Assignment2

1. What is the root cause for OUT OF MEMORY error?

A. java.lang.OutofMemory erroris a runtime error in java, which occurs when the Java Virtual Machine(JVM) is unable to allocate an object due to insufficient space in the java heap.

The Java Garbage Collector(GC) cannot free up the space required for a new object, which causes a java.lang.OutofMemory error.This error can also be thrown when the native memory is insufficient to support the loading of java class.

The Java heap size is determined by two JVM attributes, which can be set when launching Java:

-Xms to set the initial heap size.

-Xmx to set the maximum heap size.

Causes:--

- java.lang.OutofMemory error is a runtime error in java which occurs when the java virtual machine is unable to allocate an object due to insufficient space in the java heap.
- The java garbage collector cannot free up the space required for a new object, which causes a java.lang.OutofMemory error.
- This error can also be thrown when the native memory is insufficient to support the loading of a java class.

2. Xms256m and Xmx2048m what changes done in java 8 for memory utilization?

A. java-Xms256m-Xmx2048m.

- This means, JVM will start up with 256MB of memory and will allow the process to use up to 2048 MB of memory.
- By default, there is no value for Xms,but for Xmx it is 256 MB.you can specify it in multiple formats like kilobytes,megabytes,etc..
- In general , the flag Xmx specifies the maximum memory allocation pool for Java Virtual Machine(JVM).While Xms specifies the initial memory allocation pool.

3.What is memory leakage, how to avoid it?

A. In java the memory leak is a situation when the garbage collector does not recognize the unused objects and they remain in the memory indefinitely that reduces the amount of memory allocated to the application. Because the unused objects still being referenced that may lead to OutOfMemoryError. It also affects the reliability of the application. The following figure represents the memory leak.

Causes of Memory Leaks

- **Using Unwanted Object Reference:** These are the object references that are no longer needed. The garbage collector is failed to reclaim the memory because another object still refers to that unwanted object.

- **Using Long-live Static Objects:** Using static objects also leads to a memory leak. Because they live in the memory till the application's life span.
- **Failure to Clean-up Native System Resources:** Native system resources allocated by a function external to Java. It is written in C and C++. JNI APIs are used to embed native libraries in the Java code.
- **Bugs in the Third-party Libraries:** Bugs in AWT and Java Swing packages are another cause of memory leak.

Preventing Memory Leak:

- Do not create unnecessary objects.
- Avoid String Concatenation.
- Use String Builder.
- Do not store a massive amount of data in the session.
- Time out the session when no longer used.
- Do not use the System.gc() method.
- Avoid the use of static objects. Because they live for the entire life of the application, by default. So, it is better to set the reference to null, explicitly.
- Always close the ResultSet, Statements, and Connection objects in the finally block.

Example:

```
import java.util.Vector;

public class MemoryLeakExample

{

public static void main(String[] args)

{

Vector v1 = new Vector(314567);

Vector v2 = new Vector(876543987);

System.out.println("There is no memory leak in this program.");

}

}
```

4.What is Memory Model in java ?

- The Java memory model specifies how the Java virtual machine works with the computer's memory (RAM).
- The Java virtual machine is a model of a whole computer so this model naturally includes a memory model - AKA the Java memory model.
- The Java memory model specifies how and when different threads can see values written to shared variables by other threads, and how to synchronize access to shared variables when necessary.
- The original Java memory model was insufficient, so the Java memory model was revised in Java 1.5. This version of the Java memory model is still in use in Java today (Java 14+).

The Internal Java Memory Model

The Java memory model used internally in the JVM divides memory between thread stacks and the heap. This diagram illustrates the Java memory model from a logic perspective.

5.What is string pool?

A. String pool is nothing but a storage area in Java heap where string literals stores. It is also known as String Intern Pool or String Constant Pool.

- It is just like object allocation. By default, it is empty and privately maintained by the Java String class. Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also.
- The JVM performs some steps during the initialization of string literals that increase the performance and decrease the memory load. To decrease the number of String objects created in the JVM the String class keeps a pool of strings.
- When we create a string literal, the JVM first check that literal in the String pool. If the literal is already present in the pool, it returns a reference to the pooled instance. If the literal is not present in the pool, a new String object takes place in the String pool.

Example:

```
public class StringPoolExample

{

public static void main(String[] args)

{

String s1 = "Java";

String s2 = "Java";

String s3 = new String("Java");

String s4 = new String("Java").intern();

System.out.println((s1 == s2)+", String are equal."); // true

System.out.println((s1 == s3)+", String are not equal."); // false

System.out.println((s1 == s4)+", String are equal."); // true

}

}
```

Creating String in Java

- There are two ways to create a string in Java:
  - Using String Literal
  - Using new Keyword

Using String Literal:

```
String str1 = "Python";

String str2 = "Data Science";

String str3 = "Python";
```

Using new Keyword: In Java, a new keyword is also used to create String, as follows:

```
String str1 = new String ("Java");

String str2 = new String ("C++");

String str3 = new String ("Data Science");
```

Let's understand what is the difference between them. Let's compare the string literals' references.

```
s1==s3 //true
```

```
s2==s3 //false
```

Let's see how we found that equal or not.

String Pool in Java

First, we have created a string literal Python and it takes place in the pool. After that, the string Data Science is created, it also takes place in the pool. At last, again we have created the string Python. But at this time, JVM checks for the string and found that string literal is already present. Instead of taking a new instance in the String pool, it returns the reference of the pooled instance i.e. str1.