Advise to brush over basic core java interview questions (Collections, Threading etc ), JEE related( Design pattens, Web Services , Servlet, etc) and database skills.

 Books

Programming Interviews Exposed: Secrets to Landing Your Next Job

Cracking the Coding Interview: 150 Programming Questions and Solutions

Java/J2EE Job Interview Companion by Arulkumaran Kumaraswamipillai

Elements of Programming Interviews: 300 Questions and Solutions

Read more: <http://javarevisited.blogspot.com/2011/04/top-20-core-java-interview-questions.html#ixzz3Cet8xYhe>

:  
  
1. Why finance?  
  
2. What do you see as your biggest strength?  
  
3. What are three leadership qualities that you possess that you are proud of?  
  
4. Why do you want to work in services?  
  
5. Why Goldman Sachs?  
  
6. What is your weakness? - Perfectionist  
  
7. When have you been in a leadership position?  
  
8. When have you been in a leadership position and failed?  
  
9. When have you been in a group environment and not been the leader?  
  
10.Tell me about your previous work experience.

Given an nxn matrix of numbers in ascending order in both dimensions how would you go about finding if the number y is in the matrix.

Design a system to manage a parking lot

Java questions from the very basics (talking about the collections framework etc.) to multi-threading, to some details about how hash tables work, to Java garbage collection, etc. Database questions about how indexing works, the data structures used in indexing, basic SQL questions

mainly multi-threading, database transaction theory, and agile vs. waterfall methodologies

Explain how Java's garbage collection works from a high level. What different algorithms are there?

Sketch out a simple implementation for a hash table. What properties does the hash function need to have?

Explain the best practices in java

Given an integer, return all sequences of numbers that sum to it. (Example: 3 -> (1, 2), (2, 1), (1, 1, 1)).

This problem boils down to the subset sum problem . Make an list(1,1,1,2,2,2,3,3,3,1,2,3) . Fins a subset in this list which equals the given number. Handle duplicates gracefully. For each list found, convert it to a string and add it to hashmap to check for duplicates

Fibonacci implementation

how to find square root of integer

JPA implementations other than hibernate

When to use Interface and abstract class and Inheritance

what is final finally finalize  
what is static  
what is nested class  
OOPS Principles

How would you tackle a very sever production issue on your own

More questions on Core java, SQL and touch upon JSP

How do you find 2 missing elements in an array of consecutive integers that are not sorted who's size is N-2.

How do you calculate weighted averages?

then he asked me about execution of storage procedure.

He was asking me of threads, marker interfaces, serialization, synchronization, Timers, deadlocks and that too basic questions.  
Also he asked for merge concept of merge sort.

He asked me about quick sort and also randomization in it without he having any idea of basic concepts of randomization. I said that I would pick a random element as pivot and swap it with the last element in the array and perform the partitioning based on it. He was saying that I may then disturb the randomness of it. I said that as the input data distribution is not known we will not be effecting the randomness and hence we can only find expectation on the complexity in randomized algorithms. He didn't understand iota of what I was saying and hence dissatisfied.

Write a program to reverse a string

pass-by-reference  
pass-by-value

**Databases:**

Joins  
Index  
Procedures  
Functions  
Optimization  
Normalization  
Queries

[Why multiple inheritance is not supported in Java](http://javarevisited.blogspot.sg/2011/07/why-multiple-inheritances-are-not.html) 

Top 20 Core Java Interview Questions and Answers asked on Investment Banks  
  
**Why Java doesn't support multiple inheritance**

1) First reason is **ambiguity around Diamond problem**, consider a class A has foo() method and then B and C derived from A and has there own foo() implementation and now class D derive from B and C using multiple [inheritance](http://javarevisited.blogspot.com/2012/10/what-is-inheritance-in-java-and-oops-programming.html) and if we refer just foo() compiler will not be able to decide which foo() it should invoke. This is also called Diamond problem because structure on this inheritance scenario is similar to 4 edge diamond, see below

           A foo()

           / \

          /   \

   foo() B     C foo()

          \   /

           \ /

            D

           foo()

In my opinion even if we remove the top head of diamond class A and allow multiple inheritances we will see this problem of ambiguity.  
  
Some times if you give this reason to interviewer he asks if C++ can support *multiple inheritance* than why not Java. hmmmmm in that case I would try to explain him the second reason which I have given below that its not because of technical difficulty but more to maintainable and clearer design was driving factor though this can only be confirmed by any of java designer and we can just speculate. [Wikipedia link](http://en.wikipedia.org/wiki/Diamond_problem) has some good explanation on how different language address problem arises due to diamond problem while using multiple inheritances.  
  
2) Second and more convincing reason to me is that **multiple inheritances does complicate the design and creates problem during casting, constructor chaining etc** and given that there are not many scenario on which you need multiple inheritance its wise decision to omit it for the sake of simplicity. Also java avoids this ambiguity by supporting single inheritance with interfaces. Since interface only have method declaration and doesn't provide any implementation there will only be just one implementation of specific method hence there would not be any ambiguity.  
  
Read more: <http://javarevisited.blogspot.com/2011/07/why-multiple-inheritances-are-not.html#ixzz3CdvCaLsu>

How to create Immutable Class and Object in Java  
  
Read more: <http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html#ixzz3CdvpY0FO>

1. State of immutable object can not be modified after construction, any modification should result in new immutable object.

2. All fields of Immutable class should be final.

3. Object must be properly constructed i.e. object reference must not leak during construction process.

4. Object should be final in order to restrict sub-class for altering immutability of parent class.

1) All the fields must be private and preferably final  
2) Ensure the class cannot be overridden - make the class final, or use static factories and keep constructors private  
3) Fields must be populated from the Constructor/Factory  
4) Don't provide any setters for the fields  
5) Watch out for collections. Use Collections.unmodifiable\*. Also, collections should contain only immutable Objects  
6) All the getters must provide immutable objects or use defensive copying  
7) Don't provide any methods that change the internal state of the Object  
  
Read more: <http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html#ixzz3Ce1cLtNy>

By the way, you can still create immutable object by violating few rules, like String has its [hashcode](http://javarevisited.blogspot.com/2011/10/override-hashcode-in-java-example.html) in non final field, but its always guaranteed to be same. No matter how many times you calculate it, because it’s calculated from final fields, which is guaranteed to be same. This required a deep knowledge of Java memory model, and can create subtle [race conditions](http://javarevisited.blogspot.com/2012/02/what-is-race-condition-in.html) if not addressed properly. In next section we will see simple example of writing immutable class in Java. By the way, if your Immutable class has lots of optional and mandatory fields, then you can also use [Builder design pattern](http://javarevisited.blogspot.com/2012/06/builder-design-pattern-in-java-example.html) to make a class Immutable in Java.

public final class Contacts {

    private final String name;

    private final String mobile;

    public Contacts(String name, String mobile) {

        this.name = name;

        this.mobile = mobile;

    }

    public String getName(){

        return name;

    }

    public String getMobile(){

        return mobile;

    }

}

public final class ImmutableReminder{

    private final Date remindingDate;

    public ImmutableReminder (Date remindingDate) {

        if(remindingDate.getTime() < System.currentTimeMillis()){

            throw new IllegalArgumentException("Can not set reminder” +

                        “ for past time: " + remindingDate);

        }

        this.remindingDate = new Date(remindingDate.getTime());

    }

    public Date getRemindingDate() {

        return (Date) remindingDate.clone();

    }

}

Read more: <http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html#ixzz3CdxJllFM>