Collections

Q1. What are limitations of object Arrays?

The main limitations of Object arrays are

- These are fixed in size ie once we created an array object there is no chance of increasing or decreasing size based on our requirement. Hence If we don't know size in advance , arrays are not recommended to use
- Arrays can hold only homogeneous elements.
- There is no underlying data structure for arrays and hence no readymade method support for arrays. Hence for every requirement programmer has to code explicitly

To over come these problems collections are recommended to use

Q2. What are differences between arrays and collections?

Arrays	**Collections**
1. Arrays r fixed in size and hence once we created an array we are not allowed to increase or decrease the size based on our requirement.	1. Collections are growable in nature and hence based on our requirement we can increase or decrease the size.
2. Memory point of view arrays are not recommended to use	2. Memory point of view collections are recommended to use.
3. Performance point of view arrays are recommended to use	3. Performance point of view collections are not recommended to use.
4. Arrays can hold only homogeneous elements	4. Collections can hold both homogeneous and heterogeneous elements.
5. Arrays can hold both primitives as well as objects	5. Collections can hold only objects.
6. For any requirement, there is no ready method support compulsory programmer has to code explicitly.	6. For every requirement ready made method support is available. Being a programmer we have to know how to use those methods and we are not responsible to implement those.

Q3. what are differences between arrays and ArrayList?

Refer the answer of Q2

Q4. What are differences between arrays and Vector?

Refer the answer of Q2

Q5. What is Collection API ?

It defines set of classes and interfaces which can be used for representing a group of objects as single entity

Q6. What is Collection framework?

It defines set of classes and inter faces which can be used for representing a group of objects as single entity

Q7. What is difference between Collections and Collection?

Collection is an interface which can be used for representing a group of individual objects as single entity and it acts as root interface of collection frame work.

Collections is an utility class to define several utility methods for Collection implemented class objects.

Q8. Explain about Collection interface?

- This interface can be used to represent a group of objects as a single entity.
- It acts as root interface for entire collection framework.
- It defines the most commonly used methods which can be applicable for any collection implemented class object

Q9. Explain about List interface?

List interface is a child interface of Collection interface. This can be used to represent group of individual objects in as a single entity where

- Duplicates are allowed
- Insertion order is preserved

Q10. Explain about Set interface?

Set is a child interface of Collection interface. it can be used to represent a group of individual objects as a single entity where

- Duplicate objects are not allowed.
- Insertion order is not preserved

Q11. Explain about SortedSet interface?

it **is child** interface of Set interface. it can be used to represent a group of individual objects in to a single entity where

- All the objects are arranged in some sorting order (Can be natural sorting order or customizede).
- Duplicates are not allowed.

Q12. Explain about NavigableSet ?

It is child interface of SortedSet and provides several utility methods for navigation purposes

- It doesn't allows duplicates
- Insertion order is preserved
- It is introduced in 1.6 version

Q13. Explain about Queue interface?

If we want to represent a group of individual objects prior to processing, then we should go for Queue interface. It is child interface of Collection interface.

It has introduced in 1.5 version.

Q14. Explain about Map interface?

Remember it is not a child Interface of Collection Interface and hence Map and Collection Interfaces doesn't have any relationship.

- It can be used for representing a group of Objects as key, value pairs.
- Both keys and values should be objects
- Keys can t be duplicated but values can be duplicated.
- it has introduced in 1.2 version

Q15. Explain about SortedMap ?

- If we want to represent a group of objects as key value pairs where all the entries are arranged according some sorting order of keys then we should go for SortedMap.
- It is child interface of Map.
- It has introduced in 1.2 version

Q16. Explain about NavigableMap?

- It is child interface of SortedMap and defines several method for navigation purpose
- It is introduced in 1.6 version

Q17. Explain about ArrayList class?

ArrayList is a Collection which can be used to represent a group of objects as a single entity.

- it is a implemented class for List interface
- Introduced in 1.2 version
- The underlying data structure is resizable or growable array.
- Insertion order is preserved
- Duplicates are allowed
- Heterogeneous objects are allowed
- null insertion is possible
- This class implements RandomAccess , Serializable , Cloneable interfaces
- Best choice for retrieval purpose and worst if our frequent operation is insertion or deletion in the middle

Q18. What is RandomAccess Interface?

- If a collection class implements RandomAccess interface then we can access any of its element with the same speed.
- RandomAccess interface is marker interface and it dosent contains any methods.
- ArrayList and vector classes implements this interface.

Q19. Explain about LinkedList class?

LinkedList is a Collection implemented class which can be used for representing a group of objects as a single entity.

- LinkedList is the implemetation class for List interface
- Introduced in 1.2 version
- Underlying data Structure is   DoubleLinkedList
- Allows duplicates
- Insertion order is preserved
- Allows heterogeneous objects
- null insertion is possible
- LinkedList class implements Seriallizable and Cloneable interface but not RandomAccess interface
- Best choice if frequent operation is insertion or deletion an objects in middle but worst choice if frequent operation is retrieval.

Q20. Explain about Vector class?

Vector is a legacy collection class which can be used to represent a group of objects.

- Introduced in 1.0 version. it is legacy class
- The underlying data structure is resizable or growable array.
- Insertion order is preserved
- Duplicates are allowed
- Heterogeneous objects are allowed
- It is a implemented class for List interface
- null insertion is possible
- Vector class implements RandomAccess ,Serializable,Cloneable interfaces

- Best Choice if frequent operation is retrieval and worst choice if frequent operation is insertion or deletion in the middle.
- All methods present in Vector class are synchronized hence Vector class object is thread safe.

Q21. What is difference between ArrayList and Vector?

ArrayList	Vector
1. No method is synchronized in the ArrayList class	1. All methods in Vector are synchronized.
2. ArrayList object is not thread safe.	2. Vector is thread safe.
3. Relatively performance is high	3. Relatively performance is low
4. Introduced in 1.2 version and it is non legacy	4. Introduced in 1.0 version and it is legacy

Q22. How we can get synchronized version of ArrayList?

Collections class contains synchronizedList() method for this

Public static List synchronizedList(List l)

EX
ArrayList l= new ArrayList();
List l2=Collections.synchronizedList(l);

Similarly we can get synchronized versions of Set and Map objects by the following methods.

Public static List synchronizedSet(Set s)
Public static List synchronizedMap(Map m)

Q23. What is difference between size and capacity of a Collection Object?

size means number of objects present where as capacity means no of objects it can accommodate.

Q24. What is difference between ArrayList and Linked List?

ArrayList	LinkedList
1. The underlying data structure is resizable or growable array.	1. The underlying data structure is Double Linked List.
2. This is Best choice if frequent operation is retrieval and worst choice if frequent operation is insertion or deletion in the	2. This is Best choice if frequent operation is insertion or deletion in the middle and worst choice if frequent operation is retrieval .

middle.	
3. This class implements Serializable , Cloneable and RandomAccess interfaces.	3. This class implements Serializable , Cloneable but not RandomAccess interface.

Q25. What are legacy classes and interfaces present in Collections framework ?

- Enumeration ---Interface
- Dictonary ------Abstract class
- Hashtable -----Concrete class
- Properties -----Concrete class
- Vector -----Concrete class
- Stack -----Concrete class

Q26. what is difference Enumeration and Iterator?

Enumeration	Iterator
1. It is legacy interface and introduced in 1.0 version	1 It is non-legacy and introduced in 1.2 version
2Applicable only for legacy classes and it is not universal cursor	2Applicable for any Collection implemented class object.
3While iterating the elements we are not allowed to remove the objects just we can perform only read operation	3While iterating we can perform removal also in addition to read operation.
4By using elements() method we can get Enumeration object	4. By using iterator() method we can get Iterator object

Q27. What are limitations of Enumeration?

- While iterating the elements we are not allowed to perform removal operation
- It is applicable only for legacy classes and it is not a universal cursor.
- It can retrieve the elements only in forward direction

Q28. What is difference between enum and Enumeration?

An **enum** can be used to define a group of named constants .It has introduced in 1.5 version

Ex
```
Class Beer{
        KO,KF,RC,FO
}
```

Enumeration is cursor to retrieve Objects one by one from Collection objects.

Q29. What is difference between Iterator and ListIterator?

- ListIterator is the child interface of the Iterator
- Iterator is the single direction cursor where as ListIterator is bidirectional cursor.
- While iterating the elements by Iterator we can perform only read and remove operations. But by using ListIterator we can perform read,removal, replace and addition of new objects also.
- Iterator is applicable for every Collecton implemented class object but ListIterator is applicable only for List implemented class objects.
- Iterator can be get by using iterator() of Collection interface where as ListIterator can be get by using listIterator() method of List interface
- both are introduced in 1.2 version

Q30. What is relation between ListIterator and Iterator?

ListIterator is child interface of Iterator

Q31. Explain about HashSet class?

- The underlying data structure is Hashtable
- null values are accepted
- duplicates are not allowed
- insertion order is based on hashcode of the object hence insertion order is not preserved
- best suitable if frequent operation is search operations
- HashSet class implements Serializable and Cloneable
- it is implementation class for Set interface
- heterogeneous objects are allowed
- it is introduced in 1.2 version

Q32. If we are trying to insert duplicate values in Set what will happen?

If we are trying to insert duplicate objects to the HashSet , we wont get any compile time or run time errors just the add(Object o) returns false and it doesn’t add that object.

Q33. What is LinkedHashSet?
It is the child class of HashSet. The main difference between HashSet and LinkedHashSet is:

In the case of HashSet insertion order is not preserved , but in the case of LinkedHashSet insertion will be preserved.

Q34. Differences between HashSet and LinkedHashSet?

HashSet	**LinkedHashSet**

1The Underlying datastructure is Hashtable	1The underlying datastructure is combination of LinkedList and Hashtable
2Insertion Order is not preserved	2 Insertion order is preserved.
3Introduced in 1.2 version	3 Introduced in 1.4 version

Q35. What are major enhancements in 1.4 version of collection frame work?

LinkedHashSet
LinkedHashMap
IdentityHashMap

Q36. Explain about TreeSet?

It is Collection object which can be used to represent a group of objects according to some sorting order.

- The underlying datastructure is Balanced tree
- Duplicates are not allowed
- All objects are stored according to some sorting order hence insertion order is not preserved
- Heterogeneous objects are not allowed violation leads to ClassCastException
- For an Empty TreeSet as firs element null value can be inserted but after inserting that first value if we are trying to insert any other objects then we will get NullPointerException
- For an non empty TreeSet if we are trying to inser null value at run time u will get NullPointerException

Q37. What are differences between List and Set interfaces?

List	Set
1Insertion Order is preserved	1Insertion Order is not preserved
2Duplicate Objects are allowed	2 Duplicate Objects are not allowed
3The implemented classes are ArrayList,LinkedList , Vector and Stack classes	3 The implemented classes are HashSet, LinkedHashSet and Tree

Q38. What is Comparable interface?

- This interface can be used for defining natural sorting order of the objects.
- It is present in java.lang package
- It contains a method public **int compareTo(Object obj1)**

Q39. What is Comparator interface?

- This interface can be used for implementing customized sorting order.
- It is present in java.util package
- It contains two methods
    - public int **compare**(Object ,Object)
    - public boolean **equals**(Object)

Q40. What are differences between Comparable and Comparator?

Comparable	**Comparator**
1This can be used for natural sorting order	1This can be used for implementing customized sorting
2This interface present in java.lang package	2 This is present in java.util package
3Contains only one method: public int compareTo(Object obj1)	3 It contains two methods. public int compare(Object ,Object) public Boolean equals(Object)
4 It is marker interface	4 It is not a marker interface.

Q41. What is difference between HashSet and TreeSet?

HashSet	**TreeSet**
1The underlying data structure is Hashtable	1The underlying data structure is balanced tree
2Heterogeneous objects are allowed	2 Heterogeneous objects are not allowed bydefalut
3Insertion order is not preserved and it is based on hashcode of the objects	3 Insertion order is not preserved and all the objects are inserted according to some sorting order.
4null insertion is possible	4 As the first element only null insertion is possible and in all other cases we will get NullPointerException

Q42. What is Entry interface?

It is inner interface of Map.
In the Map each key value pair is considered as Entry object.

```
interface Map{
            //more code here
            interface Entry{
                        Object  getKey()
                        Object  getValue()
```

```
                    Object  setValue(Object new)
          }
}
```

Q43. Explain about HashMap?
It is a Map Object which can be used used to represent a group of objects as key-value pairs.

- The underlying data structure is Hashtable
- Duplicaes keys are not allowed duplicate values are allowed
- Insertion order is not preserved because insertion is based on hashcode of keys.
- Heterogeneous objects are allowed for both keys and values
- null key is allowed only once
- null values are allowed multiple times
- Introduced in 1.2 version

Q44. Explain about LinkedHashMap?

It is child class of HashMap. It is exactly same as HashMap except the following difference.

In the case of HashMap the insertion order is not preserved but in the case of LinkedHashMap insertion order is preserved. Introduced in 1.4 version

Q45. Differences between HashMap and LinkedHashMap ?

HashMap	**LinkedHashMap**
1.The underlying data structure is Hashtable	1.The underlying data structure is a combination of Hashtable and linkedlist
2.Insertion order is not preserved and it is based on hashcode of keys	2 Insertion order is preserved
3.Introduced in 1.2 version	3 Introduced in 1.4 version.

Q46. Differences between HashMap and Hashtable?

HashMap	**Hashtable**
1.The underlying data structure is Hashtable	1.The underlying data structure of Hashtable
2.No method is synchronized and hence HashMap object is not thread safe	2 .All methods are synchronized and hence it is thread safe
3.Performance is high	3. Performance is low
4.null insertion is possible for both keys and values	4. null insertion is not possible for both key and value violation leads to NullPointerException
5.Introduced in 1.2 version and it is non legacy	5. Introduced in 1.0 version and it is legacy

Q47. What is IdentityHashMap?

It is exactly same as HashMap except the following difference.

In the HashMap JVM uses equals() method to identify duplicate keys but in the case of IdentityHashMap JVM uses == operator for this.

Q48. What is difference between HashMap and IdentityHashMap?

Refer Q47 for the answer.

Q49. What is WeakHashMap?

It is exactly same as HashMap except the following difference.

In case of HashMap an Object is not eligible for garbage collection if it is associated with HashMap even though it dosent have any external references. ie HashMap dominates garbage collector.

But in case of WeakHashMap , if an Object is not having any external references then it is always eligible for garabage collectoion even though it is associated with weakHashMap. ie garbage collector dominates WeakHashMap

Q50. What is difference between HashMap and WeakHashMap?
Refer Q49 for the answer.

Q51. What is TreeMap?

TreeMap can be used to store a group of objects as key-value pairs where all the entries are arranged according to some sorting order of keys.

- The underlying data structure is RED-BLACK Tree
- Duplicates keys are not allowed but values can be duplicated.
- Insertion order is not preserved because insertion is based on some sorting order
- If we are depending on Natural sorting order then keys should be homogeneous(violation leads to ClassCastException) but values need not homogeneous
- In case of customized sorting order we can insert heterogeneous keys and values
- For empty TreeMap as first entry with null values are allowed but after inserting that entry if we are trying to insert any other entry we will get NullPointerException
- For non empty TreeMap if we are trying to insert null keys we will get NullPointerException
- There are no restrictions for null values.

Q52. What is Hashtable

Hashtable is a legacy Map and can be used to store objects as key value pairs.

- The underlying data sturucture is Hashtabe
- Duplicates keys are not allowed but duplicate values are allowed
- null insertion is not possible for both keys and values
- all methods are synchronized
- insertion order is not preserved because it is based on hashcode of keys
- heterogeneous Objects are allowed for both keys and values
- introduced in 1.0 version it is legacy class

Q53. What is PriorityQueue?

It represents a data structure to hold group of individual objects prior to processing based on some priority .it can be natural sorting order and it can be customized sorting order described by Comparator.
It is the implementation class of Queue interface.

- Insertion order is not preserved because here insertion is done based on some sorting order
- Duplicates are not allowed
- null insertion is not possible even as first element also
- If we are depending on natural sorting order Objects should be homogeneous violation leads to ClassCastException
- If we are depending on customized sorting order Objects can be heterogeneous also.

Q54. What is Arrays class?

- It is utility class for arrays.
- It defines several utility methods for arrays like sorting an array or searching an element in array
- present in java.util package

Q55. We are planning to do an indexed search in a list of objects. Which of the two Java collections should you use: ArrayList or LinkedList?
ArrayList
Q56. Why ArrayList is faster than Vector?

All methods present in the Vector are synchronized and hence any method can be executed by only one thread at a time. It slows down the execution.

But in ArrayList, no method is synchronized and hence multiple thread are allowed execute simultaneously which speed up the execution.

Exception

Q1.What is an Exception?
Ans.An unwanted, unexpected event that disturbs normal flow of the program is called Exception.Example: FileNotFondException.

Q2.What is the purpose of Exception Handling?
Ans.The main purpose of Exception Handling is for graceful termination of the program.

Q3.What is the meaning of Exception Handling?
Ans. Exception Handling doesn’t mean repairing an Exception, we have to define alternative way to continue rest of the code normally.
Example: If our programming requirement is to read the data from the file locating at London but at Runtime if London file is not available then we have to use local file alternatively to continue rest of program normally. This is nothing but Exception Handling.

Q4.Explain Default Exception Handling Mechanism in java?
Ans.If an exception raised, the method in which it’s raised is responsible for the creation of Exceptions object by including the following information:

- Name of the Exception
- Description of the Exception
- Stack Trace
- After creating Exception object the method handover it to the JVM.
- JVM checks for Exception Handling code in that method.
- If the method doesn’t contain any Exception handling code then JVM terminates the method abnormally and removes the corresponding entry from the stack.
- JVM identify the caller method and checks for Exception Handling code in that method. If the caller doesn’t contain any exception handling code then JVM terminates that method abnormally and removes the corresponding entry from the stack.
- This process will be continue until main() method.
- If the main() method also doesn’t contain exception handling code the JVM terminates that main() method and removes the corresponding entry from the stack.
- Just before terminating the program abnormally JVM handovers the responsibility of exception handling to the Default Exception Handler which is the component of JVM.
- Default Exception Handler just print exception information to the consol in the following format

Name of Exception: Description
Stack Trace (Location of the Exception)

Q5.What is the purpose of try?
Ans We should maintain all risky code inside the try block.
Q6. What is the purpose of catch block?
Ans.We have to maintain all Exception Handling code inside the catch block.
Q7. Is try with multiple catch block is possible?
Ans. The way of handling an exception is varied from exception to exception compulsory we

have to write a separate catch block for every exception. Hence try will multiple catch block is possible and it is recommended to use.

```
Example:
try{
        //Risky code
}
catch(IOException e)
{
        //Hndling code for IOException
}
catch(ArithmeticException e)
{
        //handling code for AE
}
catch(NullPointerExcetpion e)
{
        // handling code for NPE
}
catch(Exception e)
{
        //default exception handling code
}
```

Q8. If try with multiple catch block present is order of catch blocks important in which order we have to take?

Ans. If try with multiple catch block present then the order of catch block is very important it should be from child to parent but not from parent to child.

Q9. What are various methods to print Exception information? and differentiate them.

Ans.

Throwable class defines the following method to print exception or error information .

1. printStackTrace() :- This method print exception information in the following format.

Name of the Exception: Description
StackTrace

2.toString():- This method print exception information in the following format.

Name of the Exception: Description

3.getMessage():- This method prints only description of the exception.

Description

Q10.If an exception rised inside catch block then what will happen?
Ans. If an exception raised inside catch block and it is not part of any try block then it is always abnormal termination.

Q11. Is it possible to take try, catch inside try block?
Ans. Yes, It is possible to take try, catch inside try block. That is nesting of try catch is possible.

Q12.Is it possible to take try, catch inside catch block?
Ans. Yes, It is possible to take try, catch inside catch block.

Q13. Is it possible to take try without catch?
Ans. Yes, it is possible to take try without catch but compulsory finally block should be available.

Q14. What is the purpose of finally block?
Ans. The main purpose of finally block is, to maintain the cleanup code. This block will execute always.

Q15. Is finally block will be execute always?
Ans. Yes finally block will be executed always irrespective of whether exception raised or not raised whether exceptions are handled or not handle. There is one situation where the finally block won't be executed if the JVM is going to be shutdown.

Q16. In which situation finally block will not executed?
Ans. There is one situation where the finally block won't be executed if we are using system.exit(0) explicitly then JVM itself will be shutdown and there is no chance of executing finally block.

Q17. If return statement present inside try is finally block will be executed?
Ans. Yes, if return statement present inside try, then also finally block will be executed. finally block will dominate return statement also.

Q18. What is the difference between final, finally and finalize()?
Ans. **final:-** final is a modifier applicable for **variables, methods** and **classes**. **final variable** means constant and reassignment is not possible. **final method** means implementation is final in the child classes we can't override**. final class**means it won't participate in inheritance and child class creation is not possible.
finally:- It is a **block** associated with try catch to **maintain cleanup code**. Finally block will be executed always irrespective of whether exception is raised or not raised or whether the exception is handle or not handle.
finalize():- It is a **method**, Garbage collector always calls this method just before destroying any object to perform cleanup activities.

Q19. Is it possible to write any statement between try-catch and finally?
Ans. No, it is not possible to write any statement between try catch and finally. If we will try to write any statement between them then we will get compile time error.

Q20. Is it possible to take two finally blocks for the same try?

Ans. No, it is not possible to take two finally blocks for the same try. If we try to take then we will get compile time error.

Q21. Is syntax try-finally-catch is valid ?
Ans. No, this syntax is not valid. It should be like try-catch-finally then only code will compile.
Q22. What is the purpose of throw?
Ans. Sometimes we can create Exception object explicitly and we can handover that exception object to the JVM explicitly by throw keyword.
The purpose of throw keyword is to handover our created exception object explicitly to the JVM.
Example1:
```
class Test{
            public static void main(String[] args){
                        System.out.println(10/0);
           }
            }
```
In this case ArithmeticException object created implicitly and handover to the JVM automatically by the main method.

Example2:
```
Class Test{
            Public static void main(String[] args){
                        Throw new ArithmeticException("/by Zero");
                        }
            }
```
In this case creation of an exception object and handover to the JVM explicitly by the programmer.

Q23. Is it possible to throw an Error?
Ans. Yes, It is possible to throw any Throwable type including Error.

Q24. Is it possible to throw any java object?
Ans. No, we can use throw keyword only for throwable objects otherwise we will get compile time error saying incompatible type.

Q25. After throw is it allow to take any statement directly?
Ans. After throw statement we are not allow to place any statement directly violation leads to compile time error saying Unreachable Statement.
Q26. What is the purpose of throws?
Ans. The main purpose of throws keyword is to delegate the responsibilities of exception handling to the caller. It requires in the case of checked exception.
Q27. What is the difference between throw and throws?
Ans. Sometimes we can create Exception object explicitly and we can handover that exception object to the JVM explicitly by throw keyword.The main purpose of throw keyword is to handover our created exception object explicitly to the JVM. The main purpose of throws keyword is to delegate the responsibilities of exception handling to the caller.
It requires in the case of checkedexception.

Q28. What is the difference between throw and thrown?
Ans. There is no terminology of thrown in java.

Q29. Is it possible to use throws keyword for any java class?
Ans. No, we can use throws keyword only for Throwable classes. Otherwise we will get compile time error saying Incompatible types.
Q30. If we are taking catch block for an exception but there is no chance of rising that exception in try then what will happen?
Ans. If there is no chance of raising an exception in try then we are not allow to write catch block for that exception violation leads to compile time error. But this rule is applicable only for fully checked exception.
Q31. Explain Exception Handling keyword?
Ans. **Exception Handling keyword:**
Try :- To maintain Risky code.
Catch:- To maintain Exception Handling code.
Finally:- To maintain the clean up code.
Throw:- To handover our created exception object to the JVM explicitly.
Throws:- To delegate the responsibilities of Exception Handling to the caller.

Q32. Which class act as root for entire java Exception hierarchy?
Ans. Throwable class act as root for entire java Exception hierarchy.

Q33. What is the difference between Error and Exception?
Ans. Throwable class contain two child classes.
Exception:- These are mostly caused by our program and are recoverable.
Error:- These are not caused by our program, mostly caused by lake of system resources. These are non recoverable.

Q34. What is difference between checked exception and unchecked exception?
Ans. The exceptions which are checked by the compiler for smooth execution of the program at Runtime is called checked exception. Example: IOException, InterruptedException.The exceptions which are not checked by the compiler are called unchecked exception. Example: ArithmeticException,RuntimeException.
Q35.What is difference between partially checked and fully checked Exception?
Ans. A checked exception is said to be fully checked if and only if all the child classes also checked otherwise it is called partially checked exception.
Example:
IOException:- fully checked exception
Exception:- partially checked exception
Throwable:- partially checked exception
RuntimeException:- unchecked exception

Q36. What is a customized Exception?
Ans. Sometimes based on our programming requirement we have to create our own exception such type of exception are called customize Exception.
Example:
TooYoungException

TooOldException
InsufficientFundException

Q37. Explain the process of creating the customized Exception.

Ans. **Creating customized Exception:**

```
Class TooYoungException extends RuntimeException{
 TooYoungExcetpion(String desc){
            Super(desc);
            }
}
Class TooOldException extends RuntimeException
{
            TooOldException(String desc){
super(desc);
            }
}
Class custExcepiton{
            Public static void main(String[] args){
Int age=Integer.parseInt(args[0]);
If(age>60)
{
Throw new TooYoungException("Please wait some more time, definitely you will get best
match");
            }
            Else if(age<18)
            {
Throw new TooOldException("Your age is already crossed of marriage, no chance to getting
marriage");
            }
            Else
            {
                        System.out.println("Congratulation! You will get match details soon
by your email");
            }
}
```

Q38. Explain control flow in try, catch, finally.

Ans.

```
          Try{
            Statement1;
            Statement2;
Statement3;
}
Catch(X e){
Statement4;
}
Finally{
Statement5;
}
Statement6;
```

Case1:
If there is no Exception then output is
Statement1
Statement2
Statement3
Statement5
Statement6
Normal termination
Case2:
If an exception raised at statement2 and corresponding catch block has matched then output is
Statement1
Statement4
Statement5
Statement5
Normal termination
Case3:
An exception raised at statement2 and corresponding catch has not matched then output is
Statement1
Statement5
Abnormal termination
Case4:
An exception occurs at statement4 it always Abnormal termination but before that finally block will be executed and output is
Statement1
Statement2
Statement5
Abnormal termination
Case5:
If an exception raised at statement5 or statement6, it is always abnormal termination.
Q39. Can you give the most common occurred exception in your previous project.
Ans. NullPointerException, ArrayIndexOutofBoundException, StackOverFlowError, ClassCastException, NoClassDefFoundError, ExceptionInitilizerError, IllegalArgumentException, NumberFormatException, IllegalStateException, AssertionError.
Q40. Explain the cases where you used Exception Handling in your previous project?

Hibernate FAQs

1.what is the advantage of Hibernate over jdbc?

There are so many
1) Hibernate is data base independent, your code will work for all ORACLE,MySQL ,SQLServer etc.
In case of JDBC query must be data base specific. So hibernate based persistance logic is database independent persistance logic and JDBC based persistance logic is database dependent logic.
2) As Hibernate is set of Objects ,
3) No need to learn SQL language.You can treat TABLE as a Object . Only Java knowledge is

need.
In case of JDBC you need to learn SQL.
3) Dont need Query tuning in case of Hibernate. If you use Criteria Quires in Hibernate then hibernate automatically tuned your query and return best result with performance.
In case of JDBC you need to tune your queries.
4) You will get benefit of Cache. Hibernate support two level of cache. First level and 2nd level. So you can store your data into Cache for better performance.
In case of JDBC you need to implement your java cache .
5) Hibernate supports Query cache and It will provide the statistics about your query and database status.
JDBC Not provides any statistics.
6) Development fast in case of Hibernate because you dont need to write queries
7) No need to create any connection pool in case of Hibernate. You can use c3p0.
In case of JDBC you need to write your own connection pool
8) In the xml file you can see all the relations between tables in case of Hibernate. Easy readability.
9) You can load your objects on start up using lazy=false in case of Hibernate.
JDBC Dont have such support.
10) Hibernate Supports automatic versioning of rows but JDBC Not.

2.What is Hibernate?
Hibernate is an open source, light weight Object Relational Mapping tool to develop the database independent persistence login injava and j2ee based applications.
Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration and mapping files. Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks
3.What is ORM ?
ORM stands for object/relational mapping, means providing the mapping between class with table and member variables with columns is called ORM. ORM is the automated persistence of objects in a Java application to the tables in a relational database.

4.hat does ORM consists of ?
An ORM solution consists of the following four pieces:

- API for performing basic CRUD operations
- API to express queries referring to classes
- Facilities to specify metadata
- Optimization facilities : dirty checking,lazy associations fetching

5.What are the ORM levels ?
The ORM levels are:

- Pure relational (stored procedure.)
- Light objects mapping (JDBC)
- Medium object mapping
- Full object Mapping (composition,inheritance, polymorphism, persistence by reachability)

.

6.Why do you need ORM tools like hibernate?
The main advantage of ORM like hibernate is that it can develop the database independent persistence logic. Apart from this, ORM provides following benefits:

- **Improved productivity**
    - High-level object-oriented API
    - Less Java code to write
    - No SQL to write
- **Improved performance**
    - Sophisticated caching
    - Lazy loading
    - Eager loading
- **Improved maintainability**
    - A lot less code to write
- **Improved portability**
- ORM framework generates database-specific SQL for you

7.What Does Hibernate Simplify?
Hibernate simplifies:

- Saving and retrieving your domain objects
- Making database column and table name changes
- Centralizing pre save and post retrieve logic
- Complex joins for retrieving related items
- Schema creation from object model

8.What is the main difference between Entity Beans and Hibernate ?

1)In Entity Bean at a time we can interact with only one data Base. Where as in Hibernate we can able to establishes the connections to more than One Data Base. Only thing we need to write one more configuration file.

2) EJB need container like Weblogic, WebSphere but hibernate don't nned. It can be run on tomcat.

3) Entity Beans does not support OOPS concepts where as Hibernate does.

4) Hibernate supports multi level cacheing, where as Entity Beans doesn't.

5) In Hibernate C3P0 can be used as a connection pool.

6) Hibernate is container independent. EJB not.

9.What are the Core interfaces and classes of Hibernate framework?

The five core interfaces are used in just about every Hibernate application. Using these interfaces, you can store and retrievepersistent objects and control transactions.

- Configuration class (org.hibernate.cfg package)
- Session interface (org.hibernate package)
- SessionFactory interface (org.hibernate package)
- Transaction interface (org.hibernate package)
- Query and Criteria interfaces (org.hibernate package)

10.What is the general flow of Hibernate communication with RDBMS?
The general flow of Hibernate communication with RDBMS is :

- Load the Hibernate configuration file and create configuration object. It will automatically load all hbm mapping files because mapping file can be configured in configuration file.
- Create session factory from configuration object
- Get one session from this session factory
- Create HQL Query
- Execute query to get list containing Java objects.

11.What is the need for Hibernate mapping file?
Hibernate mapping file is used to provides the mapping between java class with table member variables with column names of the table. And also we can configure primary key generation algorithm, relations and so on. Typical mapping file look as follows:

13.What role does the Session interface play in Hibernate?
The main runtime interface between a Java application and Hibernate The Session interface is the primary interface used by Hibernate applications. It is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It allows you to create query objects to retrieve persistent objects.
The main function of the Session is to offer create, read and delete operations for instances of mapped entity classes. Instances may exist in one of three states:

transient: never persistent, not associated with any Session
persistent: associated with a unique Session
detached: previously persistent, not associated with any Session

Session session = sessionFactory.openSession();
Session interface role:

- Wraps a JDBC connection
- Factory for Transaction
- Holds a mandatory (first-level) cache of persistent objects, used when navigating the object graph or looking up objects by identifier

14.What role does the SessionFactory interface play in Hibernate?
SessionFactorys are immutable. The behaviour of a SessionFactory is controlled by properties supplied at configuration time. These properties are defined on Environment. The application obtains Session instances from a SessionFactory. There is typically a single SessionFactory for the whole application—created during application initialization. The SessionFactory caches generate SQL statements and other mapping metadata that Hibernate uses at runtime. It also holds cached data that has been read in one unit of work and may be reused in a future unit of work
Implementors must be threadsafe.
SessionFactory sessionFactory = configuration.buildSessionFactory();

15.What are the most common ways to specify the Hibernate configuration properties?
The most common methods of Hibernate configuration are:

- Programmatic configuration

  By using setProperty(-) method of org.hibernate.cfg.Configuration.

- XML configuration (hibernate.cfg.xml)
- By using .properties file
- By Using annotaions.(from Hibernate 3.3 on words)

16.How do you map Java Objects with Database tables?

- First we need to write Java domain objects (beans with setter and getter).
- Write hbm.xml, where we map java class to table and database columns to Java class variables.

Example :
```
<hibernate-mapping>
  <class name="com.durgasoft.EmployeeBean"  table="EMPLOYEE">
     <id name=”eid” colume=”id”/>
   <property name="ename" column="NAME" length="255"
     not-null="true"  type="java.lang.String"/>
   <property name="address" column="ADDR" length="255"
     not-null="true"  type="java.lang.String"/>
 </class>
</hibernate-mapping>
```

17.How do you define sequence generated primary key algorithm in hibernate?
By using <id>, <generator> tags we can configure the primary key and primary key generation algorithm.
Example:-
```
<id name="userid" column="USER_ID" type="java.lang.Long">
   <generator class="sequence">
     <param name="table">SEQ_NAME</param>
```

```
  <generator>
</id>
```

18.What is component mapping in Hibernate?

- A component is an object saved as a value, not as a reference
- A component can be saved directly without needing to declare interfaces or identifier properties
- Required to define an empty constructor
- Shared references not supported

19 . Difference between getCurrentSession() and openSession() in Hibernate ?

getCurrentSession() :
Obtains the current session. The "current session" refers to a Hibernate Session bound by Hibernate behind the scenes, to the transaction scope.
A Session is opened when getCurrentSession() is called for the first time and closed when the transaction ends. It is also flushed automatically before the transaction commits. You can call getCurrentSession() as often and anywhere you want as long as the transaction runs. Only the Session that you obtained with sf.getCurrentSession() is flushed and closed automatically.

openSession() :
If you decide to use manage the Session yourself the go for sf.openSession() , you have to flush() and close() it.
It does not flush and close() automatically.
Example :

```
Transaction tx =session.berginTransaction();

Session session = factory.openSession();

try {
tx.begin();

// Do some work
session.createQuery(...);
session.persist(...);

session.flush(); // Extra work you need to do

tx.commit();
}
catch (RuntimeException e) {
tx.rollback();
throw e; // or display error message
}
finally {
```

```
session.close(); // Extra work you need to do
}
```

20.What are the types of Hibernate instance states ?
Three types of instance states:

- Transient -The instance is not associated with any persistence context
- Persistent -The instance is associated with a persistence context
- Detached -The instance was associated with a persistence context which has been closed – currently not associated

21.What are the types of inheritance models in Hibernate?
There are three types of inheritance models in Hibernate:

- Table per class hierarchy
- Table per subclass
- Table per concrete class

22.What is Hibernate Query Language (HQL)?
Hibernate Query Language is query language which is used to develop the data independent query language in the application. This HQL queries are not related to any database. Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

23.What are the ways to express joins in HQL?
HQL provides four ways of expressing (inner and outer) joins:-

- An *implicit* association join
- An ordinary join in the FROM clause
- A fetch join in the FROM clause.
- A *theta-style* join in the WHERE clause.

24 . Transaction with plain JDBC in Hibernate ?

If you don't have JTA and don't want to deploy it along with your application, you will usually have to fall back to JDBC transaction demarcation. Instead of calling the JDBC API you better use Hibernate's Transaction and the built-in session-per-request functionality:

To enable the thread-bound strategy in your Hibernate configuration:

```
set hibernate.transaction.factory_class to
org.hibernate.transaction.JDBCTransactionFactory
set hibernate.current_session_context_class to thread

Session session = factory.openSession();
Transaction tx = null;
try {
```

```
tx = session.beginTransaction();

// Do some work
session.load(...);
session.persist(...);

tx.commit(); // Flush happens automatically
}
catch (RuntimeException e) {
tx.rollback();
throw e; // or display error message
}
finally {
session.close();
}
```

25 . What are the general considerations or best practices for defining your Hibernate persistent classes?

1.You must have a default no-argument constructor for your persistent classes and there should be getXXX()and setXXX() methods for all your persistable instance variables.

2.You should implement the equals() and hashCode() methods based on your business key and it is important not to use the id field in your equals() and hashCode() definition if the id field is a surrogate key (i.e. Hibernate managed identifier). This is because the Hibernate only generates and sets the field when saving the object.

3. It is recommended to implement the Serializable interface. This is potentially useful if you want to migrate around a multi-processor cluster.

4.The persistent class should not be final because if it is final then lazy loading cannot be used by creating proxy objects.

26 . Difference between session.update() and session.lock() in Hibernate ?

The session.update method is used to update the persistence object in the in the database. The session.lock() method simply reattaches the object to the session without checking or updating the database on the assumption that the database in sync with the detached object. It is the best practice to use either session.update(..) or session.saveOrUpdate(). Use session.lock() only if you are absolutely sure that the detached object is in sync with your detached object or if it does not matter because you will be overwriting all the columns that would have changed later on within the same transaction.

27.What are the Collection types in Hibernate ?

- Set
- List
- Array

- Map
- Bag

28.What is the difference between sorted and ordered collection in hibernate? sorted collection vs. order collection :-

sorted collection	order collection
A sorted collection is sorting a collection by utilizing the sorting features provided by the Java collections framework. The sorting occurs in the memory of JVM which running Hibernate, after the data being read from database using java comparator.	Order collection is sorting a collection by specifying the order-by clause for sorting this collection when retrieval.
If your collection is not large, it will be more efficient way to sort it.	If your collection is very large, it will be more efficient way to sort it .

29.What are the ways to express joins in HQL?
HQL provides four ways of expressing (inner and outer) joins:-

- An *implicit* association join
- An ordinary join in the FROM clause
- A fetch join in the FROM clause.
- A *theta-style* join in the WHERE clause.

30.What do you mean by Named – SQL query?
Named SQL queries are defined in the mapping xml document and called wherever required.
Example:

```
<sql-query name = "empdetails">
  <return alias="emp" class="com.durgasoft.Employee"/>
     SELECT emp.EMP_ID AS {emp.empid},
             emp.EMP_ADDRESS AS {emp.address},
             emp.EMP_NAME AS {emp.name}
     FROM Employee EMP WHERE emp.NAME LIKE :name
</sql-query>
```

Invoke Named Query :

```
List people = session.getNamedQuery("empdetails")
                 .setString("TomBrady", name)
                 .setMaxResults(50)
                 .list();
```

31.How do you invoke Stored Procedures?

```
<sql-query name="selectAllEmployees_SP" callable="true">
 <return alias="emp" class="employee">
   <return-property name="empid" column="EMP_ID"/>
   <return-property name="name" column="EMP_NAME"/>
   <return-property name="address" column="EMP_ADDRESS"/>
    { ? = call selectAllEmployees() }
 </return>
</sql-query>
```

32.Explain Criteria API

The interface org.hibernate.Criteria represents a query against a particular persistent class. The Session is a factory for Criteria instances. Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like "search" screens where there is a variable number of conditions to be placed upon the result set.

Example :

```
List employees = session.createCriteria(Employee.class)
                        .add(Restrictions.like("name", "a%") )
                        .add(Restrictions.like("address", "Boston"))
                        .addOrder(Order.asc("name") )
                        .list();
```

33.What’s the difference between load() and get()?

load()	get()
Only use the load() method if you are sure that the object exists.	If you are not sure that the object exists, then use one of the get() methods.
load() method will throw an exception if the unique id is not found in the database.	get() method will return null if the unique id is not found in the database.
load() just returns a proxy by default and database won’t be hit until the proxy is first invoked.	get() will hit the database immediately.

34.What is the difference between and merge and update ?

Use update() if you are sure that the session does not contain an already persistent instance with the same identifier, and merge() if you want to merge your modifications at any time without consideration of the state of the session.

35.Define cascade and inverse option in one-many mapping?

cascade - enable operations to cascade to child entities.

cascade="all|none|save-update|delete|all-delete-orphan"

inverse - mark this collection as the "inverse" end of a bidirectional association.
inverse="true|false"
Essentially "inverse" indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

36.Define HibernateTemplate?
org.springframework.orm.hibernate.HibernateTemplate is a helper class which provides different methods for querying/retrieving data from the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

37.What are the benefits does HibernateTemplate provide?
The benefits of HibernateTemplate are :

- HibernateTemplate, a Spring Template class simplifies interactions with Hibernate Session.
- Common functions are simplified to single method calls.
- Sessions are automatically closed.
- Exceptions are automatically caught and converted to runtime exceptions.

38. How do you switch between relational databases without code changes?
Using Hibernate SQL Dialects , we can switch databases. Hibernate will generate appropriate hql queries based on the dialect defined.

39.If you want to see the Hibernate generated SQL statements on console, what should we do?
By using “show_sql” property of the hibernate configuration file
In Hibernate configuration file set as follows:
<property name="show_sql">true</property>

40.What are derived properties?
The properties that are not mapped to a column, but calculated at runtime by evaluation of an expression are called derived properties. The expression can be defined using the formula attribute of the element.

41.Define cascade and inverse option in one-many mapping?
cascade - enable operations to cascade to child entities.
cascade="all|none|save-update|delete|all-delete-orphan"

inverse - mark this collection as the "inverse" end of a bidirectional association.

inverse="true|false"
Essentially "inverse" indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

42 . Explain about transaction file?
Transactions denote a work file which can save changes made or revert back the changes. A transaction can be started by session.beginTransaction() and it uses JDBC connection, CORBA or JTA. When this session starts several transactions may occur.

49.How can Hibernate be configured to access an instance variable directly and not through a setter method ?
By mapping the property with access="field" in Hibernate metadata. This forces hibernate to bypass the setter method and access the instance variable directly while initializing a newly loaded object.

50.How can a whole class be mapped as immutable?
Mark the class as mutable="false" (Default is true),. This specifies that instances of the class are (not) mutable. Immutableclasses, may not be updated or deleted by the application.

51 . Explain about transparent persistence of Hibernate?
Transparent persistence is provided for Plain old Java objects or POJOs. For proper functioning of the applications importance should be given to the methods equals () and hash Code methods (). It has a requirement which should be strictly followed in the applications which is a no-argument constructor.

52 . Explain about the dirty checking feature of Hibernate?
Dirty checking feature of the Hibernate allows users or developers to avoid time consuming data base write actions. This featuremakes necessary updations and changes to the fields which require a change, remaining fields are left unchanged or untouched.

53 . What is the effect when a transient mapped object is passed onto a Sessions save?
When a Sessions save () is passed to a transient mapped object it makes the method to become more persistent. Garbage collection and termination of the Java virtual machine stays as long as it is deleted explicitly. It may head back to its transient state.

54 . Explain about addClass function?
This function translates a Java class name into file name. This translated file name is then loaded as an input stream from the Javaclass loader. This addClass function is important if you want efficient usage of classes in your code.

Inner class
1. What is inner class and when we should go for inner classes?

Some times we can declare a class inside another class such type of classes are called inner classes

Example

```
  Class Car{

//more code here

Class Engine{

     //more code here

}

}
```

Without existing Car object there is no chance of existing Engine object, hence Engine class has declared inside Car class.

2.How many types of inner classes are present?

There are four types of inner classes are present

- Normal or regular inner class
- Method local inner class
- Anonymous inner class
- Static nested class

3.What is method local inner class?

- Sometimes we can declare a class inside a method such type of classes are called method local

inner classes

- The main purpose of method local inner classes is to define method specific functionality

The scope of method local inner classes is the scope of the method where it is declared.

- This is the mostly rarely used type of inner classes.

Example

```
      class Test{
   public  void  m1(){
          class Inner {
                     public void sum(int I,int j){
            System.out.println(i+J);
        }//sum
         }//inner
        Inner  i=new Inner();
        i.sum(10,20);
        //more code here
       I.sum(100,303);
       //more code here
      i.sum(102,84);
     }//m1()
 Public  static  void main(){
   New Test().m1();
}
}
```

4.What is anonymous inner class?

- Some times we can declare a inner class without name such type of inner classes are called

  anonymous inner classes

- Anonymous inner classes can be divided into 3 categories
  - Anonymous inner class that extends a class
  - Anonymous inner class that implements an interface
  - Anonymous inner class that defines inside a method argument

ANONYMOUS INNER CLASS THAT EXTENDS A CLASS

Example

```
Class popcorn{

    Public void taste(){

        System.out.println("it is salty");

    }

    //more code here

}

Class Test{

    Public static void main(String[] args)

        {

        Popcorn p=new Popcorn()

        {   // here we are creating child class for popcorn

                    Public void taste(){

                    System.out.println("it is sweet");

            }
```

```
                    };//here semicolon indicates we r creating child class object
                    with parent

                    // class reference here child class dosent contain name

        p.taste()// it is sweet

        Popcorn p=new Popcorn();

        p1.taste() //it is salty

        }

    }
```

ANONYMOUS INNER CLASS THAT IMPLEMENTS AN INTERFACE

example

```
class Test{

        Public static void main(String[] args){

    Runnable r=new Runnable(){

        Public void run(){

            for(int i=0;i<10;i++){

            System.out.printin("child thread");

                    }

                }

    };

        Thread t=new Thread(r);

    t.start();

    for(int i=0;i<10;i++){

        System.out.printin("main thread");
```

```
          }
    }
    }
```

Don’t become fool that here we are creating object of interface Runnable.Here we are actually

creating an object of class that is implemented Runnable interface.

ANONYMOUS INNER CLASS THAT DEFINES INSIDE A METHOD ARGUMENT

<u>Example</u>

```
Class Test{
                Public static void main(String[] args){
            New Thread(new Runnable()
                                                    {
                            Public void run(){
                            for(int i=0;i<10;i++){
                                  System.out.printin(“child thread”);
                                 }
                            }
                             }).start();
                      for(int i=0;i<10;i++){
            System.out.printin(“main thread”);
                       }
            }//main
       }//Test
```

5. With out having name of class how we can create an object and utilize the functionality of Anonymous inner class?

By using parent class names

6. What is difference between anonymous inner class and general class?

- A general class can extend only one class at a time of course inner class can extend only one class at a Time.
- A general class can implement any no of interfaces at a time but a anonymous inner class can
- implement only one interface at a time
- A general class can extend a class and can implement an interface simultaneously but an

  anonymous inner class can extend a class or can implement an interface one at a time but not

  both simualtaneously

7. What is difference between normal inner class and static nested class?

Normal Inner Class	**Static Nested Class**
1. Inner class object always associated with outer class object ie without existing outer class object there is no chance of existing inner class object.	1. Nested class object never associated with outer class object , ie without existing outer class object inner class object can exist
2. Inside normal inner class we cant declare static members.	3. Inside static nested class can declare static members
4. We cant place main method in normal inner class and hence innocation of inner class directly from command prompt is not possible.	2. We can place main method in static nested class and hence innocation of nested class directly from command prompt is possible
5. From normal inner class we can access both static and non static members of outer class.	3. From static nested class we can access only static member of outer class

8.What is static nested calss?why the term nested instead of inner in static nested class?

Some times we can declare inner class with static modifier such type of inner class are called static

nested classes.the term nested instead of static because without existing outer class object inner

class object can exist.

Example

```
Class outer{

Static class Nested{

      Public static void main(String[] args){

            System.out.println("nested class main()");

      }

}

Public static void main(String[] args){

      System.out.println("outer class main()");

}

}
```

- Java Outer

  O/P

  Outer class main()

- Java Outer$Nested

  O/P

Nested class main()

9. Inside inner class is it possible to declare main()?

No it is not possible to declare main () inside inner class but in static nested class it is possible for

Example refer above code

JDBC

1: What is the difference between Database and Database management system?
Ans: Database is a collection of interrelated data. Database management system is a software which can be used to manage the data by storing it on to the data base and by retrieving it from the data base. And DBMS is a collection of interrelated data and some set of programs to access the data.

There are 3 types of Database Management Systems.

- Relational DataBase Management Systems(RDBMS): It is a software system, which can be used to represents data in the form of tables. RDBMS will use SQL2 as a Queries language.
- Object Oriented DataBase Management Systems(OODBMS): It is a software system, which can be used to represent the data in the form of objects. This DBMS will use OQL as a Query language.
- Object Relational DataBase Management Systems(ORDBMS): It is a DBMS which will represents some part of the data in the form of tables and some other part of the data in the form of objects. This management system will use SQL3 as a Query Language, it is a combination of SQL2 and OQL.

2: How a query could be executed when we send a query to Database?
When we send an SQL Query from SQL prompt to the DataBaseEngine, then Database Engine will take the following steps.

Query Tokenization: This phase will take SQL query as an input and devide into stream of tokens.

- Query Parsing: This phase will take stream of tokens as an input, with them it tried to construct a query tree. If query parser constructs query tree successfully then it was an indication that no grammatical mistakes in the taken SQL query. Otherwise there are some syntactical errors in the taken SQL query.

- Query Optimization: This phase will take query tree as an input and performs number of query optimization mechanisms to reduce execution time and memory utilization.
- Query Execution: This phase will take optimized query as an input and executes that SQL query by using interpreters internally as a result we will get some output on the SQL prompt.

3: What is Driver? How many Drivers are available in JDBC? What are the types?

- It is a process of interacting with the database from a java application.
- In JDBC applications we must specify the complete database logic in java application as for the java API representations, later on we need to send java represented database logic to the database engine(DBE).
- DBE must execute the database logic but it was configured as per the java representations but DBE able to understand only Query Language representations.
- At the above situation if we want to execute our database logic, we need to use one interface in between java application and the database, that interface must convert java representations to query language representations and query language representations to java representations. Now this interface is called as a "Driver".

Driver:

- It is a software or an interface existed in between a java application and database, which will map java api calls with query language api calls and vice versa.
- Initially sun Microsystems has provided "driver interface" to the market with this sun Microsystems has given an intimation to all the database vendors to have their own implementation as per their requirements for the Driver interface.
- As a response all the database vendors are providing their own implementation for the Driver interface inorder to interact with the respective databases from a java application.
- The users of the respective databases they must get the respective database provided Driver implementation from the database software and make use as part of the JDBC applications to interact with the respective databases form a javaapplication.

Types of Drivers:

There are 180+ number of Drivers in the market. But all these Drivers could be classified into the following 4 types.

- Type 1 Driver
- Type 2 Driver
- Type 3 Driver
- Type 4 Driver

Type 1 Driver:

  - Type 1 Driver is also called as Jdbc-Odbc Driver or Bridge Driver.

- Jdbc-Odbc Driver is an implementation to Driver interface provided by the sun Microsystems along with the javasoftware.
- Jdbc-Odbc Driver internally depends on the Microsoft product Odbc Driver.
- Odbc is nothing but open database connectivity. It is a open specification which can be used to interact with any type of databases.

Advantages:

- This Driver is already available with java software that's why no need to bother about how to get the Driver implementation explicitily.
- Allmost all the databases could support this Driver.

Dis advantages:

- This Driver internally depends on Odbc Driver that's why it is not suitable for internet or web applications or network applications.
- This Driver is a slower Driver, why because Jdbc-Odbc Driver will convert java calls to Odbc calls. Then Odbc Driver has to convert Odbc calls to query language calls.
- This driver is not portable Driver why because it was not complete the java implementations in Jdbc-Odbc Driver.
- It we want to use Jdbc-Odbc Driver in our jdbc applications then we must require to install Odbc-Native Library.

Type 2 Driver:

Type 2 Driver is also called as "part java part native Driver". i.e., this Driver was designed by using some part of the javaimplementations and some other part of the database vendor provided native implementations. This Driver is also called as "native driver".

Advantages:
When compared to Type 1 driver it is efficient driver why because Type 2 driver directly will convert java api calls to database vendor api calls.

Dis advantages:

- If we want to use Type 2 Driver in our Jdbc applications then we must require to install database vendor native api.
- It is a costful Driver.
- It is not suitable for web applicadtions, distributed applications and web applications.
- Type 2 Driver performance is low when compared to Type 3 and Type 4 drivers.
- This driver is not portable driver. Why because this driver was not designed completely in java technology.

Type 3 Driver:

- It is also called as middleware database access server driver.

- This driver could be used to interact with multiple databases from the multiple clients.
- This driver could be used in collaboration with application server.
- This driver is suggestable for network applications.

Advantages:

- It is a fastest driver among all the drivers available in the market.
- To use Type 3 driver in our jdbc applications it is not required to install odbc native library and database native library.
- It is very much suitable for network applications.

Dis advantages:

- This driver is not suitable for simple jdbc applications.
- This driver requires minimum 3-Tier Architecture.
- When compared to Type1 and Type2 drivers.. Type3 driver is efficient and portable. But when compared to Type4 driver, Type3 driver is not portable.

Type 4 Driver:

- This driver is also called as pure java driver i.e, this driver was completely implemented by using java technology.
- When compared to Type1, Type2 and Type3 drivers.. Type4 driver is portable driver.
- Type4 driver can be used for any kind of applications.
- Type4 driver is a cheapest driver when compared to all the drivers that’s why it is frequently used driver.

4: What is JDBC and What are the steps to write a JDBC application?

The process of interacting with the database from a java application is called as JDBC(Java Database Connectivity)
To interact with the database from a java application we will use the following five steps.

1. load and register the driver.
2. Establish a connection between java application and the database.
3. prepare either statement object or PreparedStatement object or CallebleStatement object as per the application requirements.
4. write and executer the sql queries.
5. terminate the connection which we have established.

5: How to load a JDBC driver?_

- In general sun Microsystems has provided Driver interface for this all the database vendors has provided their own implementation.

- If we want to use the database vendor provided Driver implementation to our jdbc application, first we need to make the availability of the respective Driver's .class file to JVM, for this we need to set class path environment variable to the location where we have the driver implementation.
- Sun Microsystems is also provided an implementation to the Driver interface in the form of JdbcOdbcDriver class as part of the java software.
- If we want to use JdbcOdbcDriver in our jdbc applications no need to set class path environment variable. Why because it was already available in the java software's pre-defined library.
- JdbcOdbcDriver internally depends on the mocrosoft product Odbc driver. If we want to use the JdbcOdbcDriver in our jdbc applications first we must configure Odbc driver, for this we will use the following path.

Start/ conrlol panel / performance and maintenance / administrative tools / data source(Odbc)/ user dsn / click on Add / select microsofr Odbc for oracle / finish / provide data source name only / click on ok / ok.

- To load the driver's class byte code to the memory we will use the following method.

Public void forName(String class name)

Eg: Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Where forName() is a static method, which can be used to load the respective driver class byte code to the memory.

- Each and every driver class has already a static block at the time of loading the respective driver class byte code to the memory automatically the available static block could be executed, by this DriverManager.registerDriver(….) method will be executed as part of the static block.
- By the execution of the registerDriver(….) method automatically the specified driver will be register to the jdbc application.
- If you want to design any jdbc application, we need to use some pre-defined library, which was provided by the Jdbc API in the form of java.sql package, that's why we need to import java.sql package in our jdbc application.

Note:- The best alternative for Class.forName(..) is
DriverManagre.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());
To register the driver.

6: How to establish a Database connection between java application and Database?

If we want to establish a connection between java application and the database we will the following piece of code.

Connection con=
DriverManager.getConnection(“jdbc:odbc:nag”,”nag”,”system”,”manager”);

Where getConnectin() is a static method from DriverManager class, which can be used to return connection object.

7: Basically Connection is an interface, how getConnection() will create an object for Connection interface?
Ans: Connection is an interface from java.sql package, for which getConnection(_) was return an anonymous inner class object of the Connection interface.

Note:- Anonymous inner class is a nameless inner class, which can be sued to provide an implementation either for the interfaces or for abstract classes.

```
Eg:   interface I
      {
           void m1();
       }
      Class Outer
      {
               I  i = new I(){
                                    public void m1()
                                    {

                                    }
                                    public void m2()
                                    {

}
                                  }
         }
         Outer o = new Outer();
         o.i.m1();   à  correct
         o.i.m2();   à  wrong
```

getConnection(_) is a static method from DriverManager class, which will call internally connect() method, this connect() will establish a virtual socket connection in between the java application and the database.
8: What is the requirement to use Statement object?

- After establishing a connection between java application and the database we need to write the sql queries and we need to execute them.
- To execute the sql queries we will use some pre-defined library, which was defined in the form of Statement object, PreparedStattement object and CallableStatement object.
- As per the application requirements we need to create either Statement object or CallableStatement object and PreparedStatement object.

- To create Statement object dwe will use the following method from connection object.

public Statement createStatement()
Eg: Statement st = con.createStatement();
9*: How to execute SQL Queries from a java application?*

To execute the sql queries we will use the following methods from Statement object.

- st.executeQuery(…)
- st.executeUpdate(…)
- st.execute(…)

10: What are the differences between executeQuery(…), executeUpdate(…) and execute(…)
methods?
Ans: where executeQuery() can be used to execute selection group sql queries to fetch the data from database.
When we use selection group sql query with executeQuery() then JVM will send that sql query to the database engine, database engine will execute it, by this database engine(DBE) will fetch the data from database and send back to the java application.
Java is a purely object oriented technology. That's why the jdbc application will maintain the fetched data from database, in the form of an object at heap memory, called as ResultSet object.

public ResultSet executeQuery(String sqlquery)

where executeUpdate() can be used to execute updation group sql query to update the database. When we provide updation group sql query as a parameter to executeUpdate(), then JVM will send that sql query to DBE, here DBE will execute it and perform updations on the database, by this DBE will identify the number of records got updated value called as "records updated count" and return back to the java application.

public int executeUpdate(String sqlquery)

where execute() can be used to execute either selection group sql queries or updation group queries.
When we use selection group sql query with the execute() then we will get ResultSet object at heap memory with the fetched data. But execute() will return "true" as a Boolean value.
When we use updation group sql query with execute() then we will get " records updated count value" at jdbc application. But execute() will return "false" as a Boolean value.

public boolean execute(String sqlquery)

11: How to create a table dynamically from a jdbc application?.

```
     //import section
      import java.sql.*;
     import java.io.*;

     public class CreateTableEx
     {
            public static void main(String[] args)throws Exception
            {
                     //create buffered reader object
              BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

              //load and register the driver
              Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

              //establish connection
              Connection con =
DriverManager.getConnection("jdbc:odbc:nag","system","durga");

              //create statement object
              Statement st = con.createStatement();

              //take table name as dynamic input
              System.out.println("Enter table name");
              String tname = br.readLine();

              //execute sql query
              St.executeUpdate("create table"+tname+"(eno number,ename
varchar2(10),esal number,eaddr varchar2(10))");

              System.out.println("table created successfully");

              //closing the connection
              con.close();
     }
}
```

12: How to insert records into a table from a JDBC application?

```
import java.sql.*;
import java.io.*;
public class InsertTableEx
{
  public static void main(String[] args) throws Exception
  {
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con  =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga");
Statement st = con.createStatement();
while(true)
{
          System.out.println("Enter emp number");
          Int eno = Integer.parseInt(br.readLine());
          System.out.println("Enter emp name");
          String ename = br.readLine();
          System.out.println("Enter emp sal");
          Float esal = Float.parseFloat(br.readLine());
          System.out.println("Enter emp address");
          String eaddr = br.readLine();
st.executeUpdate("insert into emp1 values("+eno+",'"+ename+"',"+esal+",'"+eaddr+"')");
          System.out.println("read successfully inserted");
          System.out.println("one more record[y/n]);
          String option = br.readLine();
          If(option.equals("n"))
                    break;
}
     }
}
```

13: How to update a table from a jdbc application?.

```
import java.sql.*;
public class UpdateTableEx
{
  public static void main(String[] args)throws Exception
  {
          //load n register the driver in alternative way to Class.forName
          DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xee","system","durga");

Statement st = con.createStatement();
int updateCount = st.executeUpdate("update emp1 set esal = esal+500 where
esal<9000");
System.out.println("records updated……."+updateCount);

con.close();
  }
}
```

14: How to delete records from a table from jdbc application?.

```
import java.sql.*;
public class DeleteTableEx
{
  public static void main(String[] args)throws Exception
  {
        Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga");

        Statement st = con.createStatement();
        int updateCount = sst.executeUpdate("delete from emp3 where esal>=7000");
        System.out.println("records deleted………"+updateCount);

        con.close();
  }
}
```

15:What is ment by ResultSet object and How to Fetch the Data from Database?.

ResultSet:-

ResultSet is an Object which can be used to maintain the fetched data from database in the JDBC applications

When we execute a selection group sql query, either with executeQuety() or with execute() automatically a ResultSet object will be created at heap memory with the fetched data from database.

- To get the ResultSet object reference directly we will use executeQuery(..).
- When we create a ResultSet object automatically a cursor will be create called as "ResultSet cursor" to read the data from ResultSet object.
- When we create the ResultSet object by default ResultSet cursor will be created before the first record.
- If we want to read the data from ResultSet object every time we need to check whether the next record is available or not. If the next record is available automatically we need to move that ResultSet cursor to next record position.
- To perform this work we will use the following method from ResultSet interface.

```
public boolean next()
```

- After getting ResultSet cursor to a record position then we need to get the data from respective fields of the particular record, for this we will use following method.

```
public xxx getXxx(int fno)
        (or)
public xxx getXxx(String fname)

        where xxx is byte, shor, int, long, float, double, char.

Eg:   while(rs.next())
        {
System.out.println(rs.getInt(1)+"   "+rs.getString(2)+"   "+rs.getFloat(3)+"
"+rs.getString(4));
                }
```

The following example demonstrates how to fetch the data from database through ResultSet object.

```
import java.sql.*;
public class FetchEx
{
  public static void main(String[] args)throws Exception
  {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");

        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from emp1");
        System.out.println("ENO      ENAME      ESAL      EADDR");
        System.out.println("********************************");
        while(rs.next())
        {
                System.out.println(rs.getInt(1)+""+rs.getString(2)+"
"+rs.getFloat(3)+"             "+rs.getString(4));
        }
  }
}
```

16:Ingeneral execute() method can be used to execute selection group SQl queries for getting the data fromDatabase , but execute() return a boolean value true so here how it possible to fetch the data from database?

- Execute() can be used to execute both selection group sql query and updation group sql query.

- If we use execute() to execute a selection group sql query then DBE(Database engine) will execute that sql query and send back the fetched data from database to java application. Now java application will prepare a ResultSet object with the fetched data but execute() will return "true" as a Boolean value.
- At this situation to get the reference of the ResultSet object explicitily, we will use the following method from Statement object.

```
            public ResultSet getResultSet()

  Eg:  boolean b = st.execute("select * from emp1");
        System.out.println(b);
        ResultSet rs = st.getResultSet();
```

17:Ingeneral execute() method can be used to execute updatation group SQl queries for updating the data onDatabase , but execute() return a boolean value false so here how it possible to get the records updated count value(int value)?

- Execute() can be used to execute both selection group sql queries and updation group sql queries.

- If we use execute() to execute an updation group sql query then DBE will execute it and send back the records updated count value to the java application. But execute() will return "false" as a Boolean value. At this instance, to get the records updated count value explicitly we will use the following method from Statement object.

```
            public int getUpdateCount()

  Eg:      import java.sql.*;
            public class FetchEx
            {
                        public static void main(String[] args)throws Exception
                        {
                                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");

                                    Statement st = con.createStatement();
boolean b = st.execute("update emp1 set esal=esal+500 where esal<9000");
System.out.println(b);
int updateCount = st.getUpdateCount();j
System.out.println(updateCount);
                        }
            }
```

18: If we use selection group SQL query to executeUpdate() ,what happened?

- If we use selection group sql query as a parameter to executeUpdate(…) then JVM will send that sql query to the DBE, DBE will fetch the data and send back to the java application here java application will store the fetched data in the form of ResultSet object. But executeUpdate() is expecting records updated count value.

  Due to this contradiction JVM will rise an exception like java.lang.SQLException.

If we handle the above exception properly then we will get ResultSet abject and we will get the data from Database

```
import java.sql.*;
class Test
{
public static void main(String[] args)
{
        Statement st=null;
        try
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

               Connection con =
DriverManager.getConnection("jdbc:odbc:nag","system","durga");

                st = con.createStatement();
boolean b = st.executeUpdate("select * from emp1");
        }
        catch(Exception e)
        {
                ResultSet rs=st.getResultSet();
        System.out.println("ENO     ENAME     ESAL     EADDR");
        System.out.println("********************************");
        while(rs.next())
        {
                System.out.println(rs.getInt(1)+""+rs.getString(2)+"
"+rs.getFloat(3)+"           "+rs.getString(4));
        }

                e.printStackTrace();
        }

}
```

19: If we use updatation group SQL query to executeQuery() ,what happened?

- If we use updation group sql query as a parameter to executeQuery() then JVM will send that sql query to the DBE, DBE will perform updations on the database and send back records updated count value to the java application. But here executeQuery() is expecting ResultSet object reference.

Due to this contradiction JVM will rise an exception like java.lang.SQLException.

```
import java.sql.*;
class Test
{
public static void main(String[] args)
{
        Statement st=null;
        try
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

              DriverManager.getConnection("jdbc:odbc:nag","system","durga");

                st = con.createStatement();
  boolean b = st.executeQuery("update  emp1 set esal=esal+1000 where         esal
<10000");
        }
        catch(Exception e)
        {
                int count=st.getUpdateCount();
                System.out.println(count);

                e.printStackTrace();
        }

}
```

20: What is ment by ResultSet and What are the types of ResultSets are available in JDBC application?

In jdbc applications ResultSets could be classified in the following two ways.

- <u>On the basis of ResultSet privilizations (Concurancy):-</u>

-

There are 2 types of ResultSets.

  - Read only ResultSet
  - Updatable ResultSet

<u>Read only ResultSet:</u>- It is a ResultSet, which will allow the users to read the data only. To refer this ResultSet, we will use the following constant from ResultSet interface.

public static final int CONCUR_READ_ONLY;

Updatable ResultSet:- If is a ResultSet object, which will allow users to perform some updations on its content. To refer this ResultSet we will use the following constant from ResultSet interface.

public static final int CONCUR_UPDATABLE;

2)On the basis of the ResultSet cursor movement:-

There are 2 types of ResultSets.

- Forward only ResultSet
- Scrollable ResultSet

Forward only ResultSet:- It is a ResultSet object, which will allow the users to iterate the data in any forward direction. To refer this ResultSet object we will use the following constant from ResultSet interface.

public static final int TYPE_FORWARD_ONLY;

Scrollable ResultSet:- These are the ResultSet objects, which will allow the users to iterate the data in both forward and backward directions.
There are 2 types of Scrollable ResultSets.

- Scroll sensitive ResultSets
- Scroll in sensitive ResultSets.

21: What is the difference between ScrollSensitive ResultSet and ScrollInsensitive ResultSets?

Ans: Scroll sensitive ResultSet is a ResultSet object, which will allow the later updations from database automatically after creating it. To refer this ResultSet we will use the following constant.
public static final int TYPE_SCROLL_SENSITIVE;

Scroll insensitive ResultSet is a ResultSet object, which will not allow later updations from database after creating it. To refer this ResultSet we will use the following constant from ResultSet interface.

- public static final int TYPE_SCROLL_INSENSITIVE;

22:What is the default ResultSet type in JDBC application and How it is possible to create a specific type of ResultSet object?

- The default ResultSet type in the jdbc applications is Read only and forward only.
- In jdbc applications we are able to specify the following types of the ResultSet combination to any particular ResultSet.

  - read-only, forward only
  - read-only, scroll sensitive
  - read-only, scroll insensitive
  - updatable, forward only
  - updatable, scroll sensitive
  - updatable, scroll insensitive

- if we want to specity a particular type to the ResultSet object then we should use either of the above constants combinationas a parameter to createStatement() method, for this we will use the following method.

public Statement createStatement(int forward / ScrollSensitive / ScrollInsensitive, int readonly / updatable)

Eg: Statement st = con. createSensitive(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
ResultSet rs = con.executeQuery(….);

23:How to iterate the data from Scrollable ResultSet objuect in both forward and backword direction?

- to iterate the data in forward direction from a ResultSet object we will use the following 2 methods.

public Boolean next()
public xxx getXxx(int fieldno.)
Where xxx may be byte, short, char, int, long, float, double.

- To iterate the data in backward direction from Scrollable ResultSet object we will use the following 2 methods.

public Boolean previous()
public xxx getXxx(int fieldno)
Where previous() is a Boolean method, which can be used to check whether the

previous record is available or not, if it is available then cursor will be moved to previous record position.

The following example demonstrates how to iterate the data in both forward and backward direction from the ResultSet object

```
import java.sql.*;
public class ScrollResEx
{
	public static void main(String[] args)throws Exception
	{
		Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
Connection con = DriverManager.getConnection(“jdbc:odbc:nag”,”system”,”durga”);
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSEITIVE,ResultSet.CONCUR_UPDATABLE
);
ResultSet rs = st.executeQuery(“select * from emp1”);
System.out.println(“data in forward direction”);
System.out.println(“ENO       ENAME       ESAL       EADDR”);
System.out.println(“**********************************”);
While(rs.next())
{
System.out.println(rs.getInt(1)+”    ”+rs.getString(2)+”    ”+rs.getFloat(3)+”
”+rs.getString(4));

}
System.in.read();
System.out.println(“data in backward direction”);
System.out.println(“ENO       ENAME       ESAL       EADDR”);
System.out.println(“***********************************”);
While(rs.previous())
{
System.out.println(rs.getInt(1)+”    ”+rs.getString(2)+”    ”+rs.getFloat(3)+”
”+rs.getString(4));

}
}
}
```

24: how to generate ScrollSensitive Result Set and how to reflect the later updations from database automaticallyto the ResultSet object?

```
import java.sql.*;
public class Test
{
	Public static void main(String[] args)throws Exception
```

```
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSEITIVE,ResultSet.CONCUR_UPDATABLE
);
ResultSet rs = st.executeQuery("select * from emp1");
rs.next();
System.out.println("old salary emp111……."+rs.getFloat(3));
System.in.read();//application is in pause, perform database updations
Rs.refreshRow();
System.out.println("new salary of emp111…….."+rs.getFloat(3));
}
}
```

Where refreshRow() is a method from Scrollable ResultSet object, which can be used to refresh the current row in the ResultSet object to allow the later updations from database. Prototype of this method is

public void refreshRow()

25: How to insert records into Database throws Updatable ResultSet?

If we want to insert a new record on to the database through Updatable ResultSet object, we will use the following steps.

Step1: Get the Updatable ResultSet object with fetched data.

Step2: Move ResultSet cursor to the end of the ResultSet object, where we need to take a buffer to hold new records data temporarily, for this we use the following method from updatable ResultSet object.

public void moveToInsertRow()

Step3: Insert new records data on to the buffer temporarily at Updatable ResultSet object for this we will use the following method format.

public void updateXxx(int fieldno,xxx value)

Where xxx may be byte, short, int, char, double, float, long.

Step4: Make the temporary insertion as the permanent insertion in Updatable ResultSet object as will as in database, for this we will use the following method.

public void insertRow()

The following example demonstrates how to insert no. of records onto the database through Updatable ResultSet objects.

```
import java.sql.*;
import java.io.*;
public class UpdateResEx
{
        public static void main(String[] args)throws Exception
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSEITIVE,ResultSet.CONCUR_UPDATABLE
);
ResultSet rs = st.executeQuery("select * from emp1");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
rs.moveToInsertRow();
                while(true)
                {
                        System.out.println("enter employee number");
                        int eno = Integer.parseInt(br.readLine());
                        System.out.println("enter employee name");
                        String ename = br.readLine();
                        System.out.println("enter employee salary");
                        float esal = Float.parseFloat(br.readLine());
                        System.out.println("enter employee address");
                        String eaddr = br.readLine();
                        rs.updateInt(1,eno);
                        rs.updateString(2,ename);
                        rs.updateFloat(3,esal);
                        rs.updateString(4,eaddr);
                        rs.insertRow();
                        System.out.println("record successfully inserted");
                        System.out.println("one more record[y/n]);
                        String option = br.readLine();
                        if(option.equals("n"))
                                break;
}
}
```

26: How to perform updations on Database throws Updatable ResultSet?

By using Updatable ResulSet object we are able to perform some updations on to the database. To perform updations on to thedatabase through Updatable ResultSet object we will use the following steps.

Step1: Get the Updatable ResultSet objectd with the fetched data.

Step2: Move ResultSet cursor to a record where we want to perform updations, for this we will use the following method.

    public void absolute(int recordno.)

Step3: Perform Temporary updations on to the particular record, for this we will use following method.

    public void updateXxx(int fieldno,xxx value)

Step4: Make the temporary updation as a parmenent updation on to the Updatable ResultSet object as well as to the database. For this we will use the following method.

    public void updateRow()

The following example demonstrates how to perform updations on to the database through Updatable ResultSet object.

```
import java.sql.*;
public class UpdateResEx1
{
          public static void main(String[] args)throws Exception
          {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSEITIVE,ResultSet.CONCUR_UPDATABLE
);
ResultSet rs = st.executeQuery("select * from emp1");
rs.absolute(3);
float newsal = rs.getFloat(3)+500;
rs.updateFloat(3,newsal);
rs.updateRow();
                    }
           }
```

27:what is meant by ResultSetMetaData ?How to get The ResultSet metadata of a ResultSet object?

Data about the data is called as Metadata. Similarily data about the data available in ResultSet object called as "ResultSet Metadata".

- ResultSet Metadata includes the number of columns of a table in ResultSet object, all the column names, column datatypes and the column display sizes.
- To get the ResultSet Metadata object we will use the following method from ResultSet object.

public ResultSetMetaData getMetaData()

- To get the number of columns available in ResultSet object we will use the following method from ResultSetMetaData object.

public int getColumnCount()

- To get the name of a particular column, we will use the following method.

public String getColumnName(int fieldno)

- To get the column datatype of a particular column, we will use the following method

public String getColumnTypeName(int fieldno)

- To get the column display size of a particular column we will use the following method.

public int getColumnDisplaySize(int fieldno)

The following example demonstrates how to get ResultSetMetaData from a ResultSet object

```
import java.sql.*;
public class ResultSetMD
{
          public static void main(String[] args)throws Exception
          {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from emp1");
ResultSetMetaData rsmd = rs.getMetaData();
int count = rsmd.getColumnCount();
System.out.println("number of columns......"+count);
for(int i=1;i<=count;i++)
{
System.out.println(rsmd.getColumnName(i)+"  "+rsmd.getColumnTypeName(i)+"
"+rsmd.getColumnDisplaySize(i));
System.out.println()
                              }
                    }
          }
```

28: how to display the data with the respective field names

```
import java.sql.*;
public class RSMD1
{
          public static void main(String[] args)throws Exception
          {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_SENSEITIVE,ResultSet.CONCUR_UPDATABLE
);
ResultSet rs = st.executeQuery("select * from emp1");
                              ResultSetMetaData rsmd = rs.getMetaData();
System.out.println(rsmd.getColumnName(1)+"     "+rsmd.getColumnName(2)+"
"+rsmd.getColumnName(3)+"      "+rsmd.getColumnName(4));
System.out.println("******************************");
while(rs.next())
{
System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"    "rs.getFloat(3)+"
"+rs.getString(4));
                              }
                    }
          }
```

29: What are the differences between Statement and PreparedStatement?
(or)
Tell me the situations where we should go for PreparedStatement over Statement object.
Ans:

- When we have a requirement to execute same kind of sql query in the next sequence then we should go for PreparedStatement over Statement object.
- For the above requirement if we use Statement object, every time execution of the same sql query DBE must perform query tokenization, query parsing, query optimization and query execution.
- This approach will increase burden to the DBE. To reduce burden to the DBE we should go for an alternative. That is PreparedStatement over Statement object.
- For the same requirement if we use PreparedStatement object then for our complete requirement DBE will go for only one time query parsing (tokenization, parsing, optimization and execution);

If we want to use PreparedStatement object for the above requirement then
we will use following steps.
Step1: Prepare PrepareStatement object by providing generalized sql query format with the required number of parameters, for this we will use the following method from Statement object.

```
          public PreparedStatement prepareStatement(String  sqlqueryformat)
```

```
Eg:  PreparedStatement pst = con.prepareStatement("insert into emp1
values(?,?,?,?)");
```

When JVM encounters above instruction jvm will pickup specified generalized sql query format and send to the DBE, here DBE will process query format only one time and prepare a Buffer with the specified parameters, called as "query plan". As a result PreparedStatement object will be created with the parameters at java side.

Step2: Set the values to parameters in PreparedStatement object. After getting PreparedStatement object with parameters, we need to set some values to perform an operation, for this we will use the following method.

public void setXxx(int parano,xxx value)

where xxx may be byte, short, char, int, long, float, double.

Eg: pst.setInt(1,111);
pst.setString(2,"abc");

When JVM encounters the above method then jvm will set the specified values to the specified parameters at the PreparedStatement object, intern that parameter values could be reflected to query plan.

Step3: Given an intimation to DBE to perform the respective operation. After setting the values to the parameters we should give an intimation to the DBE explicitly pickup the values from query plan and perform the operation specified in generalized sql query format, for this we will use the following methods.

- If the generalized sql query belongs to selection group then we will use following method from PreparedStatement object

public ResultSet executeQuery(…)

- If the generalized sql query belongs to updation group then we will use the following method.

public int executeUpdate(…)

30: Hhow to insert number of records into a table through Prepared Statement object.

```
import java.sql.*;
import java.io.*;
public class PreparedInsertEx
{
          public static void main(String[] args)throws Exception
          {
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
PreparedStatement pst= con.prepareStatement("insert into emp1 values(?,?,?,?)");
BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
while(true)
{
;                 }
          }
```

31: how to update the database through PreparedStatement object.

```
import java.sql.*;
public class PreparedUpdateEx
```

```
{
        public static void main(String[] args)throws Exception
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
PreparedStatement pst = con.prepareStatement("update emp1 set esal = esal+? Where
esal<?");
Pst.setInt(1,500);
Pst.setFloat(2,10000.0f);
Int count = pst.executeUpdate();
System.out.println("no. of records updated:"+count);
                }
        }
```

*32:*how to fetch the data from database through PreparedStatement object.

```
import java.sql.*;
public class UpdateResEx
{
        public static void main(String[] args)throws Exception
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
PreparedStatement pst = con.prepareStatement("select * from emp1 where esal<=?");
Pst.setFloat(1,10000.0f);
ResultSet rs = pst.executeQuery();
System.out.println("ENO     ENAME     ESAL     EADDR");
System.out.println("******************************");
While(rs.next())
{
System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"     "+rs.getFloat(3)+"
"+rs.getString(4));
                        }
                }
        }
```

33:What is meant by Transaction? How it is possible to maintain Transactions in JDBC applications?

- Transaction is nothing but an unit of work performed by the applications.
- Every transaction should have the following properties.
- Atomicity
- Consistency
- Isolation
- Durability

- Where atomicity is nothing but perform all the operations or not to perform all the operations in a transaction. That is every transaction must be in either success state or failure state.
- As part of the jdbc applications when we establish a connection automatically the connection should have a default nature called as "auto commit".
- Auto commit in the sense when we send an sql query to the connection then connection will carry that to the DBE and make the DBE to execute provided sql query and store the results on the database permanently.
- The connections default auto commit nature violates the transactions atomicity property.
- To preserve transactions atomicity property we should change the connections auto commit nature to non-auto commit nature, for this we will use the following method.

```
Public void setAutoCommit(Boolean b)
Where b=true    connection is in auto commit
And b=false     connection not in auto commit.
```

- If we use connections non auto commit nature in our jdbc applications then we must use either commit or rollback operations explicitily as part of the transactions.

```
Public void commit()
Public void rollback()
```

The following example demonstrates how to maintain the transactions with atomicity property in the jdbc applications.

```
import java.sql.*;
public class TransactionEx
{
     public static void main(String[] args)throws Exception
     {
               Connection con = null;
               try
               {
                         Class.forName("sun.jdbc.odbd.JdbcOdbcDriver");
Con = DriverManager.getConnection("jdbc:odbc:nag","system","durga");
con.setAutoCommit("false");
Statement st = con.createStatement();
st.executeUpdate("insert into emp1 values(888,'fff',8000,'hhh')");
st.executeUpdate("update emp1 set esal = esal-500 where esal>= 'abc' ");
st.executeUpdate("delete emp1 where esal<7000");
con.commit();
                    }
                    catch(Exception e)
                    {
                              con.rollback();
```

```
                        System.out.println(e);
                }
        }
    }
```

34:What is meant by SavePoint?How to use Savepoints in JDBC applications?

- Save point is a concept introduced by jdbc 3.0 which can be used to block a set of instructions execution in the transactions committing operation.
- To set a save point we will use the following method.

public SavePoint setSavePoint()

- To block a set of sql queries execution prior to the save point we will use the following method.

public void rollback(savepoint s)

- To release a savepoint we will use the following method

public void releaseSavePoint();

- SavePoint concept could not be supported be type1 driver, it could be supported by type4 driver.
- Even type 4 driver is supporting up to setSavePoint() and rollback() , not releaseSavepoint();

```
    Eg:
    import java.sql.*;
public class SavePointEx
{
    public static void main(String[] args)throws Exception
    {
            Connection con = null;
            try
            {
                    Class.forName("oracle.jdbc.driver.OracleDriver");
con =
DriverManager.getConnection("jdbc:oracle:thin:@locajhost:1521:xe","system","durga");
con.setAutoCommit("false");
Statement st = con.createStatement();
st.executeUpdate("insert into emp1 values(111,'fff',8000,'hhh')");
savepoint sp= con.Savepoint();
st.executeUpdate("insert into emp1 values(222,'ggg',7000,'iii') ");
con.rollback(sp);
st.executeUpdate("insert into emp1 values(333,'hhh',9000,'jjj')");
con.commit();
```

```
                }
                catch(Exception e)
                {
                        con.rollback();
                        System.out.println(e);
                }
        }
    }
```

JSP

1. **What is JSP ? Describe its concept.**
Java Server Pages (JSP) is a server side component for the generation of dynamic information as the response. Best suitable to implement view components (presentation layer components). It is part of SUN's J2EE platform.

2 . Explain the benefits of JSP?
These are some of the benefits due to the usage of JSP they are:
Portability, reusability and logic components of the language can be used across various platforms.
Memory and exception management.
Has wide range of API which increases the output functionality.
Low maintenance and easy deployment.
Robust performance on multiple requests.

3. Is JSP technology extensible?
Yes, it is. JSP technology is extensible through the development of custom actions, or tags, which are encapsulated in tag libraries.

4 .Can we implement an interface in a JSP?

No
5 What are the advantages of JSP over Servlet?

1. Best suitable for view components
2. we can separate presentation and business logic
3. The JSP author not required to have strong java knowledge
4. If we are performing any changes to the JSP, then not required to recompile and reload explicitly
5. We can reduce development time.

6. Differences between Servlets and JSP?

Servlets	**JSP**

1. **Best suitable for processing logic**	**1. Best suitable for presentation logic**
2. we cannot separate business and presentation logic	**2. Separation of presentation and business logic is possible**
3. Servlet developer should have strong knowledge in Java	**3.JSP author is not required to have strong knowledge in Java**
4. For source code changes ,we have to perform explicitly compilation	**4. For source code changes ,it is not required to perform explicit compilation**
5. Relatively development time is more	**5. Relatively development time is less**

7 . Explain the differences between ASP and JSP?

The big difference between both of these technologies lies with the design of the software. JSP technology is server and platform independent whereas ASP relies primarily on Microsoft technologies.

8 . Can I stop JSP execution while in the midst of processing a request?
Yes. Preemptive termination of request processing on an error condition is a good way to maximize the throughput of a high-volume JSP engine. The trick (assuming Java is your scripting language) is to use the return statement when we want to terminate further processing.

9. How to Protect JSPs from direct access ?

If the JSP is secured resource then we can place inside WEB-INF folder so that end user is not allowed to access directly by the name. We can provide the url pattern by configuring in web.xml

```
<web-app>

  <servlet>
      <servlet-name>Demo JSP</servlet-name>
       <jsp-file>/WEB-INF/test.jsp</jsp-file>
  <sevlet>
     <servlet-mapping>
        <servlet-name>Demo JSP</servlet-name>
        <url-pattern>/test</url-pattern>
     </servlet-mapping>

..
</web-app>
```

10. Explain JSP API ?

The JSP API contains only one package : javax.servlet.jsp
It contains the following 2 interfaces:

1. **JspPage**:

   This interface defines the two life cycle methods jspInit() and jspDestroy().

1. **HttpJspPage**:

This interface defines only one life cyle method _jspService() method.

Every generated servlet for the jsps should implement either JspPage or HttpJspPage interface either directly or indirectly.

11. What are the lifecycle phases of a JSP?

Life cycle of JSP contains the following phases:

1. **Page translation:** -converting from .jsp file to .java file
2. **Page compilation**: converting .java to .class file
3. **Page loading** : This class file is loaded.
4. **Create an instance** :- Instance of servlet is created
5. **jspInit()** method is called
6. **_jspService()** is called to handle service calls
7. **jspDestroy()** is called to destroy it when the servlet is not required.

12. Explain the life-cycle mehtods in JSP?

The **jspInit()**- The container calls the jspInit() to initialize te servlet instance.It is called before any other method, and is called only once for a servlet instance.
The **_jspservice()**- The container calls the _jspservice() for each request, passing it the request and the response objects.
The **jspDestroy()**- The container calls this when it decides take the instance out of service. It is the last method called n the servlet instance.

13. Difference between _jspService() and other life cycle methods.

JSP contains three life cycle methods namely jspInit( ), _jspService() and jspDestroy(). In these, jspInit() and jspDestroy() can be overridden and we cannot override _jspService().

Webcontainer always generate _jspService() method with JSP content. If we are writing _jspService() method , then generated servlet contains 2 _jspService() methods which will cause compile time error.

To show this difference _jspService() method is prefixed with '_' by the JSP container and the other two methods jspInit() and jspDestroy() has no special prefixes.

14 What is the jspInit() method?
The jspInit() method of the javax.servlet.jsp.JspPage interface is similar to the init() method of servlets. This method is invoked bythe container only once when a JSP page is initialized.

It can be overridden by a page author to initialize resources such asdatabase and network connections, and to allow a JSP page to read persistent configuration data.

15**. What is the _jspService() method?**
SThe _jspService() method of the javax.servlet.jsp.HttpJspPage interface is invoked every time a new request comes to a JSP page. This method takes the HttpServletRequest and HttpServletResponse objects as its arguments. A page author cannot override this method, as its implementation is provided by the container.

16**. What is the jspDestroy() method?**
The jspDestroy() method of the javax.servlet.jsp.JspPage interface is invoked by the container when a JSP page is about to be destroyed. This method is similar to the destroy() method of servlets. It can be overridden by a page author to perform any cleanup operation such as closing a database connection.

17**. What JSP lifecycle methods can I override?**
We can override jspInit() and jspDestroy() methods but we cannot override _jspService() method.

18**. How can I override the jspInit() and jspDestroy() methods within a JSP page?**
By using JSP declation tag

```
<%!
      public void jspInit() {
            . . .
      }
%>
<%!
      public void jspDestroy() {
            . . .
      }
%>
```

19 . Explain about translation and execution of Java Server pages?

A java server page is executed within a Java container. A Java container converts a Java file into a servlet. Conversion happens only once when the application is deployed onto the web server. During the process of compilation Java compiler checks for modifications if any modifications are present it would modify and then execute it.

20 . Why is _jspService() method starting with an '_' while other life cycle methods do not?

jspService() method will be written by the container hence any methods which are not to be overridden by the end user are typically written starting with an ''. This is the reason why we don't override _jspService() method in any JSP page.

21. How to pre-compile JSP?

Add **jsp_precompile** as a request parameter and send a request to the JSP file. This will make the jsp pre-compile.

http://localhost:8080/jsp1/test.jsp?jsp_precompile=true

It causes excution of JSP life cycle until jspInit() method without executing _jspService() method.

22. The benefits of pre-compiling a JSP page?

It removes the start-up lag that occurs when a container must translate a JSP page upon receipt of the first request.

23.**How many JSP scripting elements and explain them?**

Inside JSP four types of scripting elements are allowed.

1. **Scriptlet <% any java code %>**
   Can be used to place java code.

2. declarative <%! Java declaration %>
   Can be used to declare class level variables and methods

3. expression: <%= java expression %>
   To print java expressions in the JSP

4. comment <%-- jsp comment --%>

24. What is a Scriptlet?

JSP scriptlet can be used to place java code.

Syntax:

```
<%

   Any java code

%>
```

The java code present in the scriptlet will be placed directly inside _jspService() method .

25. What is a JSP declarative?
JSP declarations are used to declare class variables and methods (both instance and static) in a JSP page. These declations will be placed directly at class level in the generated servlet and these are available to the entire JSP.

Syntax:

<%! This is my declarative %>

Eg: <%! int j = 10; %>

26. How can I declare methods within my JSP page?
We can declare methods by using JSP declarative tag.

```
<%!
public int add(inti,intj){
return i+j;
}
%>
```

27. What is the difference b/w variable declared inside a declaration and variable declared in scriplet ?

Variable declared inside declaration part is treated as a instance variable and will be placed directly at class level in the generated servlet.

<%! int k = 10; %>

Variable declared in a scriptlet will be placed inside _jspService() method of generated servlet.It acts as local variable.

```
<%
  int k = 10;

%>
```

What is a Expression?

JSP Expression can be used to print expression to the JSP.

Syntax:

<%= java expression %>

Eg: <%= new java.util.Date() %>

The expression in expression tag should not ends with semi-colon

The expression value will become argument to the out.pritln() method in the generated servlet

28.**What are the three kinds of comments in JSP and what's the difference between them?**

Three types of comments are allowed in JSP

1. **JSP Comment:**

<%-- this is jsp comment --%>

This is also known as hidden comment and it is visible only in the JSP and in rest of phases of JSP life cycle it is not visible.

1. **HTML Comment:**

<!-- this is HTMl comment -- >

This is also known as template text comment or output comment. It is visible in all phases of JSP including source code of generated response.

1. **Java Comments.**

With in the script lets we can use even java comments .

```
<%
 // single line java comment
/* this is multiline comment */

%>
```

This type of comments also known as scripting comments and these are visible in the generated servlet also.

29. What is output comment?
The comment which is visible in the source of the response is called output comment.

<!-- this is HTMl comment -- >

30. What is a Hidden Comment?

<%-- this is jsp comment --%>

This is also known as JSP comment and it is visible only in the JSP and in rest of phases of JSP life cycle it is not visible.

31. How is scripting disabled?

Scripting is disabled by setting the scripting-invalid element of the deployment descriptor to true. It is a subelement of jsp-property-group. Its valid values are true and false. The syntax for disabling scripting is as follows:

```
<jsp-property-group>
   <url-pattern>*.jsp</url-pattern>
   <scripting-invalid>true</scripting-invalid>
</jsp-property-group>
```

32. What are the JSP implicit objects?

Implicit objects are by default available to the JSP. Being JSP author we can use these and not required to create it explicitly.

1. request
2. response
3. pageContext
4. session
5. application
6. out
7. config
8. page
9. exception

33. How does JSP handle run-time exceptions?

You can use the **errorPage** attribute of the page directive to have uncaught run-time exceptions automatically forwarded to an error processing page.

For example:
<%@ page errorPage="error.jsp" %> redirects the browser to the JSP page error.jsp if an uncaught exception is encountered during request processing.

Within error.jsp, if you indicate that it is an error-processing page, via the directive:

<%@ page isErrorPage="true" %>

In the error pages we can access exception implicit object.

34. How can I implement a thread-safe JSP page? What are the advantages and Disadvantages of using it?

You can make your JSPs thread-safe by having them implement the SingleThreadModel interface. This is done by adding the directive in the JSP.

<%@ page isThreadSafe="false" %>

The generated servlet can handle only one client request at time so that we can make JSP as thread safe. We can overcome data inconsistency problems by this approach.

The main limitation is it may affect the performance of the system.

35. What is the difference between ServletContext and PageContext?

ServletContext: Gives the information about the container and it represents an application.
PageContext: Gives the information about the Request and it can provide all other implicit JSP objects .

36 . Is there a way to reference the "this" variable within a JSP page?

Yes, there is. The page implicit object is equivalent to "this", and returns a reference to the generated servlet.

37 . Can you make use of a ServletOutputStream object from within a JSP page?

Yes . By using getOutputStream() method on response implicit object we can get it.

38 .What is the page directive is used to prevent a JSP page from automatically creating a session?

session object is by default available to the JSP. We can make it unavailable by using page directive as follows.
<%@ page session="false">
39. What's a better approach for enabling thread-safe servlets and JSPs? SingleThreadModel Interface or Synchronization?

Synchronized keyword is recommended to use to get thread-safety.

40. What are various attributes Of Page Directive ?

Page directive contains the following 13 attributes.

1. language
2. extends
3. import
4. session
5. isThreadSafe
6. info
7. errorPage

8. isError page
9. contentType
10. isELIgnored
11. buffer
12. autoFlush
13. pageEncoding

41 . Explain about autoflush?

This command is used to autoflush the contents. If a value of true is used it indicates to flush the buffer whenever it is full. In case of false it indicates that an exception should be thrown whenever the buffer is full. If you are trying to access the page at the time of conversion of a JSP into servlet will result in error.

42. How do you restrict page errors display in the JSP page?

You first set "errorPage" attribute of PAGE directive to the name of the error page (ie errorPage="error.jsp")in your jsp page .
Then in the error.jsp page set "isErrorpage=TRUE".
When an error occur in your jsp page, then the control will be automatically forward to error page.

43. What are the different scopes available fos JSPs ?

There are four types of scopes are allowed in the JSP.

1. **page** - with in the same page
2. **request** - after forward or include also you will get the request scope data.
3. **session** - after senRedirect also you will get the session scope data. All data stored in session is available to end user till session closed or browser closed.
4. **application** - Data will be available throughout the application. One user can store data in application scope and other can get the data from application scope.

44. when do use application scope?

If we want to make our data available to the entire application then we have to use application scope.

45. What are the different scope valiues for the <jsp:useBean>?

The different scope values for <jsp:useBean> are

1. page
2. request
3.session
4.application

46. How do I use a scriptlet to initialize a newly instantiated bean?

jsp:useBean action may optionally have a body. If the body is specified, its contents will be automatically invoked when the specified bean is instantiated. Typically, the body will contain scriptlets or jsp:setProperty tags to initialize the newly instantiated bean, although you are not restricted to using those alone.

The following example shows the “today” property of the Foo bean initialized to the current date when it is instantiated. Note that here, we make use of a JSP expression within the jsp:setProperty action.

```
<jsp:useBean id="foo" class="com.Bar.Foo" >

<jsp:setProperty name="foo" property="x"
value="<%=java.text.DateFormat.getDateInstance().format(new java.util.Date()) %>" / >

<%-- scriptlets calling bean setter methods go here --%>

</jsp:useBean >
```

47 . Can a JSP page instantiate a serialized bean?

No problem! The use Bean action specifies the beanName attribute, which can be used for indicating a serialized bean.

For example:
A couple of important points to note. Although you would have to name your serialized file "filename.ser", you only indicate "filename" as the value for the beanName attribute. Also, you will have to place your serialized file within the WEB-INF/jspbeans directory for it to be located by the JSP engine.

48.**How do we include static files within a jsp page ?**

We can include static files in JSP by using include directive (static include)

```
<%@ include file=”header.jsp” %>
```

The content of the header.jsp will be included in the current jsp at translation time. Hence this inclusion is also known as static include.

49.**In JSPs how many ways are possible to perform inclusion?**

In JSP, we can perform inclusion in the following ways.

1. **By include directive:**

```
<%@ include file="header.jsp" %>
```

The content of the header.jsp will be included in the current jsp at translation time. Hence this inclusion is also known as static include.

1. **By include action:**

```
<jsp:include page="header.jsp" />
```

The response of the jsp will be included in the current page response at request processing time(run time) hence it is also known as dynamic include.

1. **by using pageContext implicit object**

```
<%

  pageContext.include("/header.jsp");

%>
```

This inclusion also happened at request processing time(run time).

1. **by using RequestDispatcher object**

```
 <%
 RequestDispatcher rd = request.getRequestDispatcher("/header.jsp");
  Rd.incliude(request,response);

%>
```

50.**In which situation we can use static include and dynamic include in JSPs ?**

If the target resource (included resource) won't change frequently, then it is recommended to use static include.

```
<%@ include file="header.jsp" %>
```

If the target resource(Included page) will change frequently , then it is recommended to use dynamic include.

```
< jsp:include page="header.jsp" />
```

Multithreading

Q1. What is Multitasking?
Ans. Executing several task simultaneously is called multitasking.

Q2. What is the difference between process-based and Thread-based Multitasking?
Ans.**Process-based multitasking:-** Executing several task simultaneously where each task is a separate independent process such type of multitasking is called process based Multitasking. Example:-While typing a program in the editor we can listen MP3 audio songs. At the same time we download a file from the net. all these task are executing simultaneously and each task is a separate independent program. hence it is process based multitasking. It is best suitable at operating system level. **Thread-based multitasking:-** Executing several task simultaneously where each task is a separate independent part of the same program is called Thread-based multitasking. and every independent part is called a thread. This type of multitasking is best suitable at programmatic level.

Q3. What is Multithreading and explain its application areas?
Ans. Executing several thread simultaneously where each thread is a separate independent part of the same program is called multithreading. Java language provides inbuilt support for multithreading by defining a reach library, classes and interfaces like Thread, ThreadGroup, Runnable etc. The main important application area of multithreading are video games implementation, animation development, multimedia graphics etc.

Q4.What is advantage of Multithreading?
Ans. The main advantage of multithreading is reduces response time and improves performance of the system.

Q5. When compared with C++ what is the advantage in java with respect to Multithreading?
Ans.Java language provides inbuilt support for multithreading by defining a reach library, classes and interfaces like Thread, ThreadGroup, Runnable etc. But in c++ there is no inbuilt support for multithreading.
Q6. In how many ways we can define a Thread? Among extending Thread and implementing Runnable which is recommended?
Ans. We can define a Thread in the following two ways:

1. by extending Thread class or
2. by implementing Runnable interface.

Among the two ways of defining a thread implementing Runnable mechanism is always recommended. In the first approach as ourThread class already extending Thread there is no chance of extending any other. Hence, we missing the key benefit of oops(inheritance properties).

Q7. What is the difference between t.start() and t.run() method?

Ans. In the case of t.start() method, a new thread will be created which is responsible for the execution of run() method.
But in the case of t.run() method no new thread will be created main thread

executes run() method just like a normal method call.

Q8. Explain about Thread Scheduler?

Ans. If multiple threads are waiting for getting the chance for executing then which thread will get chance first decided by Thread Scheduler. It is the part of JVM and its behavior is vendor dependent and we can't expect exact output.Whenever the situation comes to multithreading the guarantee behavior is very- very low.

Q9. If we are not overriding run() method what will happened?
Ans.If we are not overriding run() method then Thread class run() method will executed which has empty implementation and hence we will not get any output.

Q10.Is overloading of run() method is possible?
Ans.Yes, we can overload run() method but Thread class start() method always invokes no-argument run() method only. The other run() method we have to call explicitly then only will be executed.

Q11.Is it possible to override start() method?
Ans. Yes it is possible. But not recommended.

Q12.If we are overriding start() method then what will happen?
Ans. It we are overriding start() method then our own start() method will be executed just like a normal method call. In this case no new Thread will be created.

Q13. Explain life cycle of a Thread?

Ans. Once we create a Thread object then the Thread is said to be in New/Born state once we call t.start() method now the Thread will be entered into ready/Runnable state that is Thread is ready to execute. If Thread Scheduler allocates CPU now the Thread will entered into the Running state and start execution of run() method. After completing run() method the Thread entered into Dead State.

Q14. What is the importance of Thread class start() method?
Ans. Start() method present in Thread class performing all low level joining formalities for the newly created thread like registering thread with Thread Scheduler etc and then start() method invoking run() method.As the start() method is doing all low level mandatory activities, Programmer has to concentrate only on run() method to define the job. Hence, start() method is a bigassistant to the programmer.Without executing Thread class start() method there is no chance of starting a new Thread.

Q15. After starting a Thread if we trying to restart the same thread once again what will happen?
Ans. After starting a Thread restarting of the same Thread once again is not allowed violation leads to Runtime Exception saying IllegalThreadStateException.

Q16. Explain Thread class constructors?
Ans. There are eight constructors are available in Thread class:
1. Thread t=new Thread();
2. Thread t=new Thread(Runnable r);
3. Thread t=new Thread(String name);
4.Thread t=new Thread(Runnable r, String name);
5.Thread t=new Thread(ThreadGroup g, String name);
6.Thread t=new Thread(ThreadGroup g, Runnable r);
7.Thread t=new Thread(ThreadGroup g, Runnable r, String name);
8.Thread t=new Thread(ThreadGroup g, Runnable r, String name, long stacksize);

Q17. How to get and set name of a Thread?
Ans. For every Thread in java there is a name. To set and get the name of a Thread we can use the following methods. All methods are final.
1.Public final void setName(String name); - To set the name of a Thread
2.Public final String getName(); - To get the name of a Thread.

Q18. What is the range of Thread priority in java?
Ans. The valid range of a Thread priority is 1-10. (1 is least priority and 10 is highest priority)
.
Q19. Who uses Thread priority?
Ans. Thread Scheduler uses priorities while allocating CPU. The Thread which is having highest priority will get chance first for execution.

Q20. What is the default priority of the Thread?
Ans. The default priority only for the main thread is 5 but for all remaining threads default priority will be inheriting from parent to child. Whatever priority parent thread has the same will be inherited to the child thread.

Q21. Once we created a new Thread what about its priority?
Ans. Whatever priority parent thread has the same will be inherited to the new child thread.
Q22. How to get and set priority of a Thread?
Ans. To get and set priority of a Thread, Thread class defines the following two methods:;

1. Public final int
getPriority();

2. Public final void setPriority(int priority);

Q23. If we are trying to set priority of a Thread as 100 what will happen?
Ans. If we are trying to set priority of a Thread as 100 then we will not get any compile time error but at the runtime we will get Runtime exception IllegalArgumentException. Because the valid range of the Thread priority is (1-10) only.

Q24. If two threads having same priority then which thread will get chance first

for execution?
Ans. If two threads having same priority then which thread will get the chance first for execution decided by Thread Scheduler. It is the part of JVM and its behavior is vendor dependent and we can't expect exact output.

Q25. If two threads having different priority then which thread will get chance first for execution?
Ans. If two threads having different priority then the Thread which is having highest priority will get chance first for execution.

Q26 .How we can prevent a thread from execution?
Ans. We can prevent a Thread from executin by using the following methods:

1. Yield()
2. Join()
3. Sleep()

Q27. What is yield() method? Explain its purpose?
Ans. yield() method causes the current executing thread to pause execution and give the chance for waiting thread are same priority. If there is no waiting thread or all the remaining waiting thread have low priority then the same thread will get chance once again for execution. The Thread which is yielded when it will get chance once again for execution depends upon mercy of Thread scheduler.Public static native void yield();

Q28.What is purpose of join() method?
Ans. If a Thread wants to wait until some other Thread completion then we should go for join() method.
Example: if a Thread t1 execute t2.join() ; then t1 will entered into waiting state until t2 Thread completion.

Q29. Is join() method is overloaded?
Ans. Yes join() method is overloaded method.
Public final void join() throws InterruptedException
By using this method thread will wait up to another thread completion
.
Public final void join(long ms) throws InterruptedException
By using this method thread will wail upto sometime what we are passing as a argument in millisecond
Public final void join(long ms, int ns)throws InterruptedException
By using this method thread will wait up to sometime what we are passing as a argument in millisecond and nanosecond.

Q30 What is the purpose of sleep() method?
Ans. If a Thread don't want to perform any operation for a particular amount of time then we should go for sleep() method.Whenever we are using sleep() method compulsory we should handle InterruptedException either by using try-catch or by using throws keyword otherwise we will get compile time error.

Q31. What is synchronized keyword? Explain its advantages and disadvantages.
Ans. Synchronized keyword is applicable for method and blocks only. We can't use for variables and classes.
If a method declared as a synchronized then at a time only one Thread is allow to execute that method on the given object.
The main advantages of synchronized keyword are, we can prevent data inconsistency problems and we can provide Threadsafty.
But the main limitation of synchronized keyword is it increases waiting time of Threads and effect performance of the system. Hence if there is no specific requirement it is not recommended to use synchronized keyword.

Q32.Where we can use synchronized keyword?
Ans. Synchronization concept is applicable whenever multiple Threads are operating on the same object simultaneously. Butwhenever multiple Threads are operating on different objects then there is no impact of synchronization.

Q33. What is object lock? Explain when it is required?
Ans. Every object in java has a unique lock whenever we are using synchronization concept then only lock concept will coming to the picture.
If a Thread wants to execute a synchronized method first it has to get the lock of the object. Once a Thread got the lock then it is allow to execute any synchronized method on that object. After completing synchronized method execution Thread releases the lock automatically.While a Thread executing synchronized method on the given object the remaining Threads are not allow to execute any synchronized method on that object simultaneously. But remaining Threads are allow to execute any non-synchronized method simultaneously. (Lock concept is implemented based on object but not based on method.)
Q34.What is the class level lock? Explain its purpose.
Ans. Every class in java has a unique lock if a Thread wants to execute static synchronized method that Thread has to get class level lock once a Thread got class level lock then only it is allow to execute static synchronized method.
While a Thread executing any static synchronized method then remaining Threads are not allow to execute any static synchronized method of the same class simultaneously. But the remaining Threads are allow to execute the following method simultaneously:

1. Any static non-synchronized method.
2. Synchronized instance methods
3. Non-synchronized instance method.

There is no relationship between object lock and class level lock, both are independent.

Q35. While a thread executing any synchronized method on the given object is it possible to execute remaining synchronized method of the same object simultaneously by any other thread?
Ans. No, it is no possible.
Q36. What is the difference between synchronized method and static synchronized method?
Ans. If a Thread wants to execute a **synchronized method** first it has to get the **lock of the object**. Once a Thread got the lock then it is allow to execute any **synchronized**

method on that object.If a Thread wants to execute **static synchronized method** that Thread has to get **class level lock** once a Thread got class level lock then only it is allow to execute **static synchronized method**.

Q37. What is the advantage of synchronized block over synchronized method?
Ans. If very few lines of the code required synchronization then declaring entire method as the synchronized is not recommended. We have to declare those few lines of the code inside synchronized block. This approach reduces waiting time of the Thread and improves performance of the system.

Q38. What is synchronized statement?
Ans. The Statement which is inside the synchronized area (synchronized method or synchronized block) is called synchronized statement.

Q39. How we can declare synchronized block to get class level lock?
Ans. To get the class level lock we can declare synchronized block as follows:

```
synchronized(Display.class)
  {
   }
```

Q40. How two thread will communicate with each other?
Ans. Two Threads will communicate with each other by using wait(), notify(), notifyAll() methods.

Q41.wait(), notify(), notifyAll() method can available in which class?
Ans. These methods are defined in Object class.

Q42.Why wait(), notify(), notifyAll() method defines in object class instead of Thread class?
Ans. These methods are defined in Object class but not in Thread because Threads are calling this method on the shared object.

Q43.If a waiting thread got notification then it will entered into which state?
Ans. It will entered into another waiting state to get lock.

Q44.In which method threads can release the lock?
Ans. Once a Thread calls wait() method it immediately releases the lock of that object and then entered into waiting state similarly after calling notify() method Thread releases the lock but may not immediately. Except these three methods(wait(), notify(), notifyAll()) method Thread never releases the lock anywhere else.

Q45. Explain wait(), notify(), notifyAll() method uses.
Ans. Two Threads will communicate with each other by using wait(), notify() or notifyAll() methods.
These methods are defined in Object class but not in Thread because Threads are calling this method.

Q46. What is the difference between notify() and notifyAll()?

Ans. To give notification to the single waiting Thread. We use notify() method and to give notification to all waiting thread we use notifyAll() method.

Q47. Once a Thread got the notification then which waiting thread will get chance?
Ans. It is depends on the Thread Scheduler.

Q48. How a thread can interrupt another thread?

Ans. A Thread can interrupt another Thread by using interrupt() method.

Q49. What is DeadLock? Is it possible to resolve DeadLock situation?
Ans. If two Threads are waiting for each other forever such type of situation is called DeadLock.
For the DeadLock, there are no resolution techniques but prevention techniques are available.

Q50. Which keyword causes DeadLock situation?
Ans. Synchronized keyword is the thing to causes of DeadLock. If we are not using properly synchronized keyword the program will entered into DeadLock situation.

Q51. How we can stop a thread explacitly?
Ans. Thread class defines stop() method by using this method we can stop a Thread. But it is deprecated. And hence not recommended to use.

Q52. Explain about suspend() and resume() method?
Ans. A Thread can suspend another Thread by using suspend() method.
A Thread can resume a suspended Thread by using resume() method.

Q53.What is Starvation()? And Explain the difference between Deadlock and Starvation?
Ans. A long waiting Thread is said to be in starvation (because of least priority) but after certain time defiantly it will get the chance for execution. But in the case of Deadlock two Threads will wait for each other forever. It will never get the chance for execution.
Q54. What is race condition?
Ans. Multiple Threads are accessing simultaneously and causing data inconsistency problem is called race condition, we can resolve this by using synchronized keyword.
Q55. What is Daemon Thread? And give an example?
Ans. The Threads which are running in the background are called Daemon Thread.
Example: Garbage collector.
Q56. What is the purpose of a Daemon Thread?
Ans. The main purpose of Daemon Threads is to provide support for non-daemon Threads.

Q57. How we can check Daemon nature of a Thread?
Ans. We can check Daemon nature of a Thread by using **isDaemon()** method.
Q58. Is it possible to change a Daemon nature of a Thread?
Ans. Yes, we can change Daemon nature of a Thread by using **setDaemon()** method.

Q59. Is main thread is Daemon or non-daemon?
Ans. By default main thread is always non-daemon nature.
Q60. Once we created a new thread is it daemon or non-daemon.
Ans. Once we created a new Thread, The Daemon nature will be inheriting from parent to child. If the parent is Daemon the child is also Daemon and if the parent is non-daemon then child is also non-daemon.

Q61. After starting a thread is it possible to change Daemon nature?
Ans. We can change the Daemon nature before starting the Thread only. Once Thread started we are not allow to change Daemon nature otherwise we will get RuntimeException sying IllegalThreadStateException.
Q62. When the Daemon thread will be terminated?
Ans. Once last non-daemon Thread terminates automatically every Daemon Thread will be terminated.
Q63. What is green Thread?
Ans. A green thread refers to a mode of operation for the Java Virtual Machine (JVM) in which all code is executed in a single operating system thread. If the Java program has any concurrent threads, the JVM manages multi-threading internally rather than using other operating system threads.
There is a significant processing overhead for the JVM to keep track of thread states and swap between them, so green thread mode has been deprecated and removed from more recent Java implementations.
Q64.Explain about Thread group?
Ans. Every Java thread is a member of a *thread group*. Thread groups provide a mechanism for collecting multiple threads into a single object and manipulating those threads all at once, rather than individually. For example, you can start or suspend all the threads within a group with a single method call. Java thread groups are implemented by the ThreadGroup api class in the java.lang package.

Q65.What is the Thread Local?
Ans. It's a way for each thread in multi-threaded code to keep its own copy of an instance variable. Generally, instance variable are shared between all threads that use an object; ThreadLocal is a way for each thread to keep its own copy of such a variable. The purpose might be that each thread keeps different data in that variable, or that the developer wants to avoid the overhead of synchronizing access to it.
Q66. In your previous project where you used multithreading concept?

Struts

Q 1. What is MVC?

Model-View-Controller (MVC) is a design pattern put together to help control change. MVC decouples interface from business logic and data.
Model: The model contains the core of the application's functionality. The model enca psulates the state of the application. Sometimes the only functionality it contains is state. It knows nothing about the view or controller.
View: The view provides the presentation of the model. It is the *look* of the application. The view can access the model getters, but it has no knowledge of the setters. In addition, it

knows nothing about the controller. The view should be notified when changes to the model occur.
Controller: The controller reacts to the user input. It creates and sets the model.

Q 2. What is a framework?

Framework is made up of the set of classes which allow us to use a library in a best possible way for a specific requirement.

Q 3. What is Struts framework?

Struts framework is an open-source framework for developing the web applications in Java EE, based on MVC-2 architecture. It uses and extends the Java Servlet API. Struts is robust architecture and can be used for the development of application of any size. Struts framework makes it much easier to design scalable, reliable Web applications with Java. Struts provides its own Controller component and integrates with other technologies to provide the Model and the View. For the Model, Struts can interact with standard data access technologies, like JDBC and EJB, as well as most any third-party packages, like Hibernate, iBATIS, or Object Relational Bridge. For the View, Struts works well with JavaServer Pages, including JSTL and JSF, as well as Velocity Templates, XSLT, and other presentation systems.

Q 4. What is Jakarta Struts Framework?

Jakarta Struts is open source implementation of MVC (Model-View-Controller) pattern for the development of web based applications. Jakarta Struts is robust architecture and can be used for the development of application of any size. Struts framework makes it much easier to design scalable, reliable Web applications with Java.

Q 5. What is ActionServlet?

The class org.apache.struts.action.ActionServlet is the called the ActionServlet. In the the Jakarta Struts Framework this class plays the role of controller. All the requests to the server
goes through the controller. Controller is responsible for handling all the requests.

Q 6. What is role of ActionServlet?

ActionServlet performs the role of Controller:

- Process user requests
- Determine what the user is trying to achieve according to the request
- Pull data from the model (if necessary) to be given to the appropriate view,
- Select the proper view to respond to the user
- Delegates most of this grunt work to Action classes
- Is responsible for initialization and clean-up of resources

Q 7. What is Action Class?

Any java class which extends from org.apache.struts.action.Action is called Action class. The Action is part of the controller. The purpose of Action Class is to translate the HttpServletRequest to the business logic. To use the Action, we need to Subclass and overwrite the execute() method. The ActionServlet (commad) passes the parameterized class to Action Form using the execute() method. There should be no database interactions in the action. The action should receive the request, call business objects (which then handle database, or interface with J2EE, etc) and then determine where to go next. Even better, the business objects could be handed to the action at runtime (IoC style) thus removing any dependencies on the model. The return type of the execute method is ActionForward which is used by the Struts Framework to forward the request to the file as per the value of the returned ActionForward object..

Q 8. Write code of any Action Class?

```
package com.durgasoft;
import javax.servlet.http.*;
import org.apache.struts.action.*;
public class TestAction extends Action
{
public ActionForward execute(ActionMapping mapping, ActionForm form,
 HttpServletRequest request,HttpServletResponse response) throws Exception
  {
    return mapping.findForward("success");
  }
}
```

Q 9. What is ActionForm?

Any java class which extends from org.apache.struts.action.ActionForm is called ActionForm. An ActionForm is also called JavaBean. ActionForm maintains the session state for web application and the ActionForm object is automatically populated on the server side with data entered from a form on the client side.

Q10. What is Struts Validator Framework?

Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used validate the form data on the client browser. Server side validation of form can be accomplished by sub classing your From Bean with **DynaValidatorForm** class.
The Validator framework was developed by David Winterfeldt as third-party add-on to Struts. Now the Validator framework is a part of Jakarta Commons project and it can be used with or without Struts. The Validator framework comes integrated with the Struts Framework and can be used without doing any extra settings.

Q11. How you will display validation fail errors on jsp page?

Following tag displays all the errors:

```
<html:errors/>
```

Q12. What is RequestProcessor?

The controller is responsible for intercepting and translating user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations. The Controller receives the request from the browser, invoke a business operation and coordinating the view to return to the client.The controller is implemented by a java servlet, this servlet is centralized point of control for the web application. In struts framework the controller responsibilities are implemented by several different components like
The ActionServlet Class
The RequestProcessor Class
The Action Class
The ActionServlet extends the **javax.servlet.http.httpServlet** class. The ActionServlet class is not abstract and therefore can be used as a concrete controller by your application. The controller is implemented by the ActionServlet class. All incoming requests are mapped to the central controller in the deployment descriptor as follows.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
 </servlet>
```

All request URIs with the pattern *.do are mapped to this servlet in the deployment descriptor as follows.

```
<servlet-mapping>
      <servlet-name>action</servlet-name>
      <url-pattern>*.do</url-pattern>
<servlet-mapping>
```

A request URI that matches this pattern will have the following form.
http://localhost:8080/mycontext/actionName.do
The preceding mapping is called extension mapping, however, you can also specify path mapping where a pattern ends with /* as shown below.

```
<servlet-mapping>
   <servlet-name>action</servlet-name>
   <url-pattern>/do/*</url-pattern>
   <url-pattern>*.do</url-pattern>
```

A request URI that matches this pattern will have the following form.
http://localhost:8080/mycontext/do/action_Name The class org.apache.struts.action.requestProcessor process the request from the controller. You can sublass the RequestProcessor with your own version and modify how the request is processed.
Once the controller receives a client request, it delegates the handling of the request to a helper class. This helper knows how to execute the business operation associated with the requested action. In the Struts framework this helper class is descended of org.apache.struts.action.Action class. It acts as a bridge between a client-side user action and business operation. The Action class decouples the client request from the business

model. This decoupling allows for more than one-to-one mapping between the user request and an action. The Action class also can perform other functions such as authorization, logging before invoking business operation. the Struts Action class contains several methods, but most important method is the execute() method.
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws Exception

The execute() method is called by the controller when a request is received from a client. The controller creates an instance of the Action class if one doesn?t already exist. The strut framework will create only a single instance of each Action class in your application.

Action are mapped in the struts configuration file and this configuration is loaded into memory at startup and made available to the framework at runtime. Each Action element is represented in memory by an instance of the org.apache.struts.action. ActionMapping class. The ActionMapping object contains a path attribute that is matched against a portion of the URI of the incoming request.

```
<action>
      path= "/somerequest"
      type="com.somepackage.someAction"
      scope="request"
      name="someForm"
      validate="true"
      input="somejsp.jsp"
   <forward name="Success" path="/action/xys" redirect="true"/>
   <forward name="Failure" path="/somejsp.jsp" redirect="true"/>
</action>
```

Once this is done the controller should determine which view to return to the client. The execute method signature in Action class has a return type org.apache. struts.action.ActionForward class. The ActionForward class represents a destination to which the controller may send control once an action has completed. Instead of specifying an actual JSP page in the code, you can declaratively associate as action forward through out the application. The action forward are specified in the configuration file.

```
<action>
      path= "/somerequest"
      type="com.somepackage.someAction"
      scope="request"
      name="someForm"
      validate="true"
      input="somejsp.jsp"
   <forward name="Success" path="/action/xys" redirect="true"/>
   <forward name="Failure" path="/somejsp.jsp" redirect="true"/>
</action>
```

The action forward mappings also can be specified in a global section, independent of any specific action mapping.

```
<global-forwards>
   <forward name="Success" path="/action/somejsp.jsp" />
   <forward name="Failure" path="/someotherjsp.jsp" />
</global-forwards>
```

Q13. How you will handle exceptions in Struts?

In Struts you can handle the exceptions in two ways:
a) Declarative Exception Handling: You can either define global exception handling tags in your struts-config.xml or define the exception handling tags within <action>..</action> tag.

Example:
```
<exception
     key="database.error.duplicate"
     path="/UserExists.jsp"
     type="mybank.account.DuplicateUserException"/>
```
b) Programmatic Exception Handling: Here you can use try{}catch{} block to handle the exception.

Q14. What are the different kinds of actions in Struts?

The different kinds of actions in Struts are:
ForwardAction, IncludeAction, DispatchAction, LookupDispatchAction, SwitchAction

Q15. What is DispatchAction?

The DispatchAction class is used to group related actions into one class. Using this class, you can have a method for each logical action compared than a single execute method. The DispatchAction dispatches to one of the logical actions represented by the methods. It picks a method to invoke based on an incoming request parameter. The value of the incoming parameter is the name of the method that the DispatchAction will invoke.

Q16. How to use DispatchAction?

To use the DispatchAction, follow these steps :

1. Create a class that extends DispatchAction (instead of Action)
2. In a new class, add a method for every function you need to perform on the service – The method has the same signature as the execute() method of an Action class.
3. Do not override execute() method – Because DispatchAction class itself provides execute() method.
4. Add an entry to struts-config.xml

Q17. What is LookupDispatchAction?

The LookupDispatchAction is a subclass of DispatchAction. It does a reverse lookup on the resource bundle to get the key and then gets the method whose name is associated with the key into the Resource Bundle.

Q18. What is the use of LookupDispatchAction?

LookupDispatchAction is useful if the method name in the Action is not driven by its name in the front end, but by the Locale independent key into the resource bundle. Since the key is always the same, the LookupDispatchAction shields your application from the side effects of I18N.

Q19. What is difference between LookupDispatchAction and DispatchAction?

The difference between LookupDispatchAction and DispatchAction is that the actual method that gets called in LookupDispatchAction is based on a lookup of a key value instead of specifying the method name directly.

Q20. What is SwitchAction?

The SwitchAction class provides a means to switch from a resource in one module to another resource in a different module. SwitchAction is useful only if you have multiple modules in your Struts application. The SwitchAction class can be used as is, without extending.

Q21. What if <action> element has <forward> declaration with same name as global forward?

In this case the global forward is not used. Instead the <action> element's <forward> takes precendence.

Q22. What is difference between ActionForm and DynaActionForm?

An ActionForm represents an HTML form that the user interacts with over one or more pages. You will provide properties to hold the state of the form with getters and setters to access them. Whereas, using DynaActionForm there is no need of providing properties to hold the state. Instead these properties and their type are declared in the struts-config.xml.
The DynaActionForm bloats up the Struts config file with the xml based definition. This gets annoying as the Struts Config file grow larger.
The DynaActionForm is not strongly typed as the ActionForm. This means there is no compile time checking for the form fields. Detecting them at runtime is painful and makes you go through redeployment.
ActionForm can be cleanly organized in packages as against the flat organization in the Struts Config file.
ActionForm were designed to act as a Firewall between HTTP and the Action classes, i.e. isolate and encapsulate the HTTP request parameters from direct use in Actions. With DynaActionForm, the property access is no different than using request.get Parameter(..).

- DynaActionForm construction at runtime requires a lot of Java Reflection (Introspection) machinery that can be avoided.

Q23. What is the life cycle of ActionForm?

The lifecycle of ActionForm invoked by the RequestProcessor is as follows:

- Retrieve or Create Form Bean associated with Action
- "Store" FormBean in appropriate scope (request or session)
- Reset the properties of the FormBean
- Populate the properties of the FormBean
- Validate the properties of the FormBean
- Pass FormBean to Action

Q24.What are the important tags of struts-config.xml ?

```
<struts-config>
  <!-- ========== Form Bean Definitions ============ -->
<form-beans>
<form-bean name="login" type=" LoginForm" />
  </form-beans>
  <!-- ========== Global Forward Definitions ========= -->
  <global-forwards>
  </global-forwards>
  <!-- ========== Action Mapping Definitions ======== -->
  <action-mappings>
    <action
        path="/login"
        type="LoginAction" >
        </action>
  </action-mappings>
<!-- ========== Properties Definitions ============ -->
<message-resources parameter="MessageResources" />
<!-- ========== Validator framework Definitions ============ -->
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property
           property="pathnames"
            value="/org/apache/struts/validator/validator-rules.xml,
        /WEB-INF/validation.xml"/>
   </plug-in>
</struts-config>
```

Q25. What are the core classes of the Struts Framework?

A: Core classes of Struts Framework are ActionForm, Action, ActionMapping, Action Forward, ActionServlet etc.

Q26. What is action mappings?

An action mapping is a configuration file entry that, in general, associates an action name with an action. An action mapping can contain a reference to a form bean that the action can use, and can additionally define a list of local forwards that is visible only to this action.

Q27. Describe validate() and reset() methods ?

validate () and reset() methods defined inActionForm class.
validate() : Used to validate properties after they have been populated; Called before FormBean is handed to Action. Returns a collection of ActionMessage as ActionErrors. Following is the method signature for the validate() method.
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request)
reset(): reset() method is called by Struts Framework with each request that uses the defined ActionForm. The purpose of this method is to reset all of the ActionForm's data members prior to the new request values being set.
public void reset() {}

Q28. Give the Details of XML files used in Validator Framework?
The Validator Framework uses two XML configuration files **validator-rules.xml** and **validation.xml**. The **validator-rules.xml**defines the standard validation routines, these are reusable and used in **validation.xml**. to define the form specific validations. The **validation.xml** defines the validations applied to a form bean.

Q29. How you will enable front-end validation based on the xml in validation.xml?
The <html:javascript> tag to allow front-end validation based on the xml in validation.xml. For example the code: <html:javascript formName="logonForm" dynamicJavascript="true" staticJavascript="true" /> generates the client side java script for the form "logonForm" as defined in the validation.xml file. The <html:javascript> when added in the jsp file generates the client site validation script.

Q30. What is the difference between perform() and execute() methods?
perform() method defined in Struts 1.0. but it is was deprecated in the Struts Version 1.1. In Struts 1.x, Action.perform() is the method called by the ActionServlet. This is typically where your business logic resides, or at least the flow control to your JavaBeans and EJBs that handle your business logic. As we already mentioned, to support declarative exception handling, the method signature changed in perform. Now execute just throws Exception. Action.perform() is now deprecated; however, the Struts v1.1 ActionServlet is smart enough to know whether or not it should call perform or execute in the Action, depending on which one is available.

Q31. What are the various Struts tag libraries?
Struts is very rich framework and it provides very good and user friendly way to develop web application forms. Struts provide many tag libraries to ease the development of web applications. These tag libraries are:
* **Bean tag library** - Tags for accessing JavaBeans and their properties.
* **HTML tag library** - Tags to output standard HTML, including forms, text boxes, checkboxes, radio buttons etc..
* **Logic tag library** - Tags for generating conditional output, iteration capabilities and flow management
* **Tiles or Template tag library** - For the application using tiles
* **Nested tag library** - For using the nested beans in the application

Q32. What are the difference between <bean:message> and <bean:write>?
<bean:message>: This tag is used to output locale-specific text (from the properties files) from a MessageResources bundle.

<bean:write>: This tag is used to output property values from a bean. <bean:write> is a commonly used tag which enables the programmers to easily present the data.

Q33. What are difference between ActionErrors and ActionMessage?

ActionMessage: A class that encapsulates messages. Messages can be either global or they are specific to a particular bean property.
Each individual message is described by an ActionMessage object, which contains a message key (to be looked up in an appropriate message resources database), and up to four placeholder arguments used for parametric substitution in the resulting message.

ActionErrors: A class that encapsulates the error messages being reported by the validate() method of an ActionForm. Validation errors are either global to the entire ActionForm bean they are associated with, or they are specific to a particular bean property (and, therefore, a particular input field on the corresponding form).

Q34. What is the use of ForwardAction?
The ForwardAction class is useful when you're trying to integrate Struts into an existing application that uses Servlets to perform business logic functions. You can use this class to take advantage of the Struts controller and its functionality, without having to rewrite the existing Servlets. Use ForwardAction to forward a request to another resource in your application, such as a Servlet that already does business logic processing or even another JSP page. By using this predefined action, you don't have to write your own Action class. You just have to set up the struts-config file properly to use ForwardAction.

Q35. What is IncludeAction?
The IncludeAction class is useful when you want to integrate Struts into an application that uses Servlets. Use the IncludeAction class to include another resource in the response to the request being processed.

Q36. What are the steps need to use DynaActionForm?
Using a DynaActionForm instead of a custom subclass of ActionForm is relatively straightforward. You need to make changes in two places:
In struts-config.xml: change your <form-bean> to be an org.apache.struts.action.Dyna ActionForm instead of some subclass of ActionForm

- <form-bean name="loginForm"
-           type="org.apache.struts.action.DynaActionForm" >
      <form-property name="userName" type="java.lang.String"/>
      <form-property name="password" type="java.lang.String" />
  </form-bean>
- In your Action subclass that uses your form bean:
    - import org.apache.struts.action.DynaActionForm
    - downcast the ActionForm parameter in execute() to a DynaActionForm
    - access the form fields with get(field) rather than getField()

Q.37 In struts what happens if made any changes in actionservlet?

The ActionServlet plays the role of controller wich is responsible for handling the request and selecting the correct Application Module and storing ApplicationConfig and

MessageResource bundle in the request object.
If we modify the ActionServlet the Controller may or may not work what happens that depends on your modification, You have not specify whether you want to create your own custom ActionServlet by extending ActionServlet and overriding the methods in it or what exactly you want to modify.

Spring

1. What is IOC (or Dependency Injection)?
The basic concept of the Inversion of Control pattern (also known as dependency injection) is that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (in the case of the Spring framework, the IOCcontainer) is then responsible for hooking it all up.

i.e., Applying IoC, objects are given their dependencies at creation time by some external entity that coordinates each object in the system. That is, dependencies are injected into objects. So, IoC means an inversion of responsibility with regard to how an object obtains references to collaborating objects.

2. What are the different types of IOC (dependency injection) ?
There are three types of dependency injection:

- **Constructor Injection** (e.g. Pico container, Spring etc): Dependencies are provided as constructor parameters.
- **Setter Injection** (e.g. Spring): Dependencies are assigned through JavaBeans properties (ex: setter methods).
- **Interface Injection** (e.g. Avalon): Injection is done through an interface.

Note: Spring supports only Constructor and Setter Injection

3. What are the benefits of IOC (Dependency Injection)?
Benefits of IOC (Dependency Injection) are as follows:

- Minimizes the amount of code in your application. With IOC containers you do not care about how services are created and how you get references to the ones you need. You can also easily add additional services by adding a new constructor or a setter method with little or no extra configuration.
- Make your application more testable by not requiring any singletons or JNDI lookup mechanisms in your unit test cases. IOC containers make unit testing and switching implementations very easy by manually allowing you to inject your own objects into the object under test.
- Loose coupling is promoted with minimal effort and least intrusive mechanism. The factory design pattern is more intrusive because components or services need to be requested explicitly whereas in IOC the dependency is injected into requesting piece of code. Also some containers promote the design to interfaces not to

implementations design concept by encouraging managed objects to implement a well-defined service interface of your own.

- IOC containers support eager instantiation and lazy loading of services. Containers also provide support for instantiation of managed objects, cyclical dependencies, life cycles management, and dependency resolution between managed objects etc.

4. What is Spring ?

Spring is an open source framework created to address the complexity of enterprise application development. One of the chief advantages of the Spring framework is its layered architecture, which allows you to be selective about which of its components you use while also providing a cohesive framework for J2EE application development.

5. What are the advantages of Spring framework?

The advantages of Spring are as follows:

- Spring has layered architecture. Use what you need and leave you don't need now.
- Spring Enables POJO Programming. There is no behind the scene magic here. POJO programming enables continuous integration and testability.
- Dependency Injection and Inversion of Control Simplifies JDBC
- Open source and no vendor lock-in.

6. What are features of Spring ?

Lightweight:

spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 1MB. And the processing overhead is also very negligible.

Inversion of control (IOC):

Loose coupling is achieved in spring using the technique Inversion of Control. The objects give their dependencies instead of creating or looking for dependent objects.

Aspect oriented (AOP):

Spring supports Aspect oriented programming and enables cohesive development by separating application business logic from system services.

Container:

Spring contains and manages the life cycle and configuration of application objects.

MVC Framework:

Spring comes with MVC web application framework, built on core Spring functionality. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI. But other frameworks can be easily used instead of Spring MVC Framework.

Transaction Management:

Spring framework provides a generic abstraction layer for transaction management. This allowing the developer to add the pluggable transaction managers, and making it easy to demarcate transactions without dealing with low-level issues. Spring's transaction support is not tied to J2EE environments and it can be also used in container less environments.

JDBC Exception Handling:
The JDBC abstraction layer of the Spring offers a meaningful exception hierarchy, which simplifies the error handling strategy. Integration with Hibernate, JDO, and iBATIS: Spring provides best Integration services with Hibernate, JDO and iBATIS
contexts for Web-based applications. As a result, the Spring framework supports integration with Jakarta Struts. The Web module also eases the tasks of handling multi-part requests and binding request parameters to domain objects.

7 . What is web module?

This module is built on the application context module, providing a context that is appropriate for web-based applications. This module also contains support for several web-oriented tasks such as transparently handling multipart requests for file uploads and programmatic binding of request parameters to your business objects. It also contains integration support with *Jakarta Struts*.

8. What are the types of Dependency Injection Spring supports?

Setter Injection:
Setter-based DI is realized by calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

Constructor Injection:
Constructor-based DI is realized by invoking a constructor with a number of arguments, each representing a collaborator.

9. What is Bean Factory ?
A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients.

- BeanFactory is able to create associations between collaborating objects as they are instantiated. This removes the burden of configuration from bean itself and the beans client.
- BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

10. What is Application Context?
A bean factory is fine to simple applications, but to take advantage of the full power of the Spring framework, you may want to move up to Springs more advanced container, the application context. On the surface, an application context is same as a bean factory.Both load bean definitions, wire beans together, and dispense beans upon request. But it also provides:

- A means for resolving text messages, including support for internationalization.
- A generic way to load file resources.
- Events to beans that are registered as listeners.

11. What is the difference between Bean Factory and Application Context ?
On the surface, an application context is same as a bean factory. But application context offers much more..

- Application contexts provide a means for resolving text messages, including support for i18n of those messages.
- Application contexts provide a generic way to load file resources, such as images.
- Application contexts can publish events to beans that are registered as listeners.
- Certain operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context.
- ResourceLoader support: Spring's Resource interface us a flexible generic abstraction for handling low-level resources. An application context itself is a ResourceLoader, Hence provides an application with access to deployment-specific Resource instances.
- MessageSource support: The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable

12. What are the common implementations of the Application Context ?
The three commonly used implementation of 'Application Context' are

- **ClassPathXmlApplicationContext :** It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources. The application context is loaded from the application's classpath by using the code .
  ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");
- **FileSystemXmlApplicationContext :** It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code .
  ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml");
- **XmlWebApplicationContext :** It loads context definition from an XML file contained within a web application.

13. How is a typical spring implementation look like ?
For a typical Spring Application we need the following files:

- An interface that defines the functions.
- An Implementation that contains properties, its setter and getter methods, functions etc.,
- Spring AOP (Aspect Oriented Programming)
- A XML file called Spring configuration file.
- Client program that uses the function.

14. What is the typical Bean life cycle in Spring Bean Factory Container ?
Bean life cycle in Spring Bean Factory Container is as follows:

- The spring container finds the bean's definition from the XML file and instantiates the bean.
- Using the dependency injection, spring populates all of the properties as specified in the bean definition
- If the bean implements the BeanNameAware interface, the factory calls setBeanName() passing the bean's ID.
- If the bean implements the BeanFactoryAware interface, the factory calls setBeanFactory(), passing an instance of itself.
- If there are any BeanPostProcessors associated with the bean, their post-ProcessBeforeInitialization() methods will be called.
- If an init-method is specified for the bean, it will be called.
- Finally, if there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization() methods will be called.

15. What do you mean by Bean wiring ?
The act of creating associations between application components (beans) within the Spring container is reffered to as Bean wiring.

16. What do you mean by Auto Wiring?
The Spring container is able to autowire relationships between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for your bean by inspecting the contents of the BeanFactory. The autowiring functionality has *five modes*.

- no
- byName
- byType
- constructor
- autodirect

17. What is DelegatingVariableResolver?
Spring provides a custom JavaServer Faces VariableResolver implementation that extends the standard Java Server Faces managed beans mechanism which lets you use JSF and Spring together. This variable resolver is called as*DelegatingVariableResolver*

18. How to integrate Java Server Faces (JSF) with Spring?
JSF and Spring do share some of the same features, most noticeably in the area of IOC services. By declaring JSF managed-beans in the faces-config.xml configuration file, you allow the FacesServlet to instantiate that bean at startup. Your JSF pages have access to these beans and all of their properties.We can integrate JSF and Spring in two ways:

- **DelegatingVariableResolver:** Spring comes with a JSF variable resolver that lets you use JSF and Spring together.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
  "http://www.springframework.org/dtd/spring-beans.dtd">

<faces-config>
  <application>
    <variable-resolver>
      org.springframework.web.jsf.DelegatingVariableResolver
    </variable-resolver>
  </application>
</faces-config>
```

The DelegatingVariableResolver will first delegate value lookups to the default resolver of the underlying JSF implementation, and then to Spring's 'business context' WebApplicationContext. This allows one to easily inject dependencies into one's JSF-managed beans.

- FacesContextUtils:custom VariableResolver works well when mapping one's properties to beans in faces-config.xml, but at times one may need to grab a bean explicitly. The FacesContextUtils class makes this easy. It is similar to WebApplicationContextUtils, except that it takes a FacesContext parameter rather than a ServletContext parameter.

```
ApplicationContext ctx =
FacesContextUtils.getWebApplicationContext(FacesContext.getCurrentInstance());
```

19. What is Java Server Faces (JSF) - Spring integration mechanism?
Spring provides a custom JavaServer Faces VariableResolver implementation that extends the standard JavaServer Faces managed beans mechanism. When asked to resolve a variable name, the following algorithm is performed:

- Does a bean with the specified name already exist in some scope (request, session, application)? If so, return it
- Is there a standard JavaServer Faces managed bean definition for this variable name? If so, invoke it in the usual way, and return the bean that was created.
- Is there configuration information for this variable name in the Spring WebApplicationContext for this application? If so, use it to create and configure an instance, and return that instance to the caller.
- If there is no managed bean or Spring definition for this variable name, return null instead.
- BeanFactory also takes part in the life cycle of a bean, making calls to custom initialization and destruction methods.

As a result of this algorithm, you can transparently use either JavaServer Faces or Spring facilities to create beans on demand.

20. What is Significance of JSF- Spring integration ?
Spring - JSF integration is useful when an event handler wishes to explicitly invoke the bean factory to create beans on demand, such as a bean that encapsulates the business logic to be performed when a submit button is pressed.

21. How to integrate your Struts application with Spring?
To integrate your Struts application with Spring, we have two options:

- Configure Spring to manage your Actions as beans, using the ContextLoaderPlugin, and set their dependencies in a Spring context file.
- Subclass Spring's ActionSupport classes and grab your Spring-managed beans explicitly using a getWebApplicationContext() method.

22. What are ORM’s Spring supports ?
Spring supports the following ORM’s :

- Hibernate
- iBatis
- JPA (Java Persistence API)
- TopLink
- JDO (Java Data Objects)
- OJB

23. What are the ways to access Hibernate using Spring ?
There are two approaches to Spring’s Hibernate integration:

- Inversion of Control with a HibernateTemplate and Callback
- Extending HibernateDaoSupport and Applying an AOP Interceptor

24. How to integrate Spring and Hibernate using HibernateDaoSupport?
Spring and Hibernate can integrate using Spring’s SessionFactory called LocalSessionFactory. The integration process is of 3 steps.

- Configure the Hibernate SessionFactory
- Extend your DAO Implementation from HibernateDaoSupport
- Wire in Transaction Support with AOP

25. What are Bean scopes in Spring Framework ?

The Spring Framework supports exactly five scopes (of which three are available only if you are using a web-aware ApplicationContext). The scopes supported are listed below:

Scope	Description
singleton	Scopes a single bean definition to a single object instance per Spring IoC container.
prototype	Scopes a single bean definition to any number of object instances.
request	Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.
session	Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.
global session	Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

26. What is AOP?

Aspect-oriented programming, or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management. The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules.

27. How the AOP used in Spring?

*AOP is used in the Spring Framework:*To provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is declarative transaction management, which builds on the Spring Framework's transaction abstraction.To allow users to implement custom aspects, complementing their use of OOP with AOP.

28. What do you mean by Aspect ?

A modularization of a concern that cuts across multiple objects. Transaction management is a good example of a crosscutting concern in J2EE applications. In Spring AOP, aspects are implemented using regular classes (the schema-based approach) or regular classes annotated with the @Aspect annotation (@AspectJ style).

29. What do you mean by JointPoint?
A point during the execution of a program, such as the execution of a method or the handling of an exception. In Spring AOP, a join point always represents a method execution.

30. What do you mean by Advice?
Action taken by an aspect at a particular join point. Different types of advice include "around," "before" and "after" advice. Many AOP frameworks, including Spring, model an advice as an interceptor, maintaining a chain of interceptors "around" the join point.

31. What are the types of Advice?
Types of advice:

- *Before advice*: Advice that executes before a join point, but which does not have the ability to prevent execution flow proceeding to the join point (unless it throws an exception).
- *After returning advice*: Advice to be executed after a join point completes normally: for example, if a method returns without throwing an exception.
- *After throwing advice*: Advice to be executed if a method exits by throwing an exception.
- *After (finally) advice*: Advice to be executed regardless of the means by which a join point exits (normal or exceptional return).
- *Around advice*: Advice that surrounds a join point such as a method invocation. This is the most powerful kind of advice. Around advice can perform custom behavior before and after the method invocation. It is also responsible for choosing whether to proceed to the join point or to shortcut the advised method execution by returning its own return value or throwing an exception

32. What are the types of the transaction management Spring supports ?
Spring Framework supports:

- Programmatic transaction management.
- Declarative transaction management.

33. What are the benefits of the Spring Framework transaction management ?
The Spring Framework provides a consistent abstraction for transaction management that delivers the following benefits:

- Provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.
- Supports declarative transaction management.
- Provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
- Integrates very well with Spring's various data access abstractions.

34. Why most users of the Spring Framework choose declarative transaction management ?
Most users of the Spring Framework choose declarative transaction management because

it is the option with the least impact on application code, and hence is most consistent with the ideals of a non-invasive lightweight container.

35. Explain the similarities and differences between EJB CMT and the Spring Framework's declarative transaction management ?

The basic approach is similar: it is possible to specify transaction behavior (or lack of it) down to individual method level. It is possible to make a setRollbackOnly() call within a transaction context if necessary. The differences are:

- Unlike EJB CMT, which is tied to JTA, the Spring Framework's declarative transaction management works in any environment. It can work with JDBC, JDO, Hibernate or other transactions under the covers, with configuration changes only.
- The Spring Framework enables declarative transaction management to be applied to any class, not merely special classes such as EJBs.
- The Spring Framework offers declarative rollback rules: this is a feature with no EJB equivalent. Both programmatic and declarative support for rollback rules is provided.
- The Spring Framework gives you an opportunity to customize transactional behavior, using AOP. With EJB CMT, you have no way to influence the container's transaction management other than setRollbackOnly().
- The Spring Framework does not support propagation of transaction contexts across remote calls, as do high-end application servers.

36. What are object/relational mapping integration module?

Spring also supports for using of an object/relational mapping (ORM) tool over straight JDBC by providing the ORM module. Spring provide support to tie into several popular *ORM frameworks*, including *Hibernate*, *JDO*, and *iBATIS SQL Maps*. Spring's transaction management supports each of these *ORM frameworks* as well as *JDBC*.

37. When to use programmatic and declarative transaction management ?

Programmatic transaction management is usually a good idea only if you have a small number of transactional operations.
On the other hand, if your application has numerous transactional operations, declarative transaction management is usually worthwhile. It keeps transaction management out of business logic, and is not difficult to configure.

38. Explain about the Spring DAO support ?
The Data Access Object (DAO) support in Spring is aimed at making it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows one to switch between the persistence technologies fairly easily and it also allows one to code without worrying about catching
exceptions that are specific to each
technology.

39. What are the exceptions thrown by the Spring DAO classes ?
Spring DAO classes throw exceptions which are subclasses of

DataAccessException(org.springframework.dao.DataAccessException).Spring provides a convenient translation from technology-specific exceptions like SQLException to its own exception class hierarchy with the DataAccessException as the root exception. These exceptions wrap the original exception.

40. What is SQLExceptionTranslator ?
SQLExceptionTranslator, is an interface to be implemented by classes that can translate between SQLExceptions and Spring's own data-access-strategy-agnostic org.springframework.dao.DataAccessException.

41. What is Spring's JdbcTemplate ?
Spring's *JdbcTemplate* is central class to interact with a database through JDBC. JdbcTemplate provides many convenience methods for doing things such as converting database data into primitives or objects, executing prepared and callable statements, and providing custom database error handling.
JdbcTemplate template = new JdbcTemplate(myDataSource);

42. What is PreparedStatementCreator ?
PreparedStatementCreator:

- Is one of the most common used interfaces for writing data to database.
- Has one method – createPreparedStatement(Connection)
- Responsible for creating a PreparedStatement.
- Does not need to handle SQLExceptions.

43. What is SQLProvider ?
SQLProvider:

- Has one method – getSql()
- Typically implemented by PreparedStatementCreator implementers.
- Useful for debugging.

44. What is RowCallbackHandler ?
The RowCallbackHandler interface extracts values from each row of a ResultSet.

- Has one method – processRow(ResultSet)
- Called for each row in ResultSet.
- Typically stateful.

45. What are the differences between EJB and Spring ?

Spring and EJB feature comparison.

Feature	EJB	Spring
Transaction management	• Must use a JTA transaction manager.	• Supports multiple transaction environments through its

	• Supports transactions that span remote method calls.	PlatformTransactionManager interface, including JTA, Hibernate, JDO, and JDBC. • Does not natively support distributed transactions—it must be used with a JTA transaction manager.
Declarative transaction support	• Can define transactions declaratively through the deployment descriptor. • Can define transaction behavior per method or per class by using the wildcard character *. • Cannot declaratively define rollback behavior—this must be done programmatically.	• Can define transactions declaratively through the Spring configuration file or through class metadata. • Can define which methods to apply transaction behavior explicitly or by using regular expressions. • Can declaratively define rollback behavior per method and per exception type.
Persistence	Supports programmatic bean-managed persistence and declarative container managed persistence.	Provides a framework for integrating with several persistencetechnologies, including JDBC, Hibernate, JDO, and iBATIS.
Declarative security	• Supports declarative security through users and roles. The management and implementation of users and roles is container specific. • Declarative security is configured in the deployment descriptor.	• No security implementation out-of-the box. • Acegi, an open source security framework built on top of Spring, provides declarative security through the Spring configuration file or class metadata.
Distributed computing	Provides container-managed remote method calls.	Provides proxying for remote calls via RMI, JAX-RPC, and web services.

46 . Do I need any other SOAP framework to run Spring Web Services?

You don't need any other SOAP framework to use Spring Web services, though it can use some of the features of Axis 1 and 2.

47 . I get NAMESPACE_ERR exceptions when using Spring-WS. What can I do about it?
If you get the following Exception:
NAMESPACE_ERR: An attempt is made to create or change an object in a way which is incorrect with regard to namespaces.

Most often, this exception is related to an older version of Xalan being used. Make sure to upgrade to 2.7.0.

48 .Does Spring-WS run under Java 1.3?
Spring Web Services requires Java 1.4 or higher.

49. Does Spring-WS work under Java 1.4?
Spring Web Services works under Java 1.4, but it requires some effort to make it work. Java 1.4 is bundled with the older XML parser Crimson, which does not handle namespaces correctly. Additionally, it is bundled with an older version of Xalan, which also has problems. Unfortunately, placing newer versions of these on the class path does not override them. .
The only solution that works is to add newer versions of Xerces and Xalan in the lib/endorsed directory of your JDK, as explained in those FAQs (i.e.$JAVA_HOME/lib/endorsed). The following libraries are known to work with Java 1.4.2:

Library	Version
Xerces	2.8.1
Xalan	2.7.0
XML-APIs	1.3.04
SAAJ	1.2

If you want to use WS-Security, note that the XwsSecurityInterceptor requires Java 5, because an underlying library (XWSS) requires it. Instead, you can use the Wss4jSecurityInterceptor.

50 .Does Spring-WS work under Java 1.6?
Java 1.6 ships with SAAJ 1.3, JAXB 2.0, and JAXP 1.4 (a custom version of Xerces and Xalan). Overriding these libraries by putting different version on the classpath will result in various classloading issues, or exceptions in org.apache.xml.serializer.ToXMLSAXHandler. The only option for using more recent versions is to put the newer version in the endorsed directory (see above).

51 . Why do the Spring-WS unit tests fail under Mac OS X?
For some reason, Apple decided to include a Java 1.4 compatibility jar with their JDK 1.5. This jar includes the XML parsers which were included in Java 1.4. No other JDK distribution does this, so it is unclear what the purpose of this compatibility jar is.
The jar can be found at /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Classes/.compatibility/14compatibility.jar. You can safely remove or rename it, and the tests will run again.

52 . What is SAAJ?
SAAJ is the SOAP with Attachments API for Java. Like most Java EE libraries, it consists of a set of interfaces (saaj-api.jar), and implementations (saaj-impl.jar). When running in a Application Server, the implementation is typically provided by the application server. Previously, SAAJ has been part of JAXM, but it has been released as a seperate API as part of the , and also as part of J2EE 1.4. SAAJ is generally known as the packagejavax.xml.soap.
Spring-WS uses this standard SAAJ library to create representations of SOAP messages. Alternatively, it can use

53 . What version of SAAJ does my application server support?

Application Server	SAAJ Version
BEA WebLogic 8	1.1
BEA WebLogic 9	1.1/1.2*
BEA WebLogic 10	1.3**
IBM WebSphere 6	1.2
SUN Glassfish 1	1.3
JBoss 4.2	1.3***

54 .I get a NoSuchMethodError when using SAAJ. What can I do about it?
If you get the following stack trace:
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'org.springframework.ws.soap.saaj.SaajSoapMessageFactory' defined in ServletContext resource [/WEB-INF/springws-servlet.xml]:
Invocation of init method failed;
nested exception is java.lang.NoSuchMethodError:
javax.xml.soap.MessageFactory.newInstance(Ljava/lang/String;)Ljavax/xml/soap/MessageFactory;
Caused by:
java.lang.NoSuchMethodError:
javax.xml.soap.MessageFactory.newInstance(Ljava/lang/String;)Ljavax/xml/soap/MessageFactory;

Like most J2EE libraries, SAAJ consists of two parts: the API that consists of interfaces (saaj-api.jar) and the implementation (saaj-impl.jar). The stack trace is due to the fact that you are using a new version of the API (SAAJ 1.3), while your application server provides an earlier version of the implementation (SAAJ 1.2 or even 1.1). Spring-WS supports all three versions of SAAJ (1.1 through 1.3), but things break when it sees the 1.3 API, while there is no 1.3 implementation.
The solution therefore is quite simple: to remove the newer 1.3 version of the API, from the class path, and replace it with the version supported by your application server.

55 . I get an UnsupportedOperationException "This class does not support SAAJ 1.1" when I use SAAJ under WebLogic 9. What can I do about it?
WebLogic 9 has a known bug in the SAAJ 1.2 implementation: it implement all the 1.2 interfaces, but throws UnsupportedOperationExceptions when you call them. Confusingly,

the exception message is This class does not support SAAJ 1.1, even though it supports SAAJ 1.1 just fine; it just doesn't support SAAJ **1.2**. See alsot
Spring-WS has a workaround for this, we basically use SAAJ 1.1 only when dealing with WebLogic 9. Unfortunately, other frameworks which depend on SAAJ, such as XWSS, do not have this workaround. These frameworks happily call SAAJ 1.2 methods, which throw this exception.
The solution is to not use BEA's version of SAAJ, but to use another implementation, like the one from Axis 1, or SUN. In youapplication context, use the following:

```
<bean id="messageFactory"
class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
    <property name="messageFactory">
        <bean class="com.sun.xml.messaging.saaj.soap.MessageFactoryImpl"/>
    </property>
</bean>
```

56 . I get an UnsupportedOperationException "This class does not support SAAJ 1.1" when I use SAAJ under WebLogic 10. What can I do about it?
Weblogic 10 ships with two SAAJ implementations. By default the buggy 9.x implementation is used (which lives in the package weblogic.webservice.core.soap), but there is a new implementation, which supports SAAJ 1.3 (which lives in the package weblogic.xml.saaj). By looking at the DEBUG logging when Spring Web Services starts up, you can see which SAAJ implementation is used.
To use this new version, you have to create a message factory bean like so:

```
<bean id="messageFactory"
class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
    <property name="messageFactory">
        <bean class="weblogic.xml.saaj.MessageFactoryImpl"/>
    </property>
</bean>
```

57 . I get an IndexOutOfBoundsException when I use SAAJ under JBoss. What can I do about it?
The SAAJ implementation provided by JBoss has some issues. The solution is therefore not to use the JBoss implementation, but to use another implementation. For instance, you can use SUN's reference implementation like so:

```
<bean id="messageFactory"
class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
    <property name="messageFactory">
        <bean
class="com.sun.xml.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl"/>
    </property>
</bean>
```

58 . Does Spring-WS run on IBM WebSphere?
WebSphere bundles some libraries which are out-of-date, and need to be upgraded with more recent versions. Specifically, this includes XML-apis, Xerces, Xalan, and WSDL4J.

There are a couple of ways to upgrade these libraries, all using parent-last or application-first classloading.

- Package the libraries as part of the WAR (in WEB-INF/lib), and run the web application with the parent-last (application-first) classloading.
- Package the libraries as part of the EAR, add class-path entries to the manifest of the web application, and run the entire application with the parent-last classloading.
- Create a new classloader in the WebSphere console, and associate the libraries with. Set this classloader to parent-last.

The last approach has the advantage of restricting the parent-last classloading to the conflicting libraries only, and not to the entire application.

59. Why does Spring-WS only support contract-first?
Note that Spring-WS only requires you to write the XSD; the WSDL can be generated from that.

60 . How do I retrieve the WSDL from a Service?
The &WSDL query parameter does not work.
The &WSDL query parameter is a way to get a WSDL of a class. In SWS, a service is generally not implemented as a single class, but as a collection of endpoints.
There are two ways to expose a WSDL:

- Simply add the WSDL to the root of the WAR, and the file is served normally. This has the disadvantage that the "location"attribute in the WSDL is static, i.e. it does not necessarily reflect the host name of the server. You can transform locations by using a WsdlDefinitionHandlerAdapter.
- Use theMessageDispatcherServlet, which is done is the samples. Every WsdlDefinition listed in the *-servlet.xml will be exposed under the bean name. So if you define a WsdlDefinition namedecho, it will be exposed as echo.wsdl (i.e.http://localhost:8080/echo/echo.wsdl).

61 What is web module?

Spring comes with a full-featured MVC framework for building web applications. Although Spring can easily be integrated with other MVC frameworks, such as Struts, Spring's MVC framework uses IoC to provide for a clean separation of controller logic from business objects. It also allows you to declaratively bind request parameters to your business objects. It also can take advantage of any of Spring's other services, such as I18N messaging and validation.

62 What is a BeanFactory?

A BeanFactory is an implementation of the factory pattern that applies Inversion of Control to separate the application's configuration and dependencies from the actual application code.

63 What is AOP Alliance?

AOP Alliance is an open-source project whose goal is to promote adoption of AOP and interoperability among different AOP implementations by defining a common set of interfaces and components.

64 What is Spring configuration file?

Spring configuration file is an XML file. This file contains the classes information and describes how these classes are configured and introduced to each other.

65 .What does a simple spring application contain?

These applications are like any Java application. They are made up of several classes, each performing a specific purpose within the application. But these classes are configured and introduced to each other through an XML file. This XML file describes how to configure the classes, known as theSpring configuration file.

66 What is XMLBeanFactory?
BeanFactory has many implementations in Spring. But one of the most useful one is***org.springframework.beans.factory.xml.XmlBeanFactory***, which loads its beans based on the definitions contained in an XML file. To create an ***XmlBeanFactory***, pass a java.io.InputStream to the constructor. The ***InputStream*** will provide the XML to the factory. For example, the following code snippet uses a java.io.***FileInputStream*** to provide a bean definition XML file to***XmlBeanFactory***.

BeanFactory factory = new **XmlBeanFactory**(new **FileInputStream**("beans.xml"));
To retrieve the bean from a BeanFactory, call the getBean() method by passing the name of the bean you want to retrieve.

MyBean myBean = (MyBean) factory.**getBean**("myBean")

67 . What are important ApplicationContext implementations in spring framework?

- **ClassPathXmlApplicationContext –** This context loads a context definition from an XML file located in the class path, treating context definition files as class path resources.
- **FileSystemXmlApplicationContext –** This context loads a context definition from an XML file in the filesystem.
- **XmlWebApplicationContext –** This context loads the context definitions from an XML file contained within a web application.

68 . Explain Bean lifecycle in Spring framework?

- The spring container finds the bean’s definition from the XML file and instantiates the bean.

- Using the dependency injection, spring populates all of the properties as specified in the bean definition.
- If the bean implements the **BeanNameAware** interface, the factory calls **setBeanName()** passing the bean's ID.
- If the bean implements the **BeanFactoryAware** interface, the factory calls **setBeanFactory()**, passing an instance of itself.
- If there are any **BeanPostProcessors** associated with the bean, their **post-ProcessBeforeInitialization()** methods will be called.
- If an init-method is specified for the bean, it will be called.
- Finally, if there are any **BeanPostProcessors** associated with the bean, their **postProcessAfterInitialization()**methods will be called.

69 What is bean wiring?
Combining together beans within the Spring container is known as bean wiring or wiring. When wiring beans, you should tell the container what beans are needed and how the container should use dependency injection to tie them together.

70 How do add a bean in spring application?

```
	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
	<beans>
		<bean id="foo" class="com.act.Foo"/>
		<bean id="bar" class="com.act.Bar"/>
	</beans>
```

71. What are singleton beans and how can you create prototype beans?

Beans defined in spring framework are singleton beans. There is an attribute in bean tag named 'singleton' if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

```
	<beans>
	  <bean id="bar" class="com.act.Foo" singleton="false"/>
	</beans>
```

72 . What are the important beans lifecycle methods?

There are two important bean lifecycle methods. The first one is setup which is called when the bean is loaded in to the container. The second method is the teardown method which is called when the bean is unloaded from the container.

73 . How can you override beans default lifecycle methods?

The bean tag has two more important attributes with which you can define your own custom initialization and destroy methods. Here I have shown a small demonstration. Two new methods fooSetup and fooTeardown are to be added to your Foo class.

```
      <beans>
        <bean id="bar" class="com.act.Foo" init-method=”fooSetup”
destroy=”fooTeardown”/>
      </beans>
```

74 . What are Inner Beans?

When wiring beans, if a bean element is embedded to a property tag directly, then that bean is said to the Inner Bean. The drawback of this bean is that it cannot be reused anywhere else.

75. What are the different types of bean injections?

There are two types of bean injections.

- By setter
- By constructor

76. What is Auto wiring?

You can wire the beans as you wish. But spring framework also does this work for you. It can auto wire the related beans together. All you have to do is just set the autowire attribute of bean tag to an autowire type.

```
      <beans>
        <bean id="bar" class="com.act.Foo" Autowire=”autowire type”/>
      </beans>
```

77 . What are different types of Autowire types?

There are four different types by which autowiring can be done.

- byName
- byType
- constructor
- autodetect

78. What are the different types of events related to Listeners?

There are a lot of events related to ApplicationContext of spring framework. All the events are subclasses of org.springframework.context.Application-Event. They are

- ContextClosedEvent – This is fired when the context is closed.

- ContextRefreshedEvent – This is fired when the context is initialized or refreshed.
- RequestHandledEvent – This is fired when the web context handles any request.

79. What is an Aspect?

An aspect is the cross-cutting functionality that you are implementing. It is the aspect of your application you are modularizing. An example of an aspect is logging. Logging is something that is required throughout an application. However, because applications tend to be broken down into layers based on functionality, reusing a logging module through inheritance does not make sense. However, you can create a logging aspect and apply it throughout your application using AOP.

80 . What is a Jointpoint?

A joinpoint is a point in the execution of the application where an aspect can be plugged in. This point could be a method being called, an exception being thrown, or even a field being modified. These are the points where your aspect's code can be inserted into the normal flow of your application to add new behavior.

81 What is an Advice?

Advice is the implementation of an aspect. It is something like telling your application of a new behavior. Generally, and advice is inserted into an application at joinpoints.

82 What is a Pointcut?

A pointcut is something that defines at what joinpoints an advice should be applied. Advices can be applied at any joinpoint that is supported by the AOP framework. These Pointcuts allow you to specify where theadvice can be applied.

83 What is an Introduction in AOP?

An introduction allows the user to add new methods or attributes to an existing class. This can then be introduced to an existing class without having to change the structure of the class, but give them the new behavior and state.

84 What is a Target?

A target is the class that is being advised. The class can be a third party class or your own class to which you want to add your own custom behavior. By using the concepts of AOP, the target class is free to center on its major concern, unaware to any advice that is being applied.

85 . What is a Proxy?

A proxy is an object that is created after applying advice to a target object. When you think of client objects the target object and the proxy object are the same.

86 .What is meant by Weaving?

The process of applying aspects to a target object to create a new proxy object is called as Weaving. The aspects are woven intothe target object at the specified joinpoints.

87 What are the different points where weaving can be applied?

- Compile Time
- Classload Time
- Runtime

88 . What are the different advice types in spring?

- Around : Intercepts the calls to the target method
- Before : This is called before the target method is invoked
- After : This is called after the target method is returned
- Throws : This is called when the target method throws and exception
- Around : org.aopalliance.intercept.MethodInterceptor
- Before : org.springframework.aop.BeforeAdvice
- After : org.springframework.aop.AfterReturningAdvice
- Throws : org.springframework.aop.ThrowsAdvice

89 What are the different types of AutoProxying?

- BeanNameAutoProxyCreator
- DefaultAdvisorAutoProxyCreator
- Metadata autoproxying
- **90 What kind of exceptions those spring DAO classes throw?**
- The springâ€™s DAO class does not throw any technology related exceptions such as SQLException. They throw exceptions which are subclasses of DataAccessException.
- **91 What is DataAccessException?**
- DataAccessException is a RuntimeException. This is an Unchecked Exception. The user is not forced to handle these kinds of exceptions.
- **92 How can you configure a bean to get DataSource from JNDI?**
-
```
        <bean id="dataSource"
class="org.springframework.jndi.JndiObjectFactoryBean">
             <property name="jndiName">
               <value>java:comp/env/jdbc/myDatasource</value>
            </property>
        </bean>
```
- **93 How can you create a DataSource connection pool?**
-
```
        <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
           <property name="driver">
              <value>${db.driver}</value>
           </property>
           <property name="url">
             <value>${db.url}</value>
           </property>
```

```
            <property name="username">
              <value>${db.username}</value>
            </property>
            <property name="password">
             <value>${db.password}</value>
            </property>
          </bean>
```

- **94 How JDBC can be used more efficiently in spring framework?**
- JDBC can be used more efficiently with the help of a template class provided by spring framework called as JdbcTemplate.
- **95 How JdbcTemplate can be used?**
- With use of Spring JDBC framework the burden of resource management and error handling is reduced a lot. So it leavesdevelopers to write the statements and queries to get the data to and from the database.

-
  ```
  JdbcTemplate template = new JdbcTemplate(myDataSource);
  ```
  A simple DAO class looks like this.

-
  ```
  public class StudentDaoJdbc implements StudentDao {
    private JdbcTemplate jdbcTemplate;
  ```

-
  ```
  public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
     this.jdbcTemplate = jdbcTemplate;
  }
  more..
  }
  ```
  The configuration is shown below.

-
  ```
  <bean id="jdbcTemplate"
  class="org.springframework.jdbc.core.JdbcTemplate">
     <property name="dataSource">
         <ref bean="dataSource"/>
     </property>
  </bean>
  <bean id="studentDao" class="StudentDaoJdbc">
     <property name="jdbcTemplate">
         <ref bean="jdbcTemplate"/>
     </property>
  </bean>
  <bean id="courseDao" class="CourseDaoJdbc">
     <property name="jdbcTemplate">
         <ref bean="jdbcTemplate"/>
     </property>
  </bean>
  ```
- **96 How do you write data to backend in spring using JdbcTemplate?**
- The JdbcTemplate uses several of these callbacks when writing data to the database. The usefulness you will find in each of these interfaces will vary. There are two simple interfaces. One is PreparedStatementCreator and the other interface is BatchPreparedStatementSetter.
- **97 Explain about PreparedStatementCreator?**
- PreparedStatementCreator is one of the most common used interfaces for writing data to database. The interface has one method createPreparedStatement().

- PreparedStatement createPreparedStatement(Connection conn)
  throws SQLException;

  When this interface is implemented, we should create and return a PreparedStatement from the Connection argument, and the exception handling is automatically taken care off. When this interface is implemented, another interface SqlProvider is also implemented which has a method called getSql() which is used to provide sql strings to JdbcTemplate.
- **98 Explain about BatchPreparedStatementSetter?**
- If the user what to update more than one row at a shot then he can go for BatchPreparedStatementSetter. This interface provides two methods
- setValues(PreparedStatement ps, int i) throws SQLException;

  int getBatchSize();

  The getBatchSize() tells the JdbcTemplate class how many statements to create. And this also determines how many times setValues() will be called.
- **99. Explain about RowCallbackHandler and why it is used?**
- In order to navigate through the records we generally go for ResultSet. But spring provides an interface that handles this entire burden and leaves the user to decide what to do with each row. The interface provided by spring is RowCallbackHandler. There is a method processRow() which needs to be implemented so that it is applicable for each and everyrow.
- void processRow(java.sql.ResultSet rs);
-
- **100 What is JDBC abstraction and DAO module?**
- Using this module we can keep up the database code clean and simple, and prevent problems that result from a failure to close database resources. A new layer of meaningful exceptions on top of the error messages given by several database servers is bought in this module. In addition, this module uses Spring's AOP module to provide transaction management services for objects in a Spring application.
-