**Stack and Heap**

1.

package com.journaldev.test;

public class Memory {

public static void main(String[] args) { // Line 1

int i=1; // Line 2

Object obj = new Object(); // Line 3

Memory mem = new Memory(); // Line 4

mem.foo(obj); // Line 5

} // Line 9

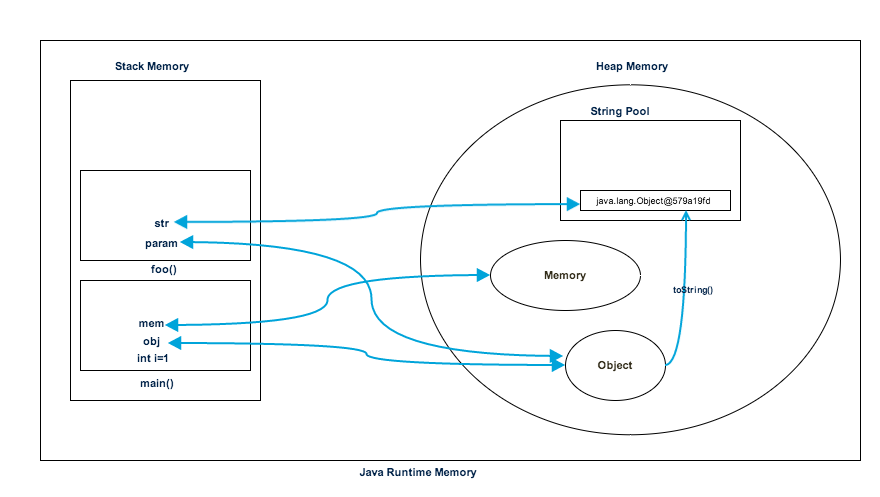
private void foo(Object param) { // Line 6

String str = param.toString(); //// Line 7

System.out.println(str);

} // Line 8

}



2. package java.memoryallocation;

// As soon as this program run it loads the runtime classes into the Heap space.

public class Employee {

    // The main() found creates a Stack memory that will be used by the main() method thread.

    public static void main(String[] args) {

        // This local variable is created and stored in the stack memory of the main() method.

        int index = 1;

        // This object creation is created in the Heap space and the Stack memory contains a reference for it.

        Object myObject = new Object();

        // This object creation is created in the Heap space and the Stack memory contains a reference for it.

        Employee myEmployee = new Employee();

        // Now calling the "someMethod()", a block in the top of the Stack memory is created and is used by

        // the "someMethod()" method. Since Java is pass-by-value in nature, a new reference to Object is created

        // in the "someMethod" stack block.

        myEmployee.someMethod(myObject);

    }   // At this point the "main()" terminates as it has done it job and the Stack space created for

        // "main()" method is destroyed. Also, the program ends and hence the JRE frees all the memory

        // and ends the program execution.

    private void someMethod(Object object) {

        // The string created goes to the "String Pool" that residers in the heap space and the reference of it

        // is created in the "someMethod()" stack space.

        String name = object.toString();

        System.out.println("Name= " + name);

    }   // At this point the "someMethod()" terminates and the memory block allocated for "someMethod()"

        // in the Stack space becomes free.

}

**MCQ’s**

1. Which of the following has the highest memory requirement?  
   a) Heap  
   b) Stack  
   c) JVM  
   d) Class

Answer:c  
Explanation: JVM is the super set which contains heap, stack, objects, pointers, etc.

1. Where is a new object allocated memory?  
   a)Youngspace  
   b)Oldspace  
   c)Young or Old space depending on space availability  
   d) JVM

Answer:a  
Explanation: A new object is always created in young space. Once young space is full, a special young collection is run where objects which have lived long enough are moved to old space and memory is freed up in young space for new objects.

1. Which of the following is a garbage collection technique?  
   a)Cleanupmodel  
   b)Markandsweepmodel  
   c)Spacemanagementmodel  
   d) Sweep model

Answer:b  
Explanation: A mark and sweep garbage collection consists of two phases, the mark phase and the sweep phase. I mark phase all the objects reachable by java threads, native handles and other root sources are marked alive and others are garbage. In sweep phase, the heap is traversed to find gaps between live objects and the gaps are marked free list used for allocating memory to new objects.

1. What is -Xms and -Xmx while starting jvm?  
   a)Initial;Maximummemory  
   b)Maximum;Initialmemory  
   c)Maximummemory  
   d) Initial memory

Answer:a  
Explanation: JVM will be started with Xms amount of memory and will be able to use a maximum of Xmx amount of memory. java -Xmx2048m -Xms256m.

5. Which exception is thrown when java is out of memory?  
a)MemoryFullException  
b)MemoryOutOfBoundsException  
c)OutOfMemoryError  
d) MemoryError

Answer:c  
Explanation: The Xms flag has no default value, and Xmx typically has a default value of 256MB. A common use for these flags is when you encounter a java.lang.OutOfMemoryError.

1. How to get prints of shared object memory maps or heap memory maps for a given process?  
   a)jmap  
   b)memorymap  
   c)memorypath  
   d) jvmmap

Answer:a  
Explanation: We can use jmap as jmap -J-d64 -heap pid.

1. What happens to the thread when garbage collection kicks off?  
   a)Thethread continues its operation  
   b) Garbage collection cannot happen until the thread is running  
   c) The thread is paused while garbage collection runs  
   d) The thread and garbage collection do not interfere with each other

Answer:c  
Explanation: The thread is paused when garbage collection runs which slows the application performance.

8.Whichof the below is not a Java Profiler?  
a)JVM  
b)JConsole  
c)JProfiler  
d) Eclipse Profiler

Answer:a  
Explanation: Memory leak is like holding a strong reference to an object although it would never be needed anymore. Objects that are reachable but not live are considered memory leaks. Various tools help us to identify memory leaks.

9. Which of the below is not a memory leak solution?  
a)Codechanges  
b)JVMparametertuning  
c)Processrestart  
d) GC parameter tuning

Answer:c  
Explanation: Process restart is not a permanent fix to memory leak problem. The problem will resurge again.

10.Garbage Collection can be controlled by a program?  
a)True  
b) False

Answer:b  
Explanation: Garbage Collection cannot be controlled by a program.