1. Brief about yourself ,about past technical and domain experience, projects etc.
2. About my projects- follow up question mostly on Architectural point of view
3. Have experience on Linux? What is bashrc file? And its significant? How can you delete / open / search word or file in linux. Just idea.
4. What is need to move from OOP to functional language? [I mentioned we have used pure functional language platform as enhancement along with Java on middleware ]
5. About design patterns I have used and technical scenarios about implementation of those design patterns
6. SOLID principles and its examples

SRP > Working with databases

* 1. Open/Get a database connection ? New database
  2. Fetch the data from database ? JDBC. Tomorrow u adopt ORM

Open/Closed Principles –

Open for extension but closed for principle

If class behavior is used by multiple classes, then better to extend it to add new implementation rather than updating in the same class

Liskov’s Principle: parent class should be easily substituted by child class.

?

Interface Segregation Principle – help to reduce unused methods from interface design

Awt package – Listeners

Capturing mouse clicks implements ActionListener \

{

Click event

}

Separate interfaces based on functionality

Dependency Inversion Principle

* To make use of interface to achieve abstraction instead of concrete class.

Service class using RepositoryInterface as dependency

Using functional programming what advantages you have over OOP?

1. What role you think will be most suitable for you? And why?
2. What java 8 features you have used?

* Using lambda expressions
* Stream API - perform operations on data using functions
* Java.time package – LocalDate, LocalTime, LocalDateTime, Zone, Duration, Period, Instant

More functionalities on Date manipulation

Dates are immutable objects

* ParallelStream – list.stream().parallel() .filter().map().average();

.parallel() – split the work into multiple threads()

What is immutability in Java

State of an object cannot be modified after its creation.

Streams – store the data? No . (it is a pipeline where data flows and operations.

1. Explain SOLID design principle.

Builder pattern

1. What points you will check while doing code reviews?
2. can you serialize and deserialize singleton class?
3. explain benefit of composition over inheritance?

Loose-coupling

Higher flexibility – building business object out of various components rather than look out for common functionalities

1. JDBC connection
   1. using Prepared Statement
   2. using Callable Statement
2. Multithreading concepts

What are threads?

* Threads are sub-processes running within the application
* Threads are assigned with tasks
* Runnable interface and Thread class

void run(){..}

* Single Thread – main() {..}
* Threads are light weight – they have their stack own, but share the resources or data
* They share same address process of the main process
* Inter thread communication is low cost
* T1 (Write)
* File (write) Lock the resource
* T2 (write)

= Synchronization is a process where one thread accessing resource wll; be locked and other threads will wait till locks are released.

Synchronized method{

}

method(){

Synchronized (object){

}

Life cycle

* + - 1. Create the thread = new Thread(Runnable object)
      2. Start the thread – put the thread in the thread pool – Runnable
      3. Running the thread – cpu will execute the task assigned to the thread
      4. Waiting/sleeping/blocked state – sleep(), wait(), notify(), notifyAll()

Synchronization can lead to problems:

Deadlock condition – avoid

Starvation

Race Condition (data) – producer / consumer solution

1. Generics concepts

* Compile time Typesafety for collection aalow programmer to catch invalid types of elemnsts in collection at compile time
* <T> - List<T> = new ArrayList<>();
* ClassCastException exceptiin is avoided
* Create Generic class with generic methods
* List <? extends Number> - upper bounding
* List< super A> ex: List<? Super Integer> - u can pass Integer, Number, Object – lower bounding

1. Can we serialize or deserialize Singleton class?
2. why moved on java 8
3. Questions on Java 8 features
4. Why were used Lambda Expressions

Use of Lambda expressions – support functional programming

Lambda expression is the implementation of a functional interfaces

Functional interfaces – interface with only abstract method (SAM)

Interface Adder

{

Int add(int a , int b); // SAM

}

// write a lambda expression for it

Adder intf ;

Intf = () -> {};

Intf = (int a, int b) -> {

Syso(“in lambda”);

return a + b;

}

var x = function(){

} // javascript

x(); // eexecute

// invoke the implementation

// call the method

int sum = Intf.add(10, 20);

Interface

1. Java 8 for Collection. How it was better than earlier.

* Earlier we use to write all the code in imperative style
* Now we use stream api - sql -like operations through methods
* Readable and concise code

1. What is  Immutable class
2. If we write class as final, private constructor, is it Immutable ?[added questions on Date as variable]

final class MyClass{

int x ;

Date birthdate;

~~private~~ public constructor(data members)

}

Deep Copy – clone() method – where to perform the deep copy – inside getter method

String class is immutable

String str = new String(“john”);

1. Concept of Defensive Copy
2. What is final keyword?
3. What are different types of collections in Java?
4. What is difference between interfaces and abstract classes?
5. Does Java allow multiple inheritance for interfaces and abstract classes?
6. Can we have non-static main method in class where static main method is already present?
7. What is static keyword?
8. What is final keyword?
9. What changes I have to do, if I don't want a class to be inherited by others?
10. What are overriding and overloading in java?
11. Thread.start() vs Thread.run()

Thread.start() -> thread can begin execution

Thread.run() -> JVM invokes the run method for threads in runnable state

1. How to create Singleton class?
2. List vs Map
3. How Hashset internally works?
4. In your last project did your client request some changes in application after deployment, if yes then how did you handle that?
5. Working of HashMap and HashSet
6. Java 8 features
7. Difference between Abstract class and Interface
8. Difference between List and Set
9. Final keyword usage in Java
10. Hashset Internal working
11. Multiple Inheritance in java
12. Singleton Design Pattern
13. Best Practices while writing Java code.
14. What are the new features in Java 8?
15. What changes were done regarding memory in Java 8? // tomorrow
16. What would you do to improve the quality of the product?
17. Explain Factory Design Pattern in detail
18. How Hashmap works internally?
19. What is difference between Hashmap and Hashset?

equals() in Object boolean equals(Object o)

int hashCode() – unique identifier

Contract between equals and hascode

1. IF objects are equal, then they should have same hash code
2. If two objects have same hashcode, they may or may be not equal

x.equals(null) ->

Set II - Java Questions

1. What is Abstraction in OOPs? Any example in Java API
2. Can we Override static methods?
3. Can we override private methods?
4. What are different types of Classloaders in Java?

Delegation – Appl class

Uniqueness

Visibility -

1. How to create instance of the class at runtime?
2. Difference between ClassNotFoundException and NoClassDefFoundError.
3. What is **transient** keyword used for?

During Serialization it will store default value for those property.

1. Explain **volatile** keyword.
2. Can we call static method from Object? If yes, then what if object is null
3. Can we create an object of the class in same class?
4. Explain **autoboxing** concepts
5. Internal working of Hashmap.
6. Can we use Object as key in Hashmap. If yes, then what extra care we need to take?
7. What happens if hashcode method returns same value?
8. What is the contract between equals and hashcode method?
9. Can I write Integer i = 2 ? If I need to compare the value in if statement, Should I compare it using i==2 or i.equals(2) ?
10. Design Pattern – Atleast 3 design patterns ( other than Singleton and Factory)
11. How to find the 2nd largest number from Array of integers
12. Which sorting algorithm is used in Array.sort()?
13. What is Functional Interface in Java 8?
14. What are default methods and Static methods in Java 8? - default() and static() – when to add these in functional interface

Comparator interface – Java 8

1. Using functional programming what advantages you have over OOP?

-------------------------------------Programmatical Questions----------------------------------------

1. Can we save this class as B.java

No. we cannot save it as class is not public

class A

{

                private String message;

public void m1() {

}

}

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

1. What will be the output (Which class method will get called?)

class A

{

               public void m1() {

}

public void m2() {

}

}

class B extends A

{

               public void m1() {

}

public void m3() {

}

public static void main(String[] args)

  {

A a = new B();

a.m1(); // child class m1() method

                a.m2(); // parent class implementation

                a.m3(); // compilation error

  }

}

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

**What is need to move from OOP to functional language? [I mentioned we have used pure?**

**Using functional programming what advantages you have over OOP?**

**Need to move functional programming?**

* **You can write concise and readable code**
* **Reduces Code Complexity**

**Object-oriented – Object contains data whose state can changed**

**Java support imperative style – What to do & how to do**

**Collection -> filter operation -> roles -> managers**

**Declarative style code – What to do .contains()**

**Pure functional – Result should be same for any value of same type (n) -> (n \* 2)**

**2 -> square of 2**

**Int x = 3;**

**N = 2, 3,4, 5, 6 -> 2, 3, 4, 10, 18…**

**Function(n)**

**{**

**N \* n \* x;**

**}**

**Access Final variables**