**Serialization in java**:

1. ObjectOutputStream in Java can be used to convert an object to OutputStream. The process of converting object to stream is called serialization in java.
2. The object that we want to serialize should implement java.io.Serializable interface. Serializable is just a marker interface and doesn’t have any abstract method that we have to implement. We will get java.io.NotSerializableException if the class doesn’t implement Serializable interface. Something like below exception stack trace.
3. **ObjectOutputStream with transient:** If we don’t want some object property to be converted to stream, we have to use transient keyword for that. For example, just change the role property like below and it will not be saved.

Private transient String role;

1. **Class Refactoring with Serialization and serialVersionUID:** Serialization in java permits some changes in the java class if they can be ignored. Example – Adding new variable, changing the variables from transient to non-transient. But for all these changes to work, the java class should have **serialVersionUID** defined for the class

**Java 8 Interface:**

1. Prior to Java 8, we could have only method declarations in the interfaces. But from Java 8, we can have **default methods** and **static methods** in the interfaces.
2. We know that Java doesn’t allow us to extend multiple classes because it will result in the “Diamond Problem” where compiler can’t decide which superclass method to use. With the default methods, the diamond problem would arise for interfaces too. Because if a class is implementing both **Interface1** and **Interface2** and doesn’t implement the common default method, compiler can’t decide which one to choose. So if a class is implementing both the above interfaces, it will have to provide implementation for the common default method.
3. **Java Functional Interfaces:** An interface with exactly one abstract method is known as Functional Interface.
4. **@FunctionalInterface** annotation is a facility to avoid accidental addition of abstract methods in the functional interfaces. It’s optional but good practice to use it.
5. FunctionalInterface enables us to use lambda expressions to instantiate them. A new package **java.util.function** with bunch of functional interfaces are added to provide target types for lambda expressions and method references.
6. Some of the commonly used functional interfaces in the Java 8 Stream API methods are:

**JAVA 8 Stream:**

1. A collection is an in-memory data structure to hold values and before we start using collection, all the values should have been populated. Whereas a java Stream is a data structure that is computed on-demand.
2. Java Stream doesn’t store data, it operates on the source data structure (collection and array) and produce pipelined data that we can use and perform specific operations. Such as we can create a stream from the list and filter it based on a condition.

|  |
| --- |
| private static int sumStream(List<Integer> list) {  **return list.stream().filter(i -> i > 10).mapToInt(i -> i).sum();**  } |

1. Java Stream operations use functional interfaces,
2. Java 8 Stream support sequential as well as parallel processing, parallel processing can be very helpful in achieving high performance for large collections

**Difference between HashMap and Hashtable:**

HashMap and Hashtable both implements Map interface and looks similar, however there are following difference between HashMap and Hashtable.

1. HashMap allows null key and values whereas Hashtable doesn’t allow null key and values.
2. Hashtable is synchronized but HashMap is not synchronized. So HashMap is better for single threaded environment, Hashtable is suitable for multi-threaded environment.
3. LinkedHashMap was introduced in Java 1.4 as a subclass of HashMap, so in case you want iteration order, you can easily switch from HashMap to LinkedHashMap but that is not the case with Hashtable whose iteration order is unpredictable.
4. HashMap provides Set of keys to iterate and hence it’s fail-fast but Hashtable provides Enumeration of keys that doesn’t support this feature.
5. Hashtable is considered to be legacy class and if you are looking for modifications of Map while iterating, you should use ConcurrentHashMap.

**Thread:**