# Final Look up:

|  |  |  |
| --- | --- | --- |
| ***SI NO*** | ***Topic*** | ***Status*** |
| 1 | SOAP WSDL components | Completed |
| 2 | Producer /Consumer, Sequential Thread, Reentrant Concept, Executor Frame Work | Completed |
| 3 | Collection Hierarchy Comparison |  |
| 4 | Serialization and Externalization. |  |
| 5. | Program to Print Fibonacci Numbers and also to print the sum of them. |  |
| 6. | Program to Print the permutation of a string and also to reverse it using recursion. |  |
| 7. | Hibernate, Mapping , Named Query |  |
| 8. | Springs IOC. |  |
| 9. | Spring MVC flow diagram. |  |
| 10. | Merge two arrays and sort them |  |
| 11. | Comparable VS Comparator , Callable VS Runnable, Procedure VS Function (SQL) |  |
| 12. | Load VS Save, Hash Set VS linked Hash Set |  |
| 13. | Bean Scopes |  |
| 14. | Concurrent Hash Map |  |
| 15. |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Links

1. ***Interview Programs :*** <http://www.geeksforgeeks.org/java/#Collection%20in%20Java>
2. ***Atomic Integer Program to print numbers in Squence :*** <http://www.tutorialsdesk.com/2016/01/3-threads-to-print-alternate-values-in.html>
3. ***Program to Print Odd and even number :***

<http://www.tutorialsdesk.com/search/label/Synchronization>

1. Externalayzation

http://www.jusfortechies.com/java/core-java/externalization.php

Importance of hash code and equals method:

<https://www.geeksforgeeks.org/equals-hashcode-methods-java/>

https://www.javaworld.com/article/2074996/hashcode-and-equals-method-in-java-object---a-pragmatic-concept.html

Executor Framework Overview:

<http://technicalstack.com/java-executor-framework-thread-pool/>

Company Based Question : <https://practice.geeksforgeeks.org/company-tags>

# Explore

## Design Pattern:

1. Factory Pattern.

2. Proxy.

3. Adapter

## Data Structure:

1. Brute Force Method.

## Data Base:

1. Second largest value.

2. Procedure vs functions.

3. Difference between Union and Union all.

4. Cluster Index

## Java:

1. Polymorphism and explain polymorphism behaviour in project.

2. Why do we have notify and wait method defined in Object Class.

3. Re-entrant Locks with example.

4. Generics.

5. OOPS Concept in terms of project.

6. Enumeration.

7. Explain Racing Condition.

8. Explain Thread Dead lock.

9. Memory model in Java 8.

10. Explain Memory Leak.

11. Methods of Collection interface.

12. Contract between equals and has code method.

13. If there is a collection in a class and if you trying to make the class immutable how would you do that?

14. Significance of Instance of operator.

15. Callable vs Runnable.

16. Difference in using int vs Integer.

17. Add short and integer through a method that can accept two integer.

18. Semaphore variable.

19. Blocking Queue and its advantages.

20. Composition VS Inheritance

21. Composition VS Aggregation.

22. Thread.sleep vs wait.

23. How do you handle SOAP Exceptions?

24. composition , aggregation and inheritance.

25. CountDownLatch and cyclic barrier.

26. Transient vs Volatile.

27. Enumeration.

## Springs

1. Difference between Application Context and Bean Factory.

2. Different ways of instantiating Application Context.

3. Bean Scopes.

## Hibernate:

1. Mappings with (Bi dierectional and Uni Directional Implementation.)

2. Load vs save.

3. ehcache Vs os cache.

4. Hibernate Session related Questions.

## Programs

1. Hash Map- with scrambled keys (ASCII Comparison as base evaluation)

a. Key 1 : “vgi”

b. Key 2 : “ivg” Both the keys are the same as they are just scrambled.

2. Reverse a string through recursion.

3. Design a singleton class.

4. Create a immutable class.

5. Producer Consumer Program.

6. Print sequnce number with two threads or three threads.

7. Print odd and even number.

8. write a program to find the number of occurrence of each character in a string using recursion.

9. if the super class implements Serializable and the subclass forgot extend Serializable can we serialize the sub class.

10. Implement a searching mechanism to retrieve the result with O (1) as performance.

11. Custom Hash Map.

12. Enhance hash map to store the values in the ascending order.

13. Program to create Custom exception.

14. Merge two arrays and sort them.

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

# Company

## NTT Data :

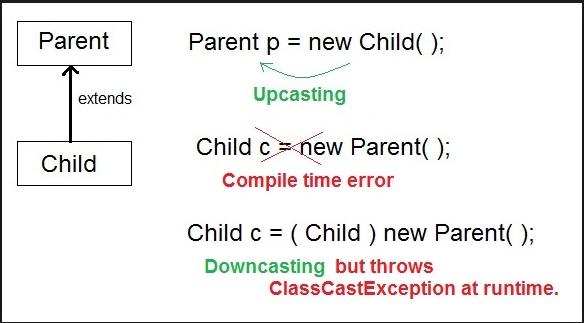
1. Write snippet of WSDL.
2. What are the components of WSDL.
3. Query to get second largest number.
4. Internal Implementation of Hash Map and program.
5. Difference between Abstract and Interface.
6. When will you use abstract and inheritance?
7. Singleton class in Project.
8. Factory Method in Project.
9. SPRING MVC flow diagram.
10. Stereo type annotations.
11. Explain Polymorphism in your project.
12. Why do we have notify and wait method defined in Object Class.
13. Explain Hibernate Mappings.
14. Eager and lazy fetch and when to use them.
15. Joins in SQL.
16. Re entrant locks.
17. Regular Expression for common patterns to match only alphabets and number.
18. Transaction management Configuration in your project.
19. Factory design pattern in your current project.

## MITEL:

### OOPS Concept in Project

* 1. **Inheritence** : The process of retrieving the basic profile information remains the same across the processing of all the API. Hence we have defined an abstract class that has the implementation to retrieve the data and the implementation for the rest of the methods are done specific the inheriting API class.
  2. **Abstraction Class** : Same as above.
  3. **Interface** is mainly used to hide the implementation of the class.
  4. **Polymorphism** – it can be described as a programming language’s ability to process objects of various classes with a single interface. There are basically two types of polymorphism; first one is Static polymorphism for example method over loading. The second type is called Dynamic polymorphism, for example method overriding.

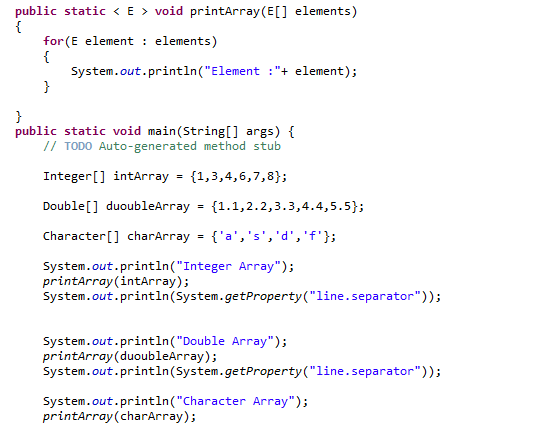
For Dynamic polymorphism, say there exists a IS-A relationship between two classes, if we create an object of sub class and assign the reference to a super class variable. The compiler determines the exact method to be invoked at the run time which is termed a runtime binding or dynamic polymorphism.



* 1. ***Encapsulation*** : It is a process of binding the data and the code acting on the date into a single unit which is a class, The main thing to be noted while performing encapsulation is the we must make the data members of a specific class as private there by restricting the access to all the members outside the current class and providing getters and setters to read and write the data.

### Generics – purpose and example

Employing this strategy in implementing functions would provide the advantage of using one function to manipulate a series of data types.



1. Hash Map- with scrambled keys (ASCII Comparison as base evaluation)
   1. Key 1 : “vgi”
   2. Key 2 : “ivg” Both the keys are the same as they are just scrambled.
2. Reverse a string through recursion.
3. Output of a program .
   1. Up Casting
   2. Down Casting
   3. String Compare .
   4. System out prtin (“”+14+14+””)
4. Hibernate Second level cache.
   1. Read Only
   2. Update.
5. Real time example of Serialization, externalization.
6. Enumeration.
7. Hibernate Mappings (important many to many).
8. Oracle Joins, triggers, Sequence package (What is the return type of package is it cursor or pointer).
9. Producer Consumer , Sequence number printing.
10. Significance of notify() and notifyAll().
11. Backend Adapter :
    1. How do we read or write message to a queue.
    2. List the respective API’s that does the JOB.
12. Notation.
13. Different ways to implement Thread Safety.
14. Join ()
15. Merge two arrays and sort them

## Accenture:

1. Memory Model in Java current version and new version.
2. Memory leak in Java.
3. Initiate two threads one for executing non static method and the other one for static method of the same object, do we need synchronization in this scenario and also explain the locks involved in the execution.
4. Explain Executor Frame work.
5. How do you parse data files of huge volume?
6. Joins in SQL.
7. Different ways of instantiating Application Context.
8. Difference between Application Context and Bean Factory.
9. Setter versus Constructor inject which would be the ideal choice and why.
10. How to create an immutable class. Considering reflection.
11. Methods of Collection Interface.
12. Why do we have wait and notify in the object class.
13. Improvements in Java 8 and 9.
14. Hash map and concurrent has map differences.
15. Read write problem.
16. Bean Scopes.
17. Explain Abstraction in terms of current project.
18. Different Segments of heap memory.
19. What is the contract between equals and hash code?
20. Re Entrant Locks.
21. Explain lock at class level and at object level.
22. Explain Collection hierarchy.
23. If there is a collection in a class and if you trying to make the class immutable how would you do that?. (Through some method of the collection class named as \*bound\*).
24. What happens if you start iteration on a collection and within the loop you add or remove some elements from the collection.
25. POM.xml file.
26. Instance of operator.
27. What is a view.
28. Callable vs Runnable.
29. Load vs Save in Hibernate.
30. If you are to adda new table to a hibernate application what are the steps.
31. How do you enable Hibernate Second level cache?
32. Procedure VS functions in SQL.

## Genpact:

1. Difference between Proxy and Adapter Design Pattern.
2. Difference between Union and Union all.
3. Final finally and finalize.
4. Wait, notify and notify all.
5. Group by.
6. What happens if you declare a variable and fail to initialize.

## L&T: 21st April

1. Proxy design pattern
2. Second highest salary - Done
3. Singleton using synchronized key word - Done
4. Computation of bucket index in hashmap.
5. Add short and integer through a method that can accept two integer arguments - Done
6. Semaphore variable
7. Steps to enable second level cache
8. Order by clause in sql
9. Write a program for proxy design pattern
10. And also for factory design pattern
11. Ehcache Vs os cache
12. In hibernate
13. Java 8 Concepts
14. Concurrent hash map
15. Blocking queue
16. And it's advantage
17. What is cluster index
18. Write a program to find the number of occurrence of each character in a string
19. Using recursion
20. Explain merge sort
21. Quick sort
22. Can marker interface have variables
23. Serialization and ssid
24. If the super class implements Serializable and the subclass forgot extend Serializable can we serialize the sub class
25. And the same question in other way
26. Functional interface
27. If you instantiate 100 thread through callable interface how would you monitor the outcome of the processing
28. Implement a searching mechanism to retrieve the result with O (1) as performance
29. write a logic to iterate and print the hash map
30. Can we have custom interface and generate lambda expression
31. Create a class implementing runnable class what happens if you call run method directly and then call start method.

## Cap Gemini: 28th April

## Wipro : 03th May

1. Composition VS Inheritance
2. Composition VS Aggregation.
3. MVC Architecture.
4. Thread.sleep vs wait
5. How do you handle SOAP Exceptions?
6. Hibernate Session related Questions.

## Century Link :

1. Create a sample application for loan processing Enterprise.
2. If there is a need to connect to different data base servers from hibernate, how will you achieve it.
3. Write custom hash map.
4. Enhance hash map to store the values in the ascending order.
5. Can you explain composition , aggregation and inheritance.
6. How do you implement data source in your project.
7. Explain out of memory exception and how to analyze it and steps to provide the fix.
8. Post serialization of any given object if the structure of the class is modified will the deserialization result in an error.
9. Can you write an example and explain proxy, adapter design pattern.

## HCL :

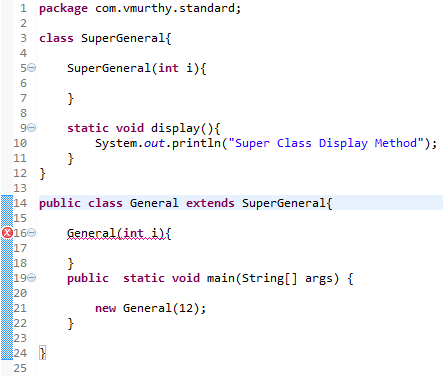
1. Usage of int VS Integer.
2. Spring Security.
3. CoutDownLatch and cyclic barrier.
4. Java 8 Memory model.
5. Optional in java(Java 8 Feature).
6. String class Method – intern.
7. Setter injection –
8. If the singleton bean has a prototype property by default it behaves as a singleton but how to retain the “prototype” behavior.

## Generic Questions :

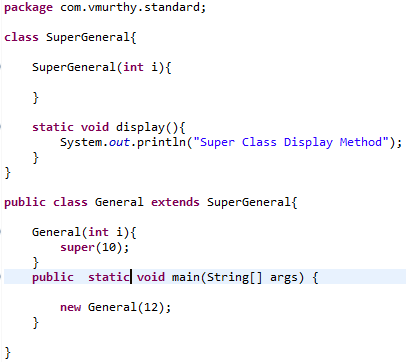
1. *What happens if we make the main method as non-Static* – No compilation error will be generated? But a runtime exception will be generated.
2. *Can we override static method* – We can declare two methods with the same signature in sub class but it is not considered as overriding as there is no run time polymorphism. So the overall answer is no.
3. *Is it Mandatory to have a default constructor* – Well it is not mandatory, if not constructors are mentioned the compiler itself creates a default no argument constructor, if a parameterized constructor exists, and then no default constructor will be created.

If there is a parameterized constructor, then we have to explicitly perform call to the parameterized super class constructor.

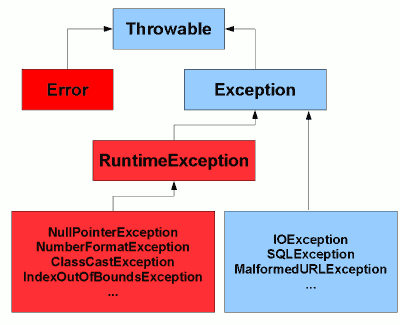
Example 1: Improper Constructor chain



Example 2: Constructor chain



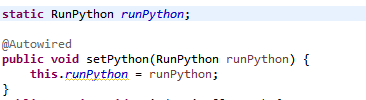
1. can an inner class access the member of the outer class.
2. Exception Hierarchy



## To Do :

1. Connection pool.
2. Atomic Variable concept.
3. Sorting Algorithm.
4. How to implement an Arraylist to filter only odd indexes effeciently without using JAVA 8 filters?
5. Views in Oracle.
6. Cursor in oracle.
7. Bulk and row id.
8. Merge Function in oracle.
9. Specific features of Oracle 11g.
10. Brute Force Method.
11. Java 8 Features : <https://www.journaldev.com/2389/java-8-features-with-examples#interface-default-static-method>
12. Custom Exception – The class should be able to throw multiple exceptions.
13. Merge two arrays and sort them.
14. Memory Model In JAVA 8
15. Usage of HashMap in BIS caching Mechanism.
16. Named Query’s In BIS.
17. Reflection real time example.
18. <http://www.technofunc.com/index.php/domain-knowledge/category/bfsi-domain-knowledge?f=1>
19. Static blocks.
20. One static and one non static block do we need synchronized keyword.
21. Interface having concrete method in Java 8.
22. In an many to many relation ship where will you define the inversion.
23. Save and merge in Hibernate.
24. Adapter and Proxy.
25. Union and union all.
26. Application context VS bean factory.
27. How to autowire static fields as the usual @Autowire on the field definition will not work.

Identify the reason in the change of behavior.

1. 
2. Determine the order of execution of the below mentioned entity.
   1. Static block
   2. Static members
   3. Static method

Note: what if the static member is used within the static block.

## CGI Wrap Up :

1. Check out BRS46.0 and BRS45.0 version of BOID controller to observe the changes in caching mechanism.
2. Producer / Consumer Problem.
3. Printing Sequence Number.
4. Serialization, Externalization.
5. Hibernate Caching.
6. <http://www.jusfortechies.com/java/core-java/externalization.php>
7. Basic Regular expression.
8. Thread to print odd and even number from an array and a third thread to print the sum od add and sum of even at the end.
9. Enumeration.
10. Five components of a WSDL.
11. Generics.
12. Design Pattern in Project.
13. Hibernate - second level cache.
14. JMS API’s used in Backend Adapter.
15. SNIPPET of WSDL.
16. Connection pool.
17. Reentrant lock.
18. Recursive function to reverse a String.
19. Transient and volatile.
20. Commonly asked difference between.
21. Joins in Oracle.
22. Fibonacci series.
23. Permutation of string.
24. Q. Write a method to print following:  
    Input:- 10.102.95.20  
    Output:- 20.95.102.10
25. Atomic Variable concept.
26. Sorting Algorithm.

## Important :

1. Bi directional and unit directional mapping in Hibernate (with mappings).
2. Mappings in Hibernate.
3. Why do we have main method in Spring Boot Application?
4. Program to print the following pattern based on the integer input

Pattern :

\*

\*.\*

\*.\*.\*

\*.\*.\*.\*

1. Consider an arry of integers, devise a look up logic that gives the two integer the product of which is lesser than or equal to the input value N.