10TH JULY GENERICS AND FILE-HANDLING

1. **What is Input and Output Stream in Java?**

   Ans: In Java, an input stream is used to read data from a source, such as a file or network, while an output stream is used to write data to a destination, like a file or network connection.

2. What are the methods of OutputStream?

Ans: The OutputStream class in Java provides various methods for writing data to an output stream. Some commonly used methods include write(int b) to write a single byte, write(byte[] b) to write an array of bytes, and flush() to ensure that any buffered data is written to the output stream.

3. What is serialization in Java?

Ans: Serialization in Java refers to the process of converting an object into a stream of bytes so that it can be saved to a file, sent over a network, or stored in a database. It allows objects to be persisted and reconstructed later.

4. What is the Serializable interface in Java?

Ans: The Serializable interface in Java is a marker interface that allows a class to be serialized. By implementing this interface, a class indicates that its objects can be converted into a stream of bytes and hence can be serialized.

5. What is deserialization in Java?

Ans: Deserialization in Java is the reverse process of serialization. It involves converting a stream of bytes back into an object, allowing the object to be reconstructed or used in memory.

6. How is serialization achieved in Java?

Ans: Serialization in Java is achieved by implementing the Serializable interface in the class whose objects need to be serialized. The ObjectOutputStream class is then used to write the objects to an output stream, which can be a file or a network connection.

7. How is deserialization achieved in Java?

Ans: Deserialization in Java is achieved by using the ObjectInputStream class to read the serialized objects from an input stream. The deserialized objects can then be used in the program as regular Java objects.

8. How can you avoid certain member variables of class from getting Serialized?

Ans: To avoid certain member variables of a class from being serialized, you can mark them as transient. Transient variables are not included in the serialization process and are skipped during the serialization and deserialization.

9. What classes are available in the Java I0 File Classes API?

Ans: The Java I/O File Classes API provides various classes for performing input and output operations on files. Some of the important classes in this API include File, FileInputStream, FileOutputStream, FileReader, FileWriter, and RandomAccessFile.

10. What is Difference between Externalizable and Serialization interface.

Ans: The Externalizable interface is an extension of the Serializable interface in Java. While Serializable provides automatic serialization and deserialization, Externalizable gives more control over the serialization process by allowing custom serialization and deserialization methods to be implemented.

11. What are Generics in Java?

Ans: Generics in Java are a way to create classes, interfaces, and methods that can work with different types without sacrificing type safety. They provide the ability to parameterize types and enable the reuse of code with different data types.

12. What are the benefits of using Generics in Java?

Ans: The benefits of using generics in Java include increased type safety, improved code readability and maintainability, and the ability to write reusable code that can operate on different types without the need for explicit type casting.

13. What is a Generic Class in Java?

Ans: A generic class in Java is a class that is parameterized with one or more type parameters. These type parameters are placeholders for actual types that will be specified when creating an instance of the generic class. The type parameters allow the class to work with different types.

14. What is a Type Parameter in Java Generics?

Ans: A type parameter in Java generics is a placeholder for a specific type that will be provided when using a generic class or method. It allows the generic code to operate on different types without specifying the actual type until the code is used.

15. What is a Generic Method in Java?

Ans: A generic method in Java is a method that is parameterized with one or more type parameters. These type parameters can be used to define the types of the method's parameters, return type, or local variables within the method. Generic methods allow flexibility in working with different types.

16. What is the difference between ArrayList and ArrayList‹T›?

Ans: There is no difference between ArrayList and ArrayList<T> in terms of functionality. Both represent a dynamic array that can grow and shrink in size. However, ArrayList<T> is a generic class, where the type parameter T specifies the type of elements that the ArrayList can hold. The use of generics allows compile-time type checking and helps in avoiding type casting errors.