1. What is Input and Output Stream in Java

- Input Stream: A stream that reads data from a source (e.g., file, network).

- Output Stream: A stream that writes data to a destination (e.g., file, network).

2. What are the methods of OutputStream

- void write(int b): Writes the specified byte to the output stream.

- void write(byte[] b): Writes b.length bytes from the specified byte array to the output stream.

- void write(byte[] b, int off, int len): Writes len bytes from the specified byte array starting at offset off to the output stream.

- void flush(): Flushes the output stream and forces any buffered output bytes to be written out.

- void close(): Closes the output stream and releases any system resources associated with it.

3. What is serialization in Java

- Serialization is the process of converting an object's state into a byte stream to save it to a file or transmit it over a network.

4. What is the Serializable interface in Java

- Serializable is a marker interface in Java that indicates a class can be serialized. It has no methods; implementing it allows an object to be serialized.

5. What is deserialization in Java

- Deserialization is the process of converting a byte stream back into a copy of the original object.

6. How is serialization achieved in Java

- Serialization is achieved by implementing the Serializable interface and using ObjectOutputStream to write the object to an output stream.

Example:

```
import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;
```

```
class MyClass implements Serializable {
    int a;
    String b;

    MyClass(int a, String b) {
        this.a = a;
        this.b = b;
    }
}

public class SerializeDemo {
    public static void main(String[] args) {
        MyClass obj = new MyClass(1, "example");
        try {
            FileOutputStream fileOut = new FileOutputStream("object.ser");
            ObjectOutputStream out = new ObjectOutputStream(fileOut);
            out.writeObject(obj);
            out.close();
            fileOut.close();
            System.out.println("Serialized data is saved in object.ser");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

7. How is deserialization achieved in Java

Deserialization is achieved using ObjectInputStream to read the object from an input stream.

Example:

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class DeserializeDemo {
    public static void main(String[] args) {
        MyClass obj = null;
        try {
            FileInputStream fileIn = new FileInputStream("object.ser");
            ObjectInputStream in = new ObjectInputStream(fileIn);
            obj = (MyClass) in.readObject();
            in.close();
            fileIn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("Deserialized Object: " + obj.a + ", " + obj.b);
    }
}
```

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

Use the transient keyword to mark member variables that should not be serialized.

Example:

```
class MyClass implements Serializable {
    int a;
    transient String b; // 'b' will not be serialized

    MyClass(int a, String b) {
        this.a = a;
        this.b = b;
```

```
    }
}
```

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

- File
- FileReader
- FileWriter
- BufferedReader
- BufferedWriter
- FileInputStream
- FileOutputStream
- RandomAccessFile

10. What is Difference between Externalizable and Serialization interface.

Serializable	Externalizable
It is a marker interface that allows an object to be serialized automatically.	It provides more control over serialization by requiring the class to implement writeExternal() and readExternal() methods.
Serializable is easier to implement but offers less control.	Externalizable provides full control over the serialization process but requires more effort.

: