**Question 1: JDBC Drivers & Database Connectivity Steps**

**Types of JDBC Drivers:**

1. Type 1: JDBC-ODBC Bridge Driver
2. Type 2: Native-API/Partly Java Driver
3. Type 3: Network Protocol/All Java Driver
4. Type 4: Thin Driver/Pure Java Driver

**Steps for Database Connectivity:**

1. Register the JDBC driver
2. Establish connection using DriverManager.getConnection()
3. Create Statement object
4. Execute SQL queries
5. Process results
6. Close connections

**Question 2: Three Classes in java.util Package**

1. **ArrayList**: Implements dynamic array, allows random access, and dynamic sizing.
2. **HashMap**: Implements Map interface using hash table, stores key-value pairs.
3. **Scanner**: Parses primitive types and strings from input streams.

**Question 3: File Copy Program**

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class FileCopy {
   public static void main(String[] args) {
      FileInputStream inputStream = null;
      FileOutputStream outputStream = null;

      try {
         // Specify the input and output file paths
         inputStream = new FileInputStream("source.txt");
         outputStream = new FileOutputStream("destination.txt");

         // Read bytes from input file and write to output file
         int byteData;
         while ((byteData = inputStream.read()) != -1) {
            outputStream.write(byteData);
         }

         System.out.println("File copied successfully!");

      } catch (IOException e) {
         System.out.println("Error copying file: " + e.getMessage());
      } finally {
         try {
            // Close resources
            if (inputStream != null) {
               inputStream.close();
            }
            if (outputStream != null) {
               outputStream.close();
            }
         } catch (IOException e) {
            System.out.println("Error closing streams: " + e.getMessage());
         }
```

```
    }
  }
}
```

**Question 4: Java Collections & Collection Interface Methods**

**Java Collections:** Collections are predefined data structures in Java that store and process data efficiently through interfaces and classes organized in a hierarchy.

**Collection Interface Methods:**

- add(E e): Adds an element
- remove(Object o): Removes an element
- size(): Returns number of elements
- isEmpty(): Checks if collection is empty
- contains(Object o): Checks if element exists
- iterator(): Returns an iterator
- clear(): Removes all elements

**Question 5: Object Serialization**

Object serialization is the process of converting Java objects into byte streams for persistent storage or transmission. The object's state is saved, and it can be deserialized later to recreate the object. Classes must implement the Serializable interface to enable serialization. The ObjectOutputStream and ObjectInputStream classes handle the serialization and deserialization processes, respectively. Serialization maintains object references, and fields marked as transient are not serialized.