**Which are the different segments of memory ?**

Method area

This part of the JVM memory area is shared by all executing threads. Class elements like constant pool, class fields, constructor codes, method codes, etc. Method area can be considered as a part of the heap area but stores per-class data only. We can say that the method area is responsible for holding class level information.

Heap area

Heap Memory in java is used by java runtime to allocate memory to objects and class during the execution of a java program. Whenever an object is created in java, it gets stored into heap memory. A Process called garbage collection runs on heap memory to free up unnecessary space that is garbage collection removes those objects from the heap area which does not have any references. Heap memory in java is divided into the following parts

Stack memory

As the name signifies, stack memory is based on LIFO (last in first out) principle. Stack memory is used for static memory allocation and each executing thread in a java program has its own stack memory. Whenever a Java method is called, a new block is created in java stack memory to hold local or intermediate variables and references to other objects in the method. As soon as the execution of the method gets completed, the block of memory in the stack becomes empty and is used by the next method. The size of Stack memory is less as compared to heap memory. Here are some of the important features of stack memory.

**What are the difference between composition and inheritance in Java?**

The composition is a design technique in which your class can have an instance of another class as a field of your class. Inheritance is a mechanism under which one object can acquire the properties and behavior of the parent object by extending a class.Composition and Inheritance both provide code reusability by relating class.

**What is string constant pool?**

String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content.

Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the string constant pool with the same content, then it returns the reference of that object. It doesn’t create a new object. If the content is different from the existing objects then only it creates new object.

**What is a servlet?**

A servlet is simply java a class which responds to a particular type of network request - most commonly an HTTP request. Basically servlets are usually used to implement web applications. They are used to process the request, produce the response, then send response back to the web server.

Servlets run in a servlet container which handles the networking side (e.g. parsing an HTTP request, connection handling etc). One of the best-known open source servlet containers is [Tomcat](http://tomcat.apache.org/).

**What is spring frame work?**

Framework is a reusable component which encapsulates lots of prewritten code to solve a problem in a specific domain. As a user, I just need to add my own code into it.

## **What Is a Spring Bean?**

Usually, Spring beans are Java objects that are managed by the Spring container.

Beans are created with the configuration metadata that the users supply to the container.

## **What Is the Spring Container?**

The Spring container is responsible for instantiating, wiring, and managing the Spring beans.

**Life cycle of a bean in Spring?**

1. Bean Definition  
Spring Bean will be defined using stereotype annotations or XML Bean configurations.

2. Bean Creation and Instantiate  
As soon as bean created and It will be instantiated and loaded into ApplicationContext and JVM memory.

3. Populating Bean properties  
Spring container will create a bean id, scope, default values based on the bean definition.

4. Post-initialization  
Spring provides [Aware](https://docs.spring.io/spring/docs/3.0.6.RELEASE_to_3.1.0.BUILD-SNAPSHOT/3.1.0.BUILD-SNAPSHOT/org/springframework/beans/factory/Aware.html) interfaces to access application bean meta-data details and callback methods to hook into the bean life cycle to execute custom application-specific logic.

5. Ready to Serve  
Now, Bean is created and injected all the dependencies and should be executed all the [Aware](https://docs.spring.io/spring/docs/3.0.6.RELEASE_to_3.1.0.BUILD-SNAPSHOT/3.1.0.BUILD-SNAPSHOT/org/springframework/beans/factory/Aware.html) and callback methods implementation. Bean is ready to serve.

6. Pre-destroy  
Spring provides callback methods to execute custom application-specific logic and clean-ups before destroying a bean from ApplicationContext.

7. Bean Destroyed  
Bean will be removed or destroyed from and JVM memory.

Bean life cycle is managed by the spring container. When we run the program then, first of all, the spring container gets started. After that, the container creates the instance of a bean as per the request and then dependencies are injected. And finally, the bean is destroyed when the spring container is closed. Therefore, if we want to execute some code on the bean instantiation and just after closing the spring container, then we can write that code inside the custom init() method and the destroy() method.

**Difference between application Context and Bean Factory?**

Both BeanFactory and ApplicationContext provides a way to get a bean from Spring IOC container by calling getBean("bean name"), BeanFactory and ApplicationContext both are [Java interfaces](http://javarevisited.blogspot.in/2012/04/10-points-on-interface-in-java-with.html) and ApplicationContext extends BeanFactory. 

One difference between the bean factory and application context is that the former only instantiate bean when you call getBean() method while ApplicationContext instantiates Singleton bean when the container is started,  It doesn't wait for getBean to be called.

Usually, the implementations use lazy loading, which means that beans are only instantiating when we directly calling them through the getBean() method.

Beanfactory doesn’t supports annotation based dependency

ApplicationContext supports annotation based dependency

**Explain Flow of Spring MVC ?**

**How update query works using hibernate?**

**What is the difference between deep copy and shallow copy?**

**How to create a thread and write a program to create a thread using Thread class**

**Q1. What are the main OOPS concepts in java and explain one by one?**

**Q2. What is singleton class in java?**

**Q3. Why java is platform independent ?**

**Q4. What is thread ?**

**Q5. Difference between Array and linkedlist ?**

**Q6. Can we override start() method ?**

**Q7. What is Exception and what is Exception Handling in java ?**

**Q8. Difference between (==) and (.equals()) in java ?**