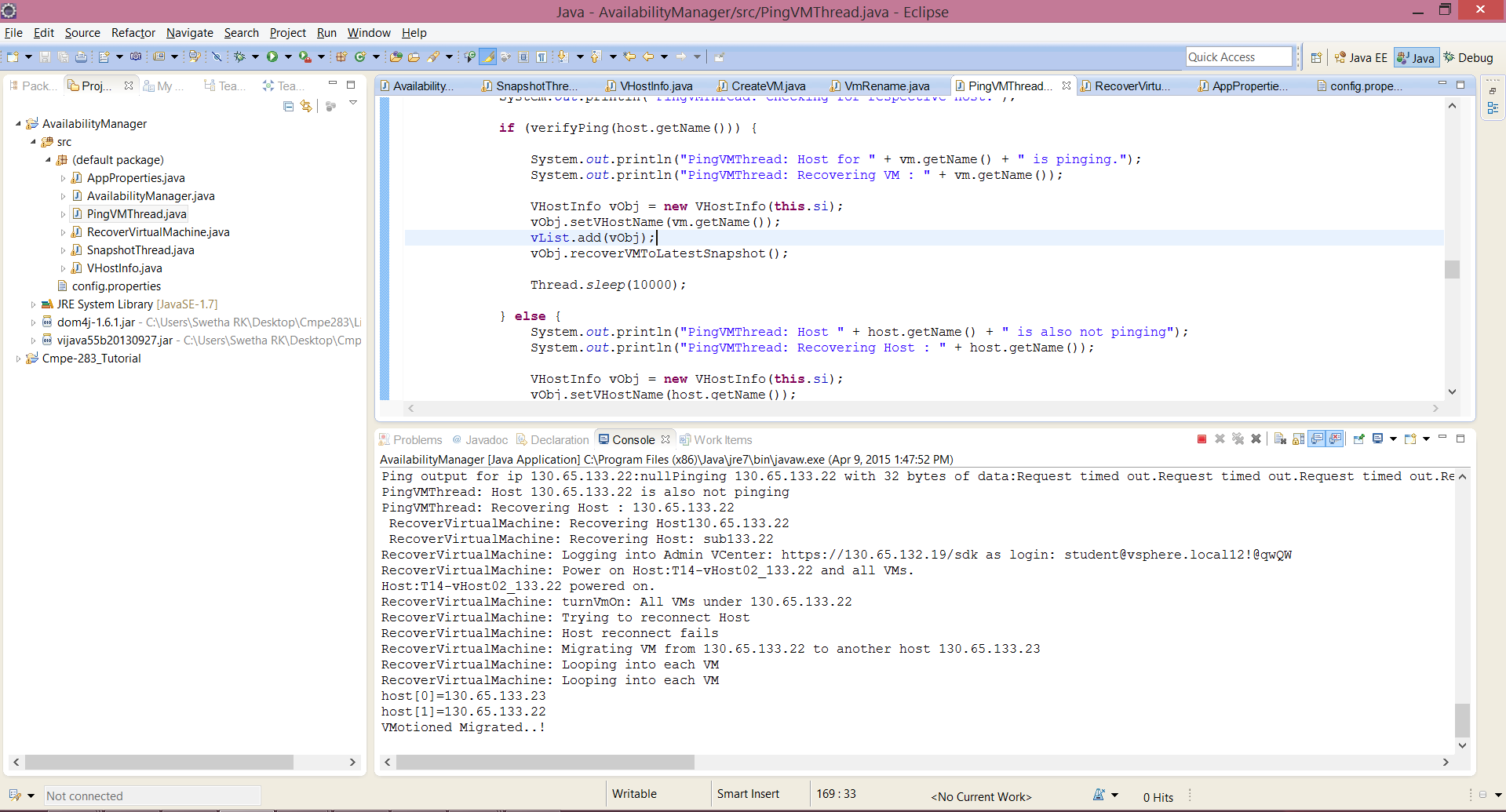
**5. Ping scenario when VM is down and VHost is down. Recover VHost from snapshot and its VMs.**

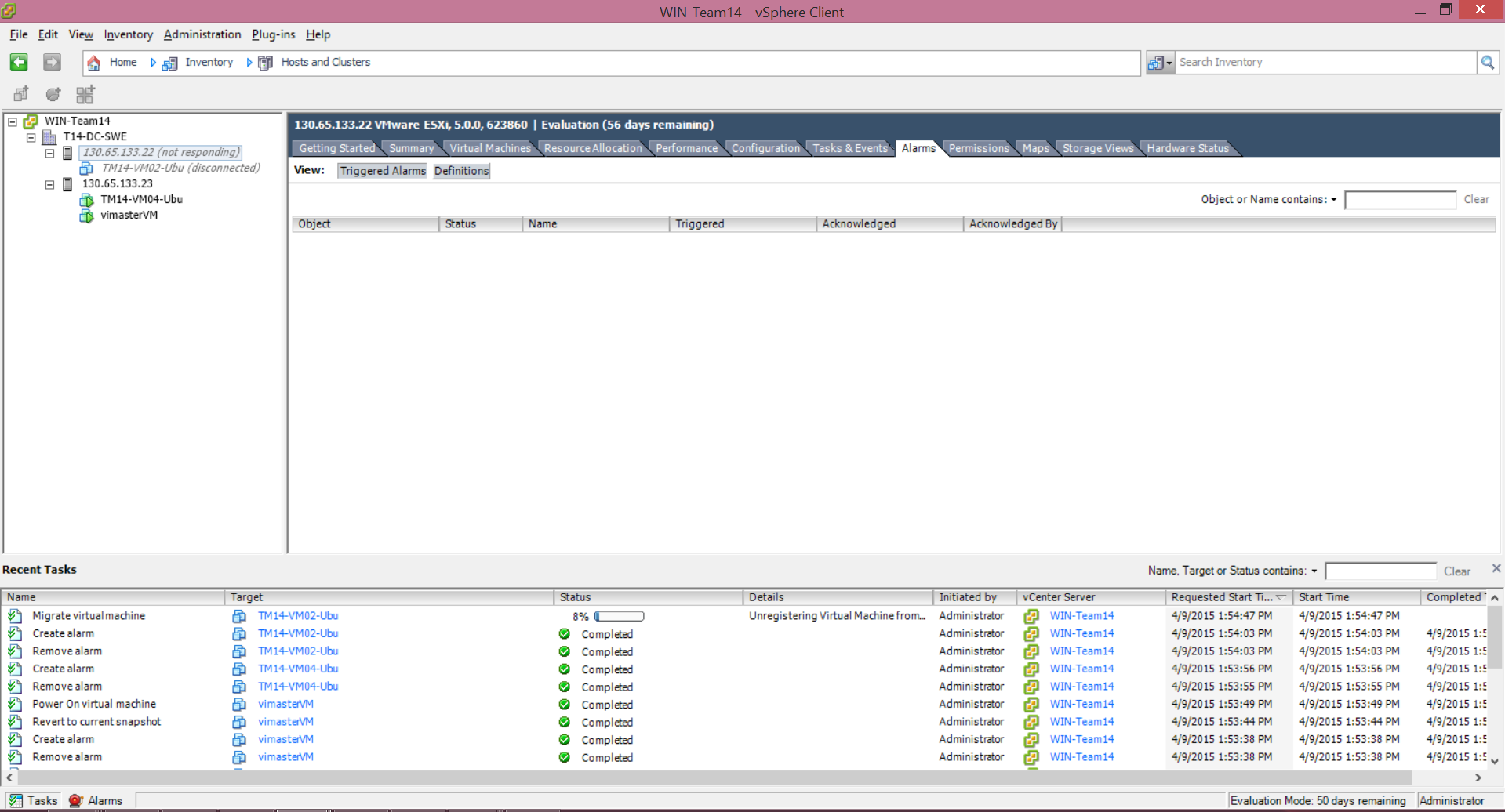
****

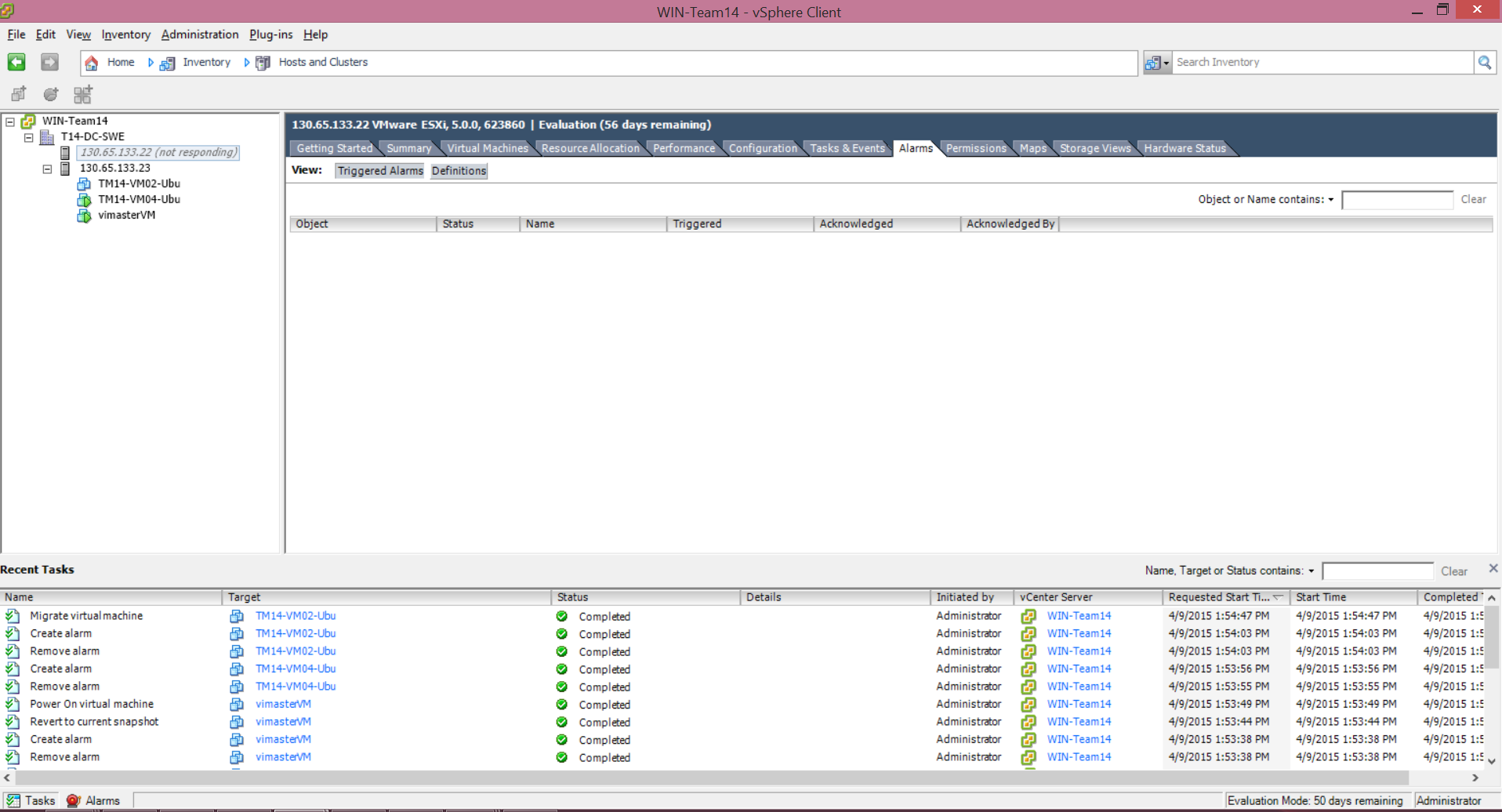
****

****

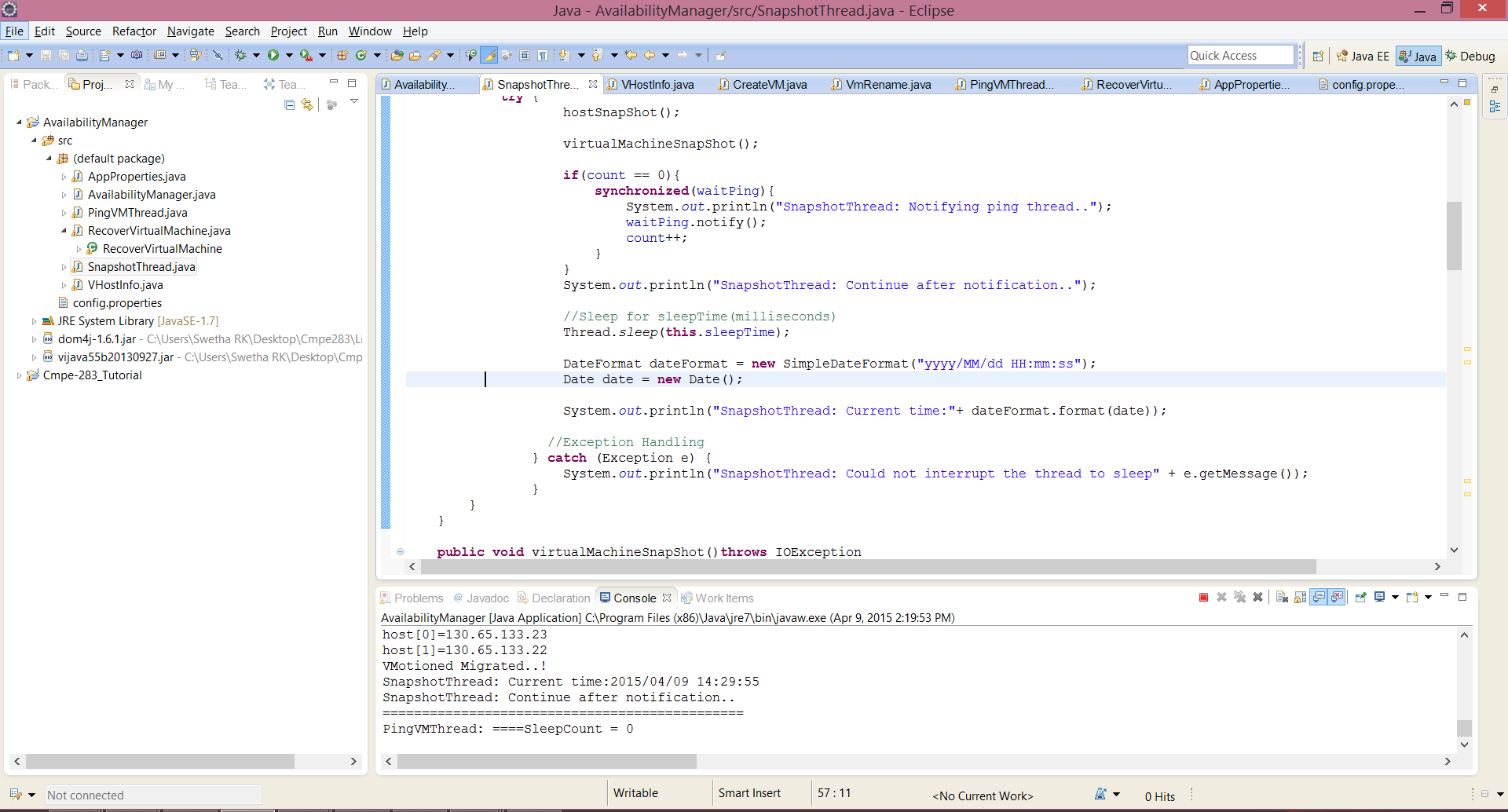
**6. Migrate to another host if VHost is not alive.**

****

****

****

**7. Refresh backup cache every 10 mins**

****

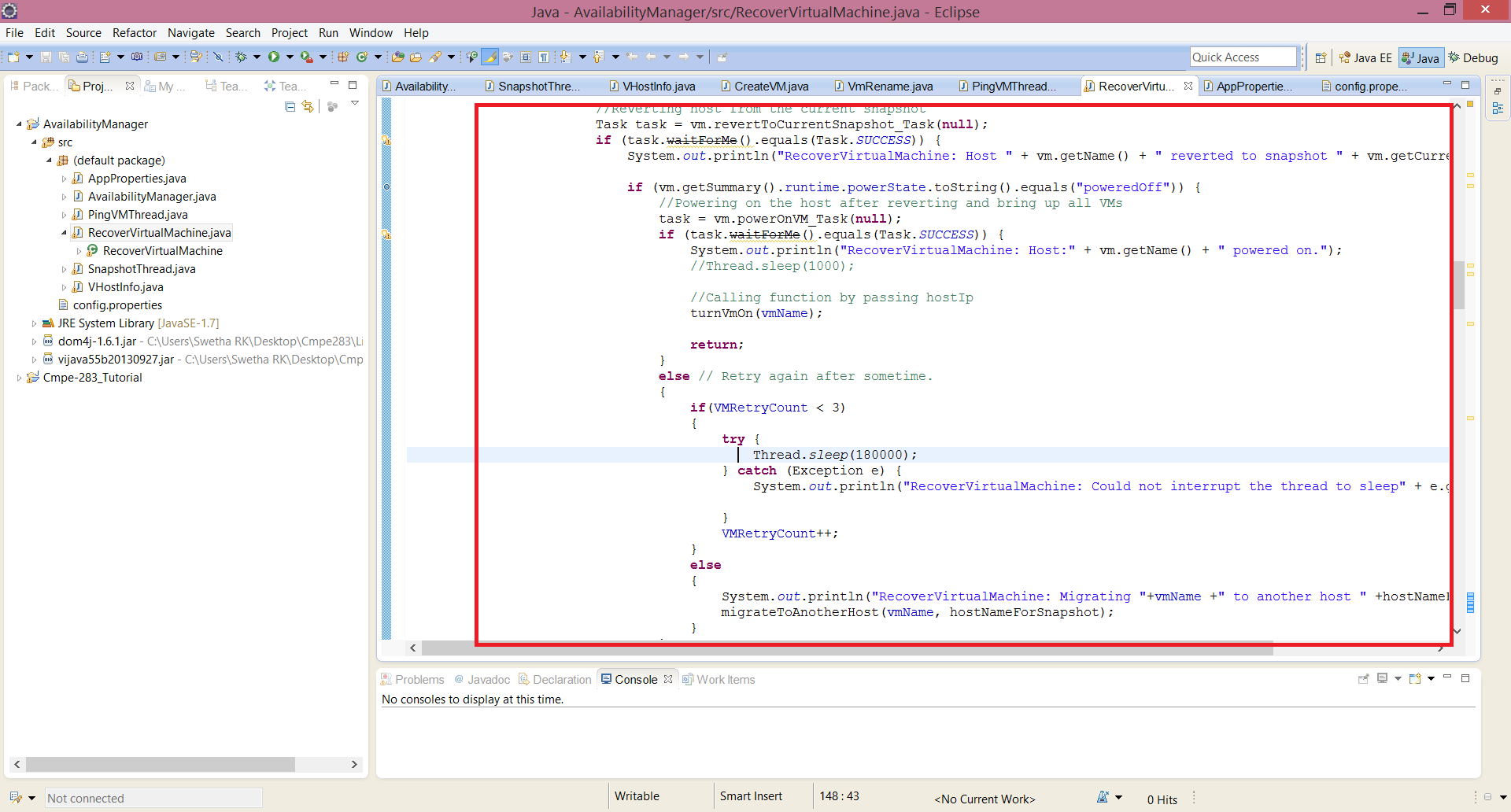
# QUESTIONS

1. Briefly explain the design of your Availability Manager with the help of a class diagram. Also explain the number of threads you have used for the Availability Manager.

Answer: This part is explained in [Design section and part of AvailabilityManager class](#_DESIGN).

2. How does your availability manager handle the scenarios where in the vHost itself is found not to be alive?

Answer: In case of VHost is not alive, the availability manager creates a recovery thread (RecoverVirtualMachine class) to revert the VHost to the latest snapshot and bring up all the VMs under it. If the host recovery fails after 3 times, then a cold migration of all the VMs to another host is performed. The code snippet is shown below:



3. In case of failure, what is the good approach during Disaster Management of Virtual Machines:

* Check the Host first, then the Virtual Machine
* Check the Virtual Machine, then the host

Justify your answer with sufficient reasons

Answer:

In my opinion, checking for VHost first and then the virtual machine is a good approach for disaster management. Say in the case of VHost is down, it’ll reduce the overhead of pining all the VMs under it. But if we ping each VM first and then its host, then there will lot of ping send to host by each VM, each ping call doing the same check for Host status. So choosing first approach will reduce the redundancy of multiple calls. Also, this approach will take less time to deduct the actual problem if host itself is down, rather than the VMs.

# DISCUSSION POINTS

* **The host add/remove mechanism.**

The VHostInfo class defines Apis for adding and removing hosts. Based on the ping to host status, the host is reverted to latest snapshot Or cold migration is perfomed. In these scenarios, these APIs will be used.

* **The approach used to configure the failure detection for each VM.**

VMWare tools are installed on each of the VM and a ping to all VMs is performed to verify if the VM are reachable. To verify the scenarios of ping not reachable, some of the VMs are not installed with VMWware tools. So ping to such VMs would fail and recovery mechanism is initiated. An alarm is also created for monitoring the manual power off state for each of the VM. In the cases where an alarm is raised for a VM, then no action is performed, but proceeded with the next VM in the loop.

* **How host failures are detected.**

The hosts are also handled in the same way of for each VMs. Ping operation is performed using the host IP address in the scenarios where a particular VM is down.

* **The mechanism used to convert between image formats used by the hypervisors.**

A snapshot thread is created by the AvailabilityManager which maintains snapshots of all the Hosts under Team 14 and its respective VMs. The latest snapshot is taken every 10mins duration with a given name and the old snapshots are removed simultaneously.

# CONCLUSION

Working on this project, gave me an indepth knowledge of the VMWare architecture, usage of VMWare Infrastructure APIs and the basic application on how to maintain and recover VMs and VHost periodically. Implementing basic operations like snapshots management, handling all VM/VHost infrastructure, cold migration, inter-thread communication was challenging.

# REFERENCES

* [http://vijava.sourceforge.net//](http://vijava.sourceforge.net/)
* <http://vijava.sourceforge.net/vSphereAPIDoc/ver5/ReferenceGuide/>
* Project Description document.