1. **What is I/O redirection in java? Give a code example.**

Redirecting standard I/O

The Java System class allows you to redirect the standard input, output, and error I/O streams using simple static method calls:

setIn(InputStream)
setOut(PrintStream)
setErr(PrintStream**)**

```
import java.io.*;

public class Redirecting {
    // Throw exceptions to console:
    public static void main(String[] args)
            throws IOException {
        PrintStream console = System.out;
        BufferedInputStream in = new BufferedInputStream(
                new FileInputStream("Redirecting.java"));
        PrintStream out = new PrintStream(
                new BufferedOutputStream(
                        new FileOutputStream("test.out")));
        System.setIn(in);
        System.setOut(out);
        System.setErr(out);
        BufferedReader br = new BufferedReader(
                new InputStreamReader(System.in));
        String s;
        while((s = br.readLine()) != null)
            System.out.println(s);
        out.close(); // Remember this!
        System.setOut(console);
    }
}
```

2. **What is Serialization in java? Give a code example.**
   Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

Code :

```java
import java.io.*;

class Demo implements java.io.Serializable
{
    public int a;
    public String b;
    public Demo(int a, String b)
    {
        this.a = a;
        this.b = b;
    }
}
```

```java
class Test
{
    public static void main(String[] args)
    {
        Demo object = new Demo( a: 1,  b: "geeksforgeeks");
        String filename = "file.ser";
        try
        {
            FileOutputStream file = new FileOutputStream(filename);
            ObjectOutputStream out = new ObjectOutputStream(file);
            out.writeObject(object);
            out.close();
            file.close();
            System.out.println("Object has been serialized");
        }
        catch(IOException ex)
        {
            System.out.println("IOException is caught");
        }
        Demo object1 = null;
        catch(ClassNotFoundException ex)
        {
            System.out.println("ClassNotFoundException is caught");
        }

    }
}
```

3. **Compare between serial access and random-access files in terms of their definition, usage, advantages, and disadvantages.**
    - **Sequential Access**

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation -read next- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the -write next- append to the end of the file and advance to the newly written material.

Key points:

Data is accessed one record right after another record in an order.

When we use read command, it move ahead pointer by one

When we use write command, it will allocate memory and move the pointer to the end of the file

Such a method is reasonable for tape.

- **Direct Access**

Another method is direct access method also known as relative access method. A filed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59, and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a relative block number, the first relative block of the file is 0 and then 1 and so on.

4. What is *JFileChooser* in java? Give a code example.

JFileChooser is a part of java Swing package. The java Swing package is part of JavaTM Foundation Classes(JFC) . JFC contains many features that help in building graphical user interface in java . Java Swing provides components such as buttons, panels, dialogs, etc . JFileChooser is a easy and an effective way to prompt the user to choose a file or a directory .
In this article we will see how to use JFileChooser in java swing .
Constructors of JFileChooser are :

1. JFileChooser() – empty constructor that points to user's default directory

```
import java.io.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.filechooser.*;
class filechooser extends JFrame implements ActionListener {
    public static void main(String args[]) {

        JFileChooser j = new JFileChooser();
        j.showSaveDialog(null);
    }
    }
```