DOCUMENTATION

SECRETS PROJECT

INTRODUCTION:

The ‘secret’ is a maven configuration open source project cloned form the Git repository. This secret application can be used to create new text files of type AES-128 and save them with a password which keeps your information as a secret.

PRE-REQUISITES:

IntelliJ IDEA, java SE, GIT, Gradle , SBT, Scala.

About My Project:

INTELLIJ IDEA SETUP AND PROJECT BUILD:

* I downloaded java SE from oracle site and created JAVA\_HOME, JDK HOME, and I setup the path in environmental variables.
* Then i downloaded IntelliJ idea 2017.2.3 community edition and installed it in my PC.
* I cloned an open source project and created a new project in my intelliJ IDEA.I added some jar files like org.apache.commons: commons-lang3:3.6, commons-io: commons-io: 2.5 to resolve the issues with dependencies.
* I imported plug-in required to run the project and verified that all the lifecycle options in my maven project are built successfully.
* Then I built the project using build artifact option which creates a jar.

JUNIT TESTS:

* To create Junit tests for my project, I downloaded Junit jars and added that path as Junit\_Home to the environmental variables.
* I tried to understand the whole project structure of my project and wrote junit tests using assert statements to test if my project is correct and created a folder named tests which contains all my tests. I ran all the tests that I have created and they ran successfully.

GRADLE SCRIPT:

* I installed Gradle 2.8 in my PC and set its GRADLE\_HOME path in environmental variables.
* To build the project Gradle imported plug-in and libraries required.
* Then I started writing the gradle script in build.gradle file after understanding the pom.xml file and build.xml file in my maven project.
* I created a lib folder in the project and added my secrets.jar file to it.
* I wrote a task called task runjar in my build.gradle script which runs the jar in my lib folder.
* I wrote a task called task execute which will execute my project by accessing my main class.

Commands used to access gradle:

* ‘gradle build’ to build the project.
* ‘gradle runjar’ to run the jar file that I defined in the build.gradle script.
* ‘gradle execute’ to run the project using main class that I defined in build.gradle script.
* ‘gradle test’ to test the Junit tests in the project.

SBT SCRIPT:

* I installed sbt and scala in my pc, set their paths in environmental variables.
* Then i imported my project as a sbt project.
* Then it generated a build.sbt file in my project.
* I wrote the sbt script adding all the dependencies and wrote a task to access my main class.
* Then i built and ran it.

Commands used to access Sbt:

* ‘sbt compile’ to compile the project.
* ‘sbt run’ to run the project.
* ‘sbt test’ to build the tests.

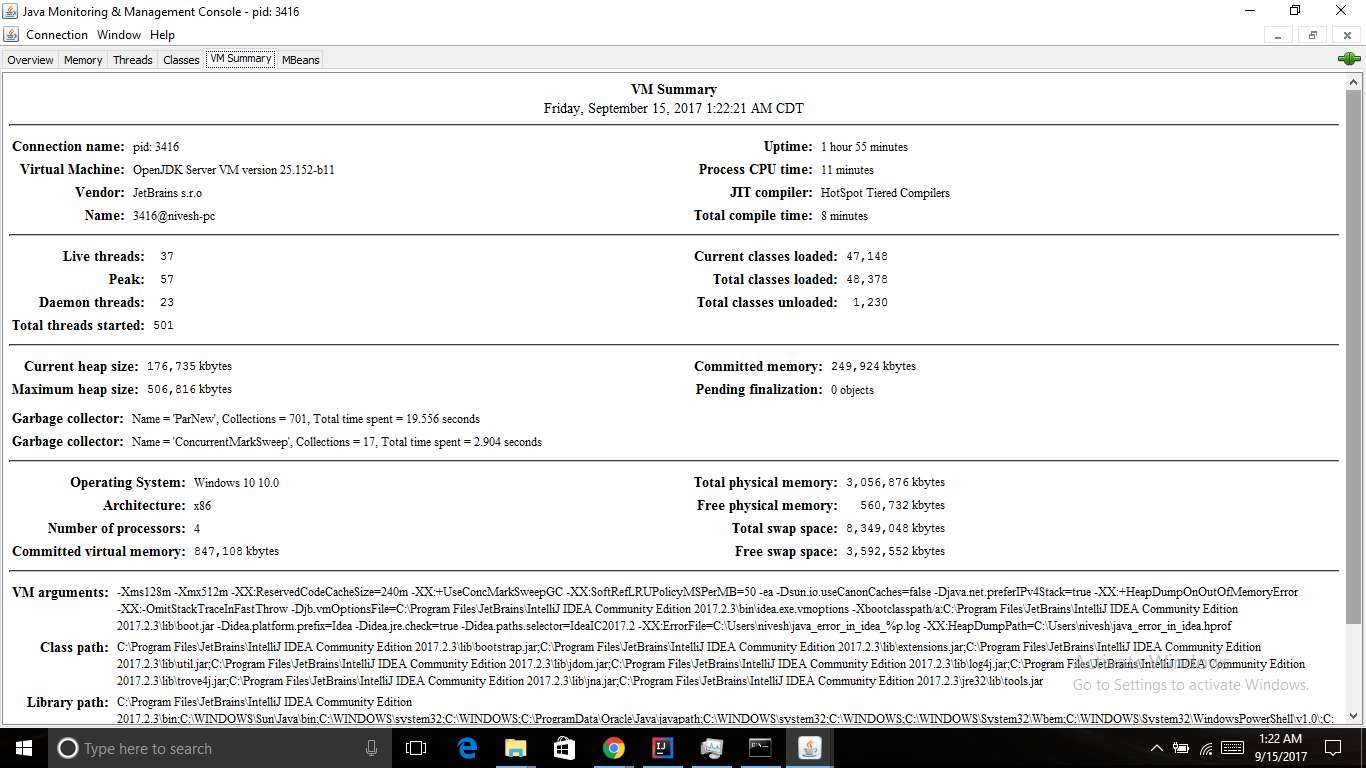
JAVA MONITORING TOOLS:

I used jconsole, visualvm, and java mission control tools to monitor the project.

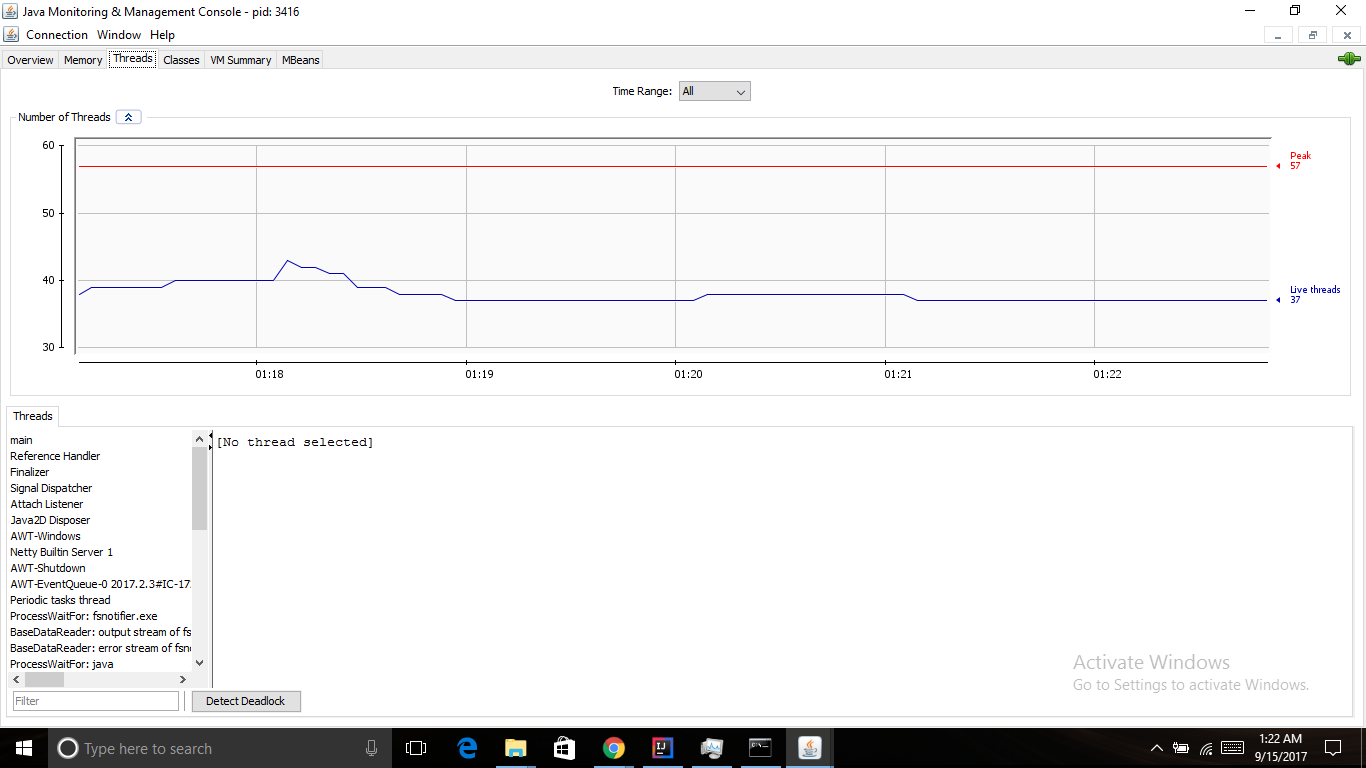
JCONSOLE:

* The jconsole executable can be found in java - > jdk -> bin folder.
* Then to check if it is in my system i used ‘jconsole’ command.
* To access a particular application that is running, we need to know the PID (process id). To know it, i opened my task manager and searched for IJ idea PID and gave a command ‘jconsole pid’.
* Then it opened and I checked all the options that it provides.

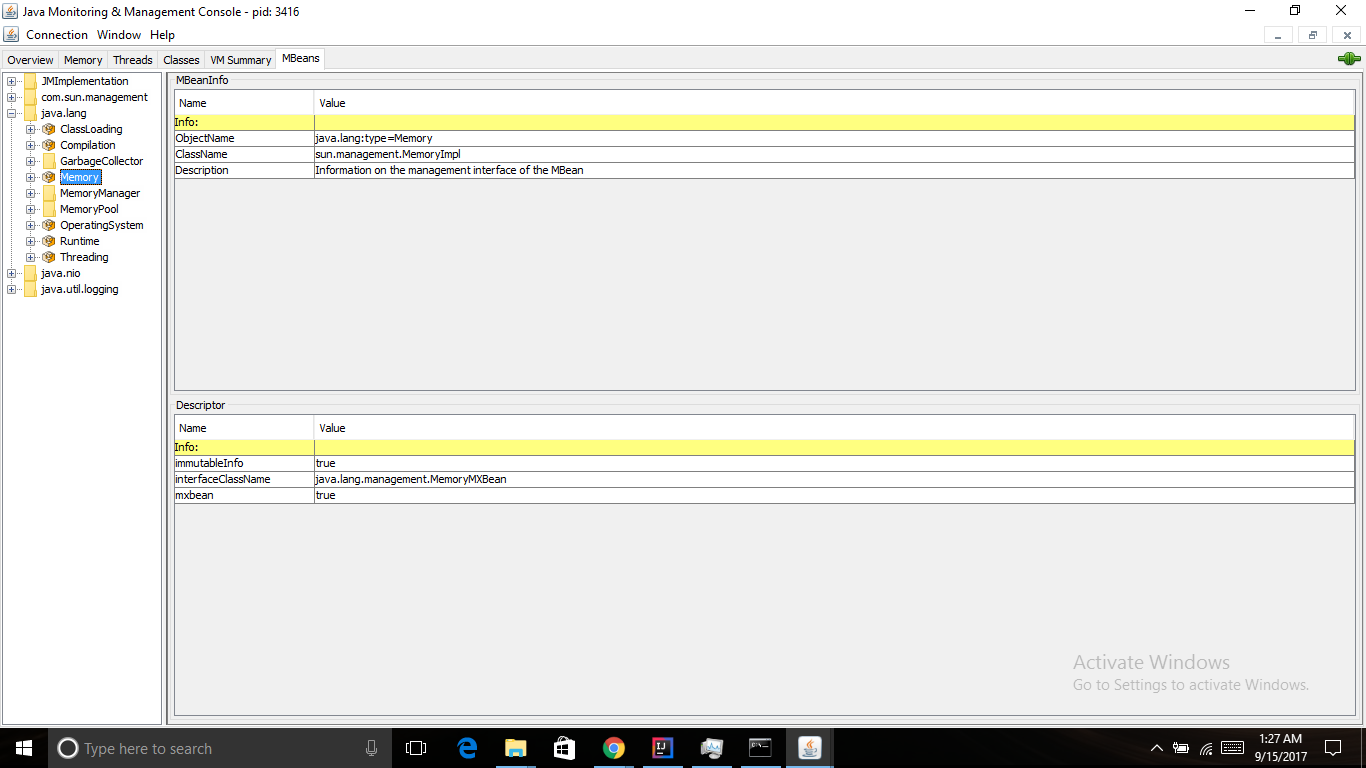
Below are the snapshots of the various tasks that it performs.





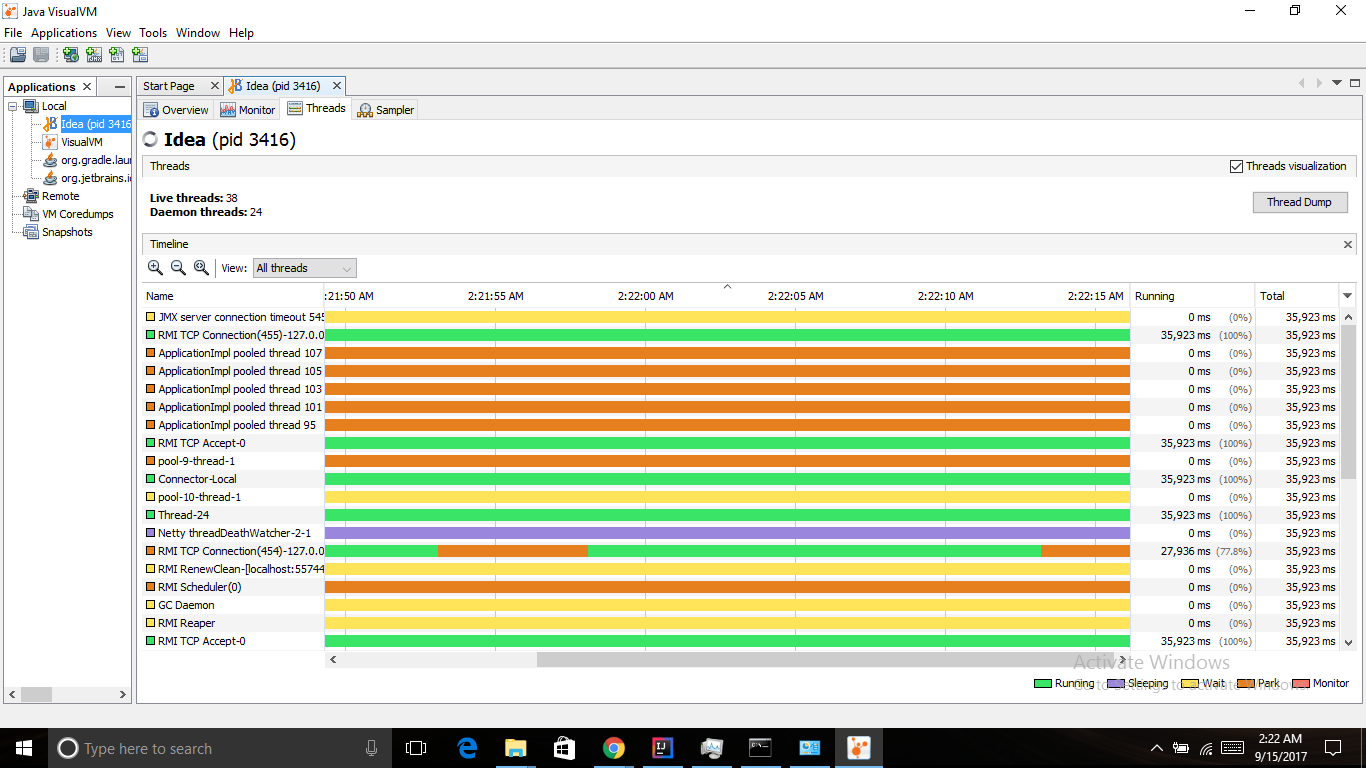


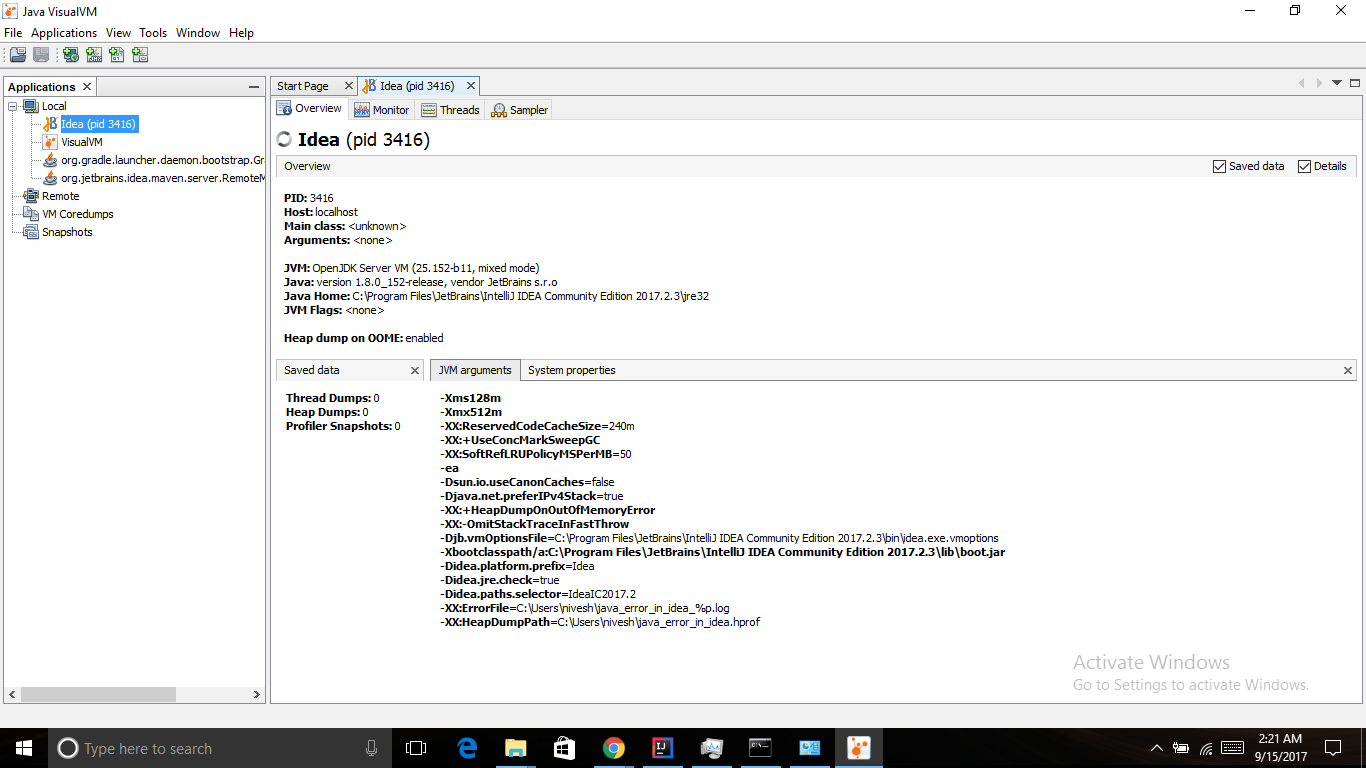


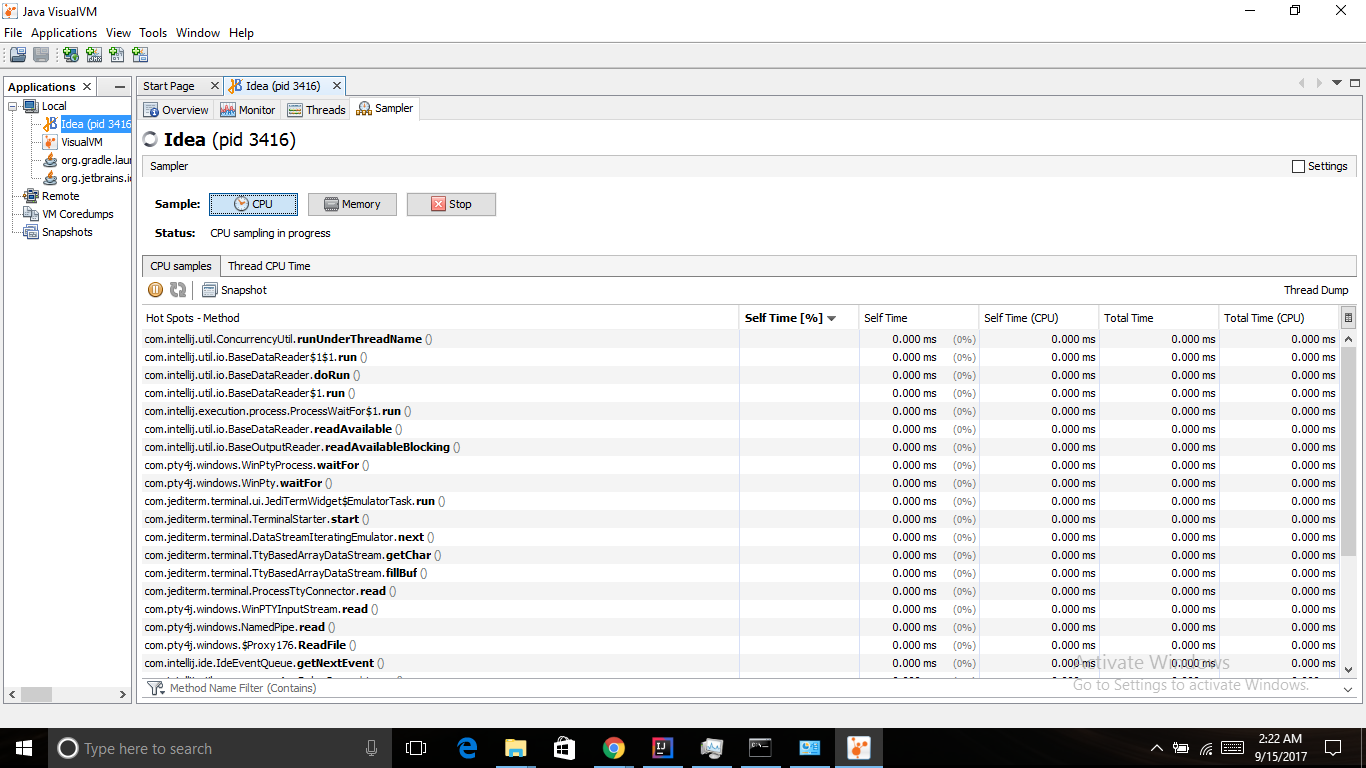


VISUAL VM:

* The visualvm executable can be found in java - > jdk -> bin folder.
* You can use ‘jvisualvm’ command and access it.
* In the menu panel i selected the application that I want to monitor that is Idea and observed these results which I have attached in the below snapshots.







*Java Mission Control*:

* I started the jmc executable from the JDK bin directory.
* Under JVM Browser on the left pane, I have selected the MBean Server.
* Once loaded, it will show a cool dashboard of the JVM CPU/memory usage. I have attached a snapshot of the same.

