Spring Boot Profiles

Spring Boot Actuator

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Reverse the first n elements\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*

Given Array A. Reverse the first n elements

ex: Array[] A= {4,7,6,9,2,4,3};

o/p : {6,7,4,9,2,4,3}

\*/

**package** com.venkates.InterviewQuestions;

**import** java.util.Arrays;

**public** **class** ReverseTillNthElement {

**public** **static** **void** main(String[] args) {

**int**[] a= {23,5,57,14,51,2,8,3};

**int** n=4;

System.***out***.println(Arrays.*toString*(*reverseTillNthElement*(a,n)));

}

**private** **static** **int**[] reverseTillNthElement(**int**[] a, **int** n) {

**int** startIdx=0;

**int** endIdx = n-1;

**int** tmp=0;

**while**(startIdx < endIdx) {

tmp=a[startIdx];

a[startIdx] = a[endIdx];

a[endIdx] = tmp;

startIdx++;

endIdx--;

}

**return** a;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

How do we make a class immutable ?

<https://www.digitalocean.com/community/tutorials/how-to-create-immutable-class-in-java>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Singleton Design Pattern:**

[**https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples**](https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL query to display the employee names along with salary where the employee’s salaries are the top three.

select top 3 empName , sal from employee order by salary DESC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Difference between string and string buffer ?

String s = new String("Venkates");

s.concat("software");

System.***out***.println("string ...."+ s);

O/P : Venkates

Strings are immutable => Venkates and Venkatessoftware objects are created. But as Venkatessoftware is not assigned to any variable it is eligible for garbage collection.

--------------------------------------------------------------

StringBuffer sb = new StringBuffer("Venkates");

sb.append("software");

System.***out***.println("StringBuffer..."+sb);

O/p : Venkatessoftware

In this case only one object is created and modifications are done on the same object.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

difference between == and equals() on strings:

String s1=new String("Venkates");

String s2=new String("Venkates");

System.***out***.println("... s1==s2"+ s1==s2); // Reference comparison(address comparison)

System.***out***.println("..s1.equals(s2)"+s1.equals(s2)); // content of the objects are compared

equals() method of Object class checks for reference comparison but as the equals() of object class is overridden by String class to check the content of the String objects s1.equals(s2) will return true. Whereas StringBuffer class doesn't override the equals() hence returns false if we call equals method.

StringBuffer sb1=new StringBuffer("Venkates123");

StringBuffer sb2=new StringBuffer("Venkates123");

System.***out***.println(sb1==sb2);

System.***out***.println("......."+ sb2.equals(sb1));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Difference of using new operator and = to while creating strings

// Two objects are created - One in heap area and one in String Constant pool(SCP)

String s=new String("Tanvee");

// Only one oject is created in SCP

String s2="Karthik";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**What is the advantage of String Immutability.**

Immutable objects are simply objects whose state (the object's data) cannot change after construction**.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Grep and locate commands in UNIX.

==============================

Programme for Fibonacci Series.

0 1 1 2 3 5 8 13 21

public class JavaApplication1 {

public static void main(String[] args) {

// int number\_terms = args[0];

// JavaApplication1.fibonacci(number\_terms);

JavaApplication1.fibonacci(15);

}

public static void fibonacci(int n){

int first\_number=0;

int second\_number=1;

int nterms=n;

for(int i=0;i < nterms; i++){

System.out.println(first\_number);

int new\_firstnumber = second\_number;

second\_number = first\_number + second\_number;

first\_number = new\_firstnumber;

}

}

}

**===============================**

public class X   
{   
 public static void main(String [] args)   
 {  
 try   
 {  
 badMethod();   
 System.out.print("A");   
 }   
 catch (Exception ex)   
 {  
 System.out.print("B");   
 }   
 finally   
 {  
 System.out.print("C");   
 }   
 System.out.print("D");   
}   
 public static void badMethod()   
 {  
 throw new Error(); /\* Line 22 \*/  
 }   
}

**A. ABCD**

**B. Compilation Fails**

**C. C is printed before exiting with an error message.**

**D. BC is printed before exiting with an error message.**

**Answer:** Option **C**

**Explanation:**

Error is thrown but not recognised line(22) because the only catch attempts to catch anException and Exception is not a superclass of Error. Therefore only the code in the finally statement can be run before exiting with a runtime error (Exception in thread "main" java.lang.Error).

**Difference between Array and ArrayList:**

1. An array needs to know its size when it is created.

2. To put an object in an array you must assign it to a specific location.

3. Arrays use array syntax that is not used any where else in java.

4. ArrayLists in java 5 are parameterized.

=============================

### Why Wait And Notify Methods Are In Object Class Not In Thread?

It was a design decision.

Java’s concurrency model needs “locks” and it was decided that each object was to be associated with a lock.

The wait and notify methods are not associated with a thread but with a lock, so the decision of coupling locks and objects meant that each object should have wait and notify methods that operate on that object’s lock.

**Explanation and Example**

A gas station has a single toilet, the key for which is kept at the service desk. The toilet is a shared resource for passing motorists. To use this shared resource the prospective user must acquire a key to the lock on the toilet. The user goes to the service desk and acquires the key, opens the door, locks it from the inside and uses the facilities.

Meanwhile, if a second prospective user arrives at the gas station he finds the toilet locked and therefore unavailable to him. He goes to the service desk but the key is not there because it is in the hands of the current user. When the current user finishes, he unlocks the door and returns the key to the service desk. He does not bother about waiting customers. The service desk gives thekey to the waiting customer. If more than one prospective user turns up while the toilet is locked, they must form a queue waiting for the key to the lock. Each thread has no idea who is in the toilet.

Obviously in applying this analogy to Java, a Java thread is a user and the toilet is a block of code which the thread wishes to execute. Java provides a way to lock the code for a thread which is currently executing it using the synchorinized keywokd, and making other threads that wish to use it wait until the first thread is finished. These other threads are placed in the waiting state. Java is NOT AS FAIR as the service station because there is no queue for waiting threads. Any one of the waiting threads may get the monitor next, regardless of the order they asked for it. The only guarantee is that all threads will get to use the monitored code sooner or later.

Finally the answer to your question: the lock could be the key object or the service desk. None of which is a Thread.

However, these are the objects that currently decide whether the toilet is locked or open. These are the objects that are in a position to notify that the bath room is open (“notify”) or ask people to wait when it is locked wait.

If the threads were designed to give the lock to one another, then one thread might ‘chose’ a ‘friendly’ thread leading to nepotism.

Hence, the wait and notify methods have to be in Object class.

Threads borrow keys from JVM and return to JVM.

NOW what is a lock?

Locks are inbuilt, hidden objects in a class. For static synchorinized methods the class object has a lock and for non-staic classes the objects instances themselves are the lock.

That is why while blocking a piece of code (instead of an entire method) we use synchronized(this){ …;}

If the same thread has to access various piece of code which are mutually exclusive to modifications then simply create two objects and call them lock 1 and lock 2 and use these two locks to synchornize.

example. if within the same code there are two areas that have to be synchronized and if the two areas are mutually exclusive then we SHOULD NOT USE this keyword for locking.

In our analogy, if there is a condom vending machine in the toilet, then if an user who does not want to buy condoms is using the bathroom, then the same key will lock the toilet and the vending machine and if there is a person in the queue who wants to use the toilet only to get a condom, then he is unnecessarily locked. In this case the gas station has to use two rooms and two keys. That way using one will not affect the other.

Need clarifications let me know.

Most importantly wait, notify , notifyall come into picture only when synchonizing a piece of code.

**What is JDK and JVM ?**

Java Virtual Machine (JVM) is an abstract computing machine. Java Runtime Environment (JRE) is an implementation of the JVM. Java Development Kit (JDK) contains JRE along with various development tools like Java libraries, Java source compilers, Java debuggers, bundling and deployment tools.

JVM becomes an instance of JRE at runtime of a java program. It is widely known as a runtime interpreter. The Java virtual machine (JVM) is the cornerstone on top of which the Java technology is built upon. It is the component of the Java technology responsible for its hardware and platform independence. JVM largely helps in the abstraction of inner implementation from the programmers who make use of libraries for their programmes from JDK.



---------------------------------------------------------

-------------------------------------------------

**If you want print something in java without using static blocks or main method.**

class Google

{

static int i=m1();`

public static int m1()

{

System.out.println("Hi......");

return 10;

}

}

---------- Output ----------

Hi......

java.lang.NoSuchMethodError: main

Exception in thread "main"

------------------------------------------

**What is the difference between yielding and sleeping?**

When a task invokes its yield() method, it returns to the ready state. When a task invokes its sleep() method, it returns to the waiting state.

-------------------------------------------

**What is synchronization and why is it important?**

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

---------------------

**Can there be an abstract class with no abstract methods in it?**

**Can an Interface be final?**

**Can an Interface have an inner class?**

----------------------------------------------------------

**Difference between WebServer and Web Container?**

**WebServer**: A machine which can take in client request over the web(internet) and give back a static response back to the client (static HTML page, jar file etc)

Ex: Apache Web Server

**Web Container**: Servlets/JSPs have the capability to develop dynamic pages (pages whose content/layouts changes based on the request). Servlets/JSPs live within the Web Container (i.e. Web Container manages the Servlets/JSPs). The WebServer communicates with the Web Container to generate dynamic web pages.

Ex: Tomcat

------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------

**Polymorphism:**

Polymorphism provides one of the most useful programming techniques of the object-oriented paradigm. *Polymorphism*, means "many forms," is the ability to treat an object of any subclass of a base class as if it were an object of the base class. A base class has, therefore, many forms: the base class itself, and any of its subclasses.

If you need to write code that deals with a family of types, the code can ignore type-specific details and just interact with the base type of the family. Even though the code thinks it is sending messages to an object of the base class, the object's class could actually be the base class or any one of its subclasses. This makes your code easier for you to write and easier for others to understand. It also makes your code extensible, because other subclasses could be added later to the family of types, and objects of those new subclasses would also work with the existing code.

--------------------------------------------

**What does it mean that a method or field is “static”?** -

Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That’s how library methods like System.out.println() work. out is a static field in the java.lang.System class.

-----------------------------------------------------------------------------

**How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?**

String hostname = InetAddress.getByName("192.18.97.39").getHostName();

-------------------------------------------------------

**What happens if we get an exception in finally block.**

The code in the finally block is intended to be "cleanup" code, so we can handle this situation in the following way:

public int myExeptionMethod() throws IOException, NumberFormatException {

...

. . .

. . .

try {

-----------

---------

}

finally {

**try {**

**}**

catch (IOException e) {

}

}

}

What will be the result of the following code:

public class Tester {

public static void main(String[] args) {

System.out.print("1");

try {

System.out.print("2");

System.exit(0);

} finally {

System.out.print("3");

}

}

}

Ans : 12 : System.exit(0) will cause the program to exit and finally block will not be executed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Write a programme to print the reverse of a number.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**What is interface and its use?**

- Interface is similar to a class which may contain method’s signature only but not bodies and it is a formal set of methods and constants declarations that must be defined by the class that implements it. Interfaces are useful for:

a)Declaring methods that one or more classes are expected to implement.

b)Capturing similarities between unrelated classes without forcing a class relationship.

c)Determining an object’s programming interface without revealing the actual body of the class.

**What are the similarities/difference between an Abstract class and Interface**?  
Differences are as follows:

Interfaces provide a form of multiple inheritance. A class can extend only one other class.

Interfaces are limited to public methods and constants with no implementation. Abstract classes can have a partial implementation, protected parts, static methods, etc.

A Class may implement several interfaces. But in case of abstract class, a class may extend only one abstract class.

Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast.

Neither Abstract classes or Interface can be instantiated.

Abstract classes are very much useful when there is a some functionality across various classes. Interfaces are well suited for the classes which varies in functionality but with the same method signatures.

**Explain the Polymorphism principle**

Answer: The meaning of Polymorphism is something like one name many forms. Polymorphism enables one entity to be used as as general category for different types of actions. The specific action is determined by the exact nature of the situation. The concept of polymorphism can be explained as "one interface, multiple methods".

**Different forms of Polymorphism:**  
 Polymorphism exists in three distinct forms in Java:

Method overloading

Method overriding through inheritance

Method overriding through the Java interface.

**What is final, finalize() and finally?**- final : final keyword can be used for class, method and variables. A final class cannot be subclassed and it prevents other programmers from subclassing a secure class to invoke insecure methods. A final method can’t be overridden. A final variable can’t change from its initialized value. finalize() : finalize() method is used just before an object is destroyed and can be called just prior to garbage collection. finally : finally, a key word used in exception handling, creates a block of code that will be executed after a try/catch block has completed and before the code following the try/catch block. The finally block will execute whether or not an exception is thrown. For example, if a method opens a file upon exit, then you will not want the code that closes the file to be bypassed by the exception-handling mechanism. This finally keyword is designed to address this contingency.