constructor arguments for mandatory dependencies and setters for optional dependencies.

difference between map and props in spring

-> Arrays.sort(arrayName);

-> Arrays.toString(arrayName); - print array

-> Quicksort time complexity O(nlogn)

-> Mergesort time complexity O(nlogn)

-> Arrays.java's sort method uses quicksort for arrays of primitives and merge sort for arrays of objects.

-> write algorithm to sort an array in Java

-> check for palindrome

-> oops concepts

-> exceptional handling

**-> What is the purpose of serialization?**

Answer: Serialization is the conversion of an object to a series of bytes, so that the object can be easily saved to persistent storage or streamed across a communication link. The byte stream can then be deserialised – converted into a replica of the original object.

**-> What is the difference between JDK and JRE?**

Answer: JRE is an acronym for Java Runtime Environment.It is used to provide runtime environment.It is the implementation of JVM.It physically exists.It contains set of libraries + other files that JVM uses at runtime.

JDK is an acronym for Java Development Kit.It physically exists.It contains JRE + development tools.

**-> What is the difference between equals() and “==” ?**

Answer: Equals is intended to check logical equality and == checks if both references point to same object.

a == b; // Compares references, not values.

a.equals(b); // Compares values for equality.

String a = "hello"; String b = "Kathe";

if(a.equals(b)) s.o.p(Equal for .equals )

else if(a == b) s.o.p(Equal for ==)

o/p: Equals for .equals

-> **When will you use Comparator and Comparable interfaces?**

Answer: java.util.Comparator and java.lang.Comparable

java.util.Comparator compares some other class’s instances, while java.lang.Comparable compares itself with another object.

* To sort set -> use Tree Set
* To sort List use Comparable
* **What is the wait/notify mechanism?**  
  Answer: This deals with concurrent programming. The wait() and notify() methods are designed to provide a mechanism to allow a thread to be block until a specific condition is met.  
  However, java.util.concurrent should be used instead of wait() and notify() to reduce complexity.
* **Write program for deadlock**
* **What is the difference between checked and unchecked exceptions?**  
  Answer:  
  In general, unchecked exceptions represent defects in the program (bugs), which are normally Runtime exceptions.  
  Furthermore, checked exceptions represent invalid conditions in areas outside the immediate control of the program.
* **What is the difference between final, finally and finalize?**  
  Answer: “final” is the keyword to declare a constant **AND** prevents a class from producing subclasses. (Thanks Tom Ellis)  
  “finally” is a block of code that always executes when the try block is finished, unless System.exit() was called. finalize() is an method that is invoked before an object is discarded by the garbage collector.
* **What is the difference between web server and app server?**  
  Answer: A Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.
* **Explain the Struts1/Struts2/MVC application architecture**  
  Answer: Struts was adopted by the Java developer community as a default web framework for developing web applications  
  The MVC(Model–view–controller) an application that consist of three distinct parts. The problem domain is represented by the Model. The output to the user is represented by the View. And, the input from the user is represented by Controller.
* **What is the difference between forward and sendredirect?**  
  Answer: Both method calls redirect you to new resource/page/servlet. The difference between the two is that sendRedirect always sends a header back to the client/browser, containing the data in which you wanted to be redirected.
* **How does a 3 tier application differ from a 2 tier one?**  
  Answer: Tiers are the physical units of separation or deployment, while layers are the logical units of separation.  
  Imagine that you’re designing an e-commerce website. A 3 tier architecture would consist of web pages, a web server and a database, with the corresponding 3 layers being the “Presentation”, “Business Logic” and “Database” layers.  
  If you take the database tier and layer out then your have a 2 tier architecture.
* **How does the version control process works?**  
  Answer: Initiate, pull, branch, merge, commit, push.  
  (Init) Make your own repository. (Pull) Download an existing repository from a url. (Branch / Merge )Make revisions. Commit then push your modifications.
* **What is the difference between JAR and WAR files**  
  Answer: JAR files (Java ARchive) allows aggregating many files into one, it is usually used to hold Java classes in a library.  
  WAR files (Web Application aRchive) stores XML, java classes, and JavaServer pages for Web Application purposes.
* **What is a Left outer join?**  
  Answer: *This deals with SQL.* Left outer join preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.
* **What is the difference between UNION and UNION ALL?**  
  Answer: *This deals with SQL.* UNION only selects distinct values, UNION ALL selects all values.
* **what is diff between procedural oriented language and object oriented language**

|  |  |
| --- | --- |
| **Procedure Oriented Programming** | Object Oriented Programming |
| **Divided Into** | In POP, program is divided into small parts called**functions**. | In OOP, program is divided into parts called **objects**. |
| **Importance** | In POP,Importance is not given to **data** but to functions as well as **sequence** of actions to be done. | In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**. |
| **Approach** | POP follows**Top Down approach**. | OOP follows **Bottom Up approach**. |
| **Access Specifiers** | POP does not have any access specifier. | OOP has access specifiers named Public, Private, Protected, etc. |
| **Data Moving** | In POP, Data can move freely from function to function in the system. | In OOP, objects can move and communicate with each other through member functions. |
| **Expansion** | To add new data and function in POP is not so easy. | OOP provides an easy way to add new data and function. |
| **Data Access** | In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system. | In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data. |
| **Data Hiding** | POP does not have any proper way for hiding data so it is **less secure**. | OOP provides Data Hiding so provides **more security**. |
| **Overloading** | In POP, Overloading is not possible. | In OOP, overloading is possible in the form of Function Overloading and Operator Overloading. |
| **Examples** | Example of POP are : C, VB, FORTRAN, Pascal. | Example of OOP are : C++, JAVA, VB.NET, C#.NET. |

* **difference between abstraction and encapsulation ?**

|  |  |
| --- | --- |
| Encapsulation | binding the data with its related functionalities |
| Abstraction | **Hiding the data.** We achieve abstraction through encapsulation. Without binding the data we don’t have class. Without class and declaring variables as private we can’t hide the data. Hence without encapsulation we can not have abstraction |

* **Is Abstraction and abstract class both are same?**

|  |  |
| --- | --- |
| abstract class | partially implemented and partially unimplemented structure |

* **difference between abstraction and interface**

|  |  |
| --- | --- |
| Interface | fully unimplemented structure |
| What an interface can contain?  1. abstract methods (methods without implementation) 2. constants (public static final variables) 3. default methods - methods declared with default keyword 4. static methods - methods declared with static keyword |  |

* **can muliple inheritan e possible in java using abstract class?**

No.

* **.what is o/p of below code**  
  interface a{  
  int i=10}  
    
  interface b{  
  int i=5}  
    
  class c a,b{  
  sop(i)  
  }  
    
  o/p: compile error: call to I is ambiguous.
* write code for jdbc connection in java

//STEP 2: Register JDBC driver

Class.forName("com.mysql.jdbc.Driver");

//STEP 3: Open a connection

System.out.println("Connecting to database...");

conn = DriverManager.getConnection(DB\_URL,USER,PASS);

//STEP 4: Execute a query

System.out.println("Creating statement...");

stmt = conn.createStatement();

String sql;

sql = "SELECT id, first, last, age FROM Employees";

ResultSet rs = stmt.executeQuery(sql);

//STEP 5: Extract data from result set

while(rs.next()){

//Retrieve by column name

int id = rs.getInt("id");

int age = rs.getInt("age");

String first = rs.getString("first");

String last = rs.getString("last");

//Display values

System.out.print("ID: " + id);

System.out.print(", Age: " + age);

System.out.print(", First: " + first);

System.out.println(", Last: " + last);

}

//STEP 6: Clean-up environment

rs.close();

stmt.close();

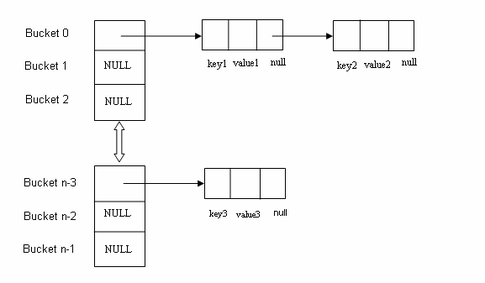
conn.close();

* **How Hashmap internally work**

**Refer :** [**http://netjs.blogspot.in/2015/05/how-hashmap-internally-works-in-java.html**](http://netjs.blogspot.in/2015/05/how-hashmap-internally-works-in-java.html)

There are four things we should know about before going into internals of how HashMap works -

* **HashMap** works on the principal of hashing.
* **Map.Entry interface** - This interface gives a map entry (key-value pair). HashMap in Java stores both key and value object, in bucket, as Entry object which implements this nested interface Map.Entry.
* **hashCode()** -HashMap provides put(key, value) for **storing** and get(key) method for**retrieving** Values from HashMap. When put() method is used to store (Key, Value) pair, HashMap implementation **calls hashcode** on Key object to calculate a hash that is used to find a bucket where Entry object will be stored. When get() method is used to retrieve value, again key object is used to calculate a hash which is used then to find a bucket where that particular key is stored.
* **equals()** - equals() method is used to **compare objects for equality**. In case of HashMap key object is used for comparison, also using equals() method Map knows how to handle **hashing collision** (hashing collision means more than one key having the same hash value, thus assigned to the same bucket. In that case objects are stored in a linked list.  
  Where hashCode method helps in finding the bucket where that key is stored, equals method helps in finding the right key as there may be more than one key-value pair stored in a single bucket.



***HashMap changes in Java 8***  
Though HashMap implementation provides constant time performance O(1) for get() and put() method but that is in the ideal case when the Hash function distributes the objects evenly among the buckets.  
But the performance may worsen in the case hashCode() used is not proper and there are lots of hash collisions. As we know now that in case of hash collision entry objects are stored as a node in a linked-list and equals() method is used to compare keys. That comparison to find the correct key with in a linked-list is a linear operation so in a worst case scenario the complexity becomes O(n).  
To address this issue in Java 8 hash elements use balanced trees instead of linked lists after a certain threshold is reached. Which means HashMap starts with storing Entry objects in linked list but after the number of items in a hash becomes larger than a certain threshold, the hash will change from using a linked list to a balanced tree, this will improve the worst case performance from O(n) to O(log n).  
  
**🡪 11.write code to retrieve elements from hashtable**

Hashtable<Integer,String> h = **new** Hashtable<Integer,String>();

h.put(1, "one");

h.put(2, "two");

h.put(3, "three");

// first method

System.***out***.println("first method");

Enumeration enums = h.keys();

**while**(enums.hasMoreElements())

System.***out***.println(h.get(enums.nextElement()));

//second method

System.***out***.println("Second method");

Iterator i= h.entrySet().iterator();

**while**(i.hasNext())

System.***out***.println(i.next());

//third method

System.***out***.println("third method");

Set<Integer> s = h.keySet();

**for**(**int** j:s)

System.***out***.println(j+": "+h.get(j));

// fourth method

System.***out***.println("fourth Method");

**for**(Map.Entry<Integer,String> row : h.entrySet())

System.***out***.println(row);  
  
**🡪 12.diff between vector and arraylist**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| 1) ArrayList is **not synchronized**. | Vector is **synchronized**. |
| 2) ArrayList **increments 50%** of current array size if number of element exceeds from its capacity. | Vector **increments 100%** means doubles the array size if total number of element exceeds than its capacity. |
| 3) ArrayList is **not a legacy** class, it is introduced in JDK 1.2. | Vector is a **legacy** class. |
| 4) ArrayList is **fast** because it is non-synchronized. | Vector is **slow** because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable state until current thread releases the lock of object. |
| 5) ArrayList uses **Iterator** interface to traverse the elements. | Vector uses **Enumeration** interface to traverse the elements. But it can use Iterator also. |

**1**.  Synchronization and Thread-Safe  
  
Vector is  synchronized while ArrayList is not synchronized  . Synchronization and thread safe means at a time only one thread can access the code .In Vector class all the methods are synchronized .Thats why the Vector object is already synchronized when it is created .  
  
2.  Performance  
  
Vector is slow as it is thread safe . In comparison ArrayList is fast as it is non synchronized . Thus     in ArrayList two or more threads  can access the code at the same time  , while Vector is limited to one thread at a time.  
  
3. Automatic Increase in Capacity  
  
A Vector defaults to doubling size of its array . While when you insert an element into the ArrayList ,      it increases  
its Array size by 50%  .  
  
  
By default ArrayList size is 10 . It checks whether it reaches the       last  element then it will create the new array ,copy the new data of last array to new array ,then old array     is garbage collected by the Java Virtual Machine (JVM) .   
  
4. Set Increment Size  
  
ArrayList does not define the increment size . Vector defines the increment size .  
  
You can find the following method in Vector Class  
  
public synchronized void setSize(int i) { //some code  }  
  
There is no setSize() method or any other method in ArrayList which can manually set the increment size.  
  
5. Enumerator  
  
Other than Hashtable ,Vector is the only other class which uses both[Enumeration and Iterator](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html) .While ArrayList can only use Iterator for traversing an ArrayList .  
  
6.  Introduction in Java   
  
java.util.Vector  class was there in java since the very first version of the java development kit (jdk).  
java.util.ArrayList  was introduced in java version 1.2 , as part of Java Collections framework . In java version 1.2 , Vector class has been refactored to implement the List Inteface .  
  
13.what is synchronization  
  
14.what ibm websphere?is it application server or web server?why you use it?  
  
15.diff between application server and web server  
  
16.what is o/p of below code  
  
class a  
{  
int i=10}  
  
class b  
{  
int i=5}  
  
public class c extends a,b{  
psvm{  
  
 sop(i)  
  
 }

**Methods to load driver’s class.**

1. Class.forName();

dynamically load the driver's class file into memory, which automatically registers it. This method is preferable because it allows you to make the driver registration configurable and portable.

Class.forName("oracle.jdbc.driver.OracleDriver");

1. DriverManager.registerDriver()

You should use the *registerDriver()* method if you are using a non-JDK compliant JVM

Driver myDriver = new oracle.jdbc.driver.OracleDriver();

DriverManager.registerDriver( myDriver );

* **Difference between add() and addElement in Vector**

The add() methods inserts an element at a given position of the vector.  
  
- The addElement () method adds an object at the end of the vector and increases the size of the vector by one.

List has an add method, so an implementation was added to Vector, but to maintain backwards-compatibility, addElement wasn't removed

* **Which all can use both Enumerator and Iterator?**

Hashtable and Vector.

* **Converting Array to ArrayList**

ArrayList al = new ArrayList();

Al.addAll(Arrays.asList(arrayName)));

* **Difference between Enumerator and Iterator**

Iterators are **fail-fast** . i.e. when one thread changes the collection by add / remove operations , while another thread is traversing it through an Iterator using hasNext() or next() method, the iterator fails quickly by throwing ConcurrentModificationException . The fail-fast behavior of iterators can be used only to detect bugs. The Enumerations returned by the methods of classes like Hashtable, Vector are not fail-fast that is achieved by synchronizing the block of code inside the nextElement()method that locks the current Vector object which costs lots of time.

Iterators differ from enumerations in two ways:

* Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.
* Method names have been improved.

The bottom line is, both Enumeration and Iterator will give successive elements, but Iterator is improved in such a way so the method names are shorter, and has an additional remove method. Here is a side-by-side comparison:

Enumeration Iterator

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

hasMoreElement() hasNext()

nextElement() next()

N/A remove()

If you're writing your own collection class, and you're extending any of the existing classes or implementing any of the Collections framework interfaces, you basically have no choice but to use Iterator.

* We cannot add elements into ArrayList after converting from Array to ArrayList
* Converting Iterator to ArrayList

List<T> list = new ArrayList<>();

iterator.forEachRemaining(list::add);

**Hibernate:**

* **Difference between primitive type and wrapper type in hibernate POJO.**

The difference between the two is nullability. the primitive type is unable to be null, while the "Wrapped" type can be null.

I prefer to use the wrapped type as you can tell if the object has been saved/loaded to/from the database whether or not the id value is null.

* **Different ways of printing array**

There are 5 ways

1. For loop
2. For each loop
3. Arrays.toString(array);
4. Arrays.deepToString(array);
5. Arrays.asList(array);