linkedlist

treeset

linkedhashset

Hashset

linkedlist

arraylist

Queue

Sorted Set

Set

List

Collection

Iterable

Main interface are : list ,set, sortedset,queue.

Array List : The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed.

LinkedList : LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

Queue : Queue interface maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed. There are various classes like PriorityQueue, Deque, and ArrayDeque which implements the Queue interface.

Set : Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

2.ans

Map

>SortedMap

>TreeMap

>HashMap

>LinkedHashMap

MAP : A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.A Map is useful if you have to search, update or delete elements on the basis of a key.

Map.entry : Entry is the subinterface of Map. So we will be accessed it by Map.Entry name. It returns a collection-view of the map, whose elements are of this class. It provides methods to get key and value.

3.ans :

Hash Set :

>It does not provide a guarantee to sort the data.

>In HashSet**, only an element** can be null.

>It uses **hashCode( )­­** or **equals( )** method for comparison.

>It is **faster** than TreeSet.

>It allows only **heterogeneous** value.

TreeSet :

>It provides a guarantee to sort the data. The sorting depends on the supplied Comparator.

>It does not allow null elements.

>It uses **compare()** or **compareTo()** method for comparison.

>It is **slower** in comparison to HashSet.

>It allows only **homogeneous** value.

9.ans

Core (spring-core) is the core of the framework that power features such as Inversion of Control and dependency injection. Beans (spring-beans) provides Beanfactory, which is a sophisticated implementation of the factory pattern.

10 ans.

**dependency injection** is a technique whereby one object supplies the dependencies of another object.

11.ans : By using @Autowired and @ ComponentScan annotation

12 ans : yes we can create more than one config file and in config file put an @Configuration Annotation

Spring will check the Annotations whether that file have @Configuration annotation ie there are not

13.ans : String Builder : StringBuilder class is used to create mutable (modifiable) String. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.

String Buffer : StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

15 ans:

15.4 : @Component is a class-level annotation. It is used to denote a class as a Component. We can use @Component across the application to mark the beans as Spring's managed components. A component is responsible for some operations.

15.5 : @ComponentScan which is used along with the @Configuration annotation tospecify the packages that we want to be scanned. @ComponentScan without arguments tells Spring to scan the current package and all of its sub-packages.

15.6 : @Bean annotation which is applied on a method to specify that it returns a bean to be managed by Spring context. Spring Bean annotation is usually declared in Configuration classes methods. This annotation is also a part of the spring core framework.

17 ans :

17.1 : Maven. It allows you to package executable jar or war archives, run Spring Boot applications, generate build information and start your Spring Boot application prior to running integration tests.

17.3 : Tomcat is a web container. It allows the users to run Servlet and JAVA Server Pages that are based on the web-applications.

17.4 : Quarkus allows developers to automatically generate Kubernetes resources including building and deploying container images without having to manually create YAML files.

17.6 : A server stores, sends, and receives data. It accept client request and send the response to client.

16 ans :

**public** **class** CountryCode {

**public** **static** **void** main(String[] args) {

Map<Integer,String> pa = **new** HashMap<Integer,String>();

pa.put(577201,"India");

pa.put(577301,"ShriLanka");

pa.put(577401,"Nepal");

pa.put(577418,"Japan");

pa.put(577427,"Buthan");

System.***out***.println("Lopping only Keys \n");

Collection<Integer> key =pa.keySet();

key.forEach(k-> System.***out***.println(k));

System.***out***.println("Lopping only Values \n");

Collection<String> values =pa.values();

values.forEach(v-> System.***out***.println(v));

System.***out***.println("ENTRIES\n");

Set<Entry<Integer, String>> entries = pa.entrySet();

**for**(Entry<Integer,String> entry:entries)

{

System.***out***.println(entry.getKey()+" : " +entry.getValue());

}

}

14. ans :

ApplicationContext provides basic features in addition to enterprise-specific functionalities which are as follows: Publishing events to registered listeners by resolving property files. Methods for accessing application components. Supports Internationalization.

20. Ans

public class Exercise13 {

public static void main(String[] args) {

LinkedList<String> l\_list = new LinkedList<String>();

l\_list.add("Red");

l\_list.add("Green");

l\_list.add("Black");

l\_list.add("Pink");

l\_list.add("orange");

System.out.println("The Original linked list: " + l\_list);

Object firstElement = l\_list.removeFirst();

System.out.println("Element removed: "+ firstElement);

Object lastElement = l\_list.removeLast();

System.out.println("Element removed: "+ lastElement);

System.out.println("The New linked list: " + l\_list);

}

}

18.ans

public class Replace {

    static void manipulateString(String str)

    {

         char[] str1 = str.toCharArray();

      for (int i = 0; i < str.length(); i++)

{

         int asc = str1[i];

         int rem = asc - (26 - (str1[i] - 97));

         int m = rem % 26;

         str1[i] = (char)(m + 'a');

     }

     String str2 = String.valueOf(str1);

     System.out.println(str2);

    }

    public static void main(String[] args) {

         String str = "ManoharaAB";

         manipulateString(str);

    }

}