Day9 Revisit

IO Streams, File Handling, Serialization & De-Serialization

1. Streams in java, represents flow of data
2. Types of Streams a) Byte Stream (1 Byte) b) character Stream (2 bytes)
3. In Byte stream each data is of size 1 byte (8 bits)
4. In Character Stream each data is of size 2 bytes (16 bits)
5. All the classes belongs to byte stream will have Stream at the end
6. All the classes belongs to character stream will have Reader/Writter at the end
7. InputStream handles byte stream based input operation
8. OutputSream handles byte stream based output operation.
9. In Java, the default input device is keyboard and default output device is console (monitor)
10. File is a class in java, which is used to handle any file/directory in the underlying operating system
11. File class will not help to read/write the data. Instead it will help to create a file/directory (folder) and also to check various options of file & directory
12. In DOS based OS, the file path should have \\, where as in unix based OS, file path should contain /
13. File/Object/Piped InputStream – File/Object/Piped Output Stream (Buffered & Un-buffered)
14. StringTokenizer (java.util) StreamTokenizer(java.io)
15. All the classes, abstract classes and interfaces related to IO streams are defined in java.io package
16. While doing any IO related operation we need to handle few checked exceptions. They are FileNotFoundException, IOException,
17. Serialization – It’s the process of storing the state of an object to a permanent storage (Flat file)
18. To Serialize an object of a class, the class must implement a marker interface called Serializable
19. In order to customize few state values while serialization we can use a sub interface of Serializable called Externalizable
20. De-Serialization is the reverse of serialization process. Reading a flat file where we have serialized an object, and creating java object from it.
21. For Serialization & De-Serialization we normally use ObjectInput/OutputStream along with FileInput/OutputStream
22. Using Externalizable interface we can Encrypt or decrypt passwords and other sensitive informations, so that it will not be available to the external world.

Class Employee implements Serializable, Externalizable {

Private int id;

Private String name;

Private String pan; //this property will not be serialized

Private String aadhaar;

<https://codegym.cc/groups/posts/113-introducing-the-externalizable-interface>

Agenda

Collections, Reflection, Generics

Collection is an API (Set of classes and interfaces used for specific purpose)

* Collections are used to handle group of objects
* Collections will not work with primitive data types
* Important interfaces available in collection api List, Set & Map
* Collection Class Hierarchy Iterable(I) – Collection (I) – Set(I), List(I), Map(I)
* Collection api classes and interfaces are defined in java.util package
* Generalisation (Interface & abstract class) & Specialization (class)
* Collection API is introduced in Java 1.2
* Collection in API we also have a importance class called Collections using which we can sort, search, compare,replace,swap, fill data of collection
* 7 interfaces of Collection API/Framework Collection, Set, List, Queue, SortedSet, Map, SortedMap
* Important utility interfaces are Comparator, Iterator, Enumerator
* Important utility class Collections
* List [can have duplicates, can have null many times, dynamically growable array, used to represent sequence of values/data] (ArrayList, Vector, LinkedList)
* Set [Duplicates are not allowed, null is allowed only once, insertion order] (HashSet, LinkedHashSet)
* Map [Stores the data in the form of Key & Value, Key should be unique, values can hold duplicates data] (hashMap, LinkedHashMap, HashTable)
* SortedSet (Stores the data in a sorted order, duplicates are not allowed, it will not maintain insertion order – default sort behavior is ascending order) [TreeSet]
* SortedMap (Stores the key in ascending order) [TreeMap]
* In Arraylist insert in the middle is costlier operation (It consumes either more memory or more cpu)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| ABC | Sundar | XYZ | Satya | Mark | Elon |



<https://docs.oracle.com/javase/8/docs/technotes/guides/collections/reference.html> - collection framework reference

Boxing – is the process of converting primitive value to it’s corresponding object using wrapper classes

Un-Boxing – is the process of converting Object to it’s primitive value

Java will automatically convert primitive to it’s corresponding object whenever required – This process is called as auto-boxing.

Wrapper classes are used for converting primitive to objects, converting decimal to binary, octal & hexa decimal, for Parsing Int value from a string.

Problem with ArrayList

1. Inserting in the middle or beginning is the costlier process in ArrayList bcos it involves lot of shifting operation
2. In Arraylist the elements/data are stored in a continuous memory location.
3. ArrayList is best suitable for Adding/Removing the data at the end. Easiest operation in arraylist are adding/removing data/element at the end.
4. Arraylist is not suitable for inserting data/element in the beginning or middle

Linked List – It’s also a implementation of List interface. It allows duplicate values, null many times

1. In linked list, the elements/data is not stored in continuous memory location.



1. The easiest operation in linked list is inserting in the beginning or middle bcos no need to perform multiple shift right operation.
2. The costlier operation in linkedlist, adding or removing data at the end.

Set(I)

1. It won’t allow duplicate values
2. Null is allowed only once
3. It will not maintain insertion order
4. HashSet & TreeSet (Stores the data in sorted order)

Map(I)

1. Stores the data/element in key,value pair
2. Key should be unique, value can have duplicate
3. It will not maintain the insertion order
4. Put() & get() is used to store/retrieve the data/element

Comparable (I)

1. It has a public int compareTo(Object obj) method that can be used for sorting the data in ascending or descending order. (Natural sort order – only one sort order operation can be performed)

Comparator (I)

1. It’s used for comparing user defined objects
2. It has a public int compare(Object obj1, Object obj2) method to sort on any property of object