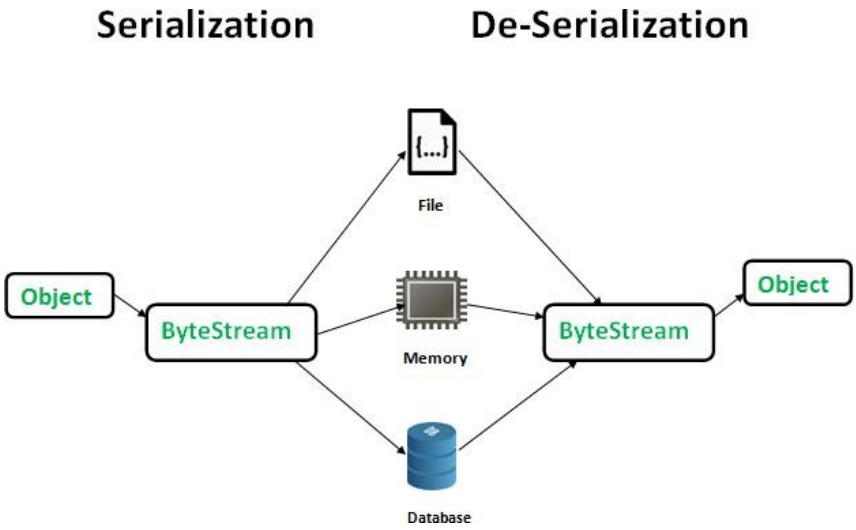
UNIT IV – SERIALIZATION AND DESERIALIZATION

Reading and writing objects using Serialization:

- ✓ In Java, serialization is the process of converting an object's state into a byte stream, while deserialization is the reverse process of recreating the object from that byte stream.
- ✓ This is commonly used to save and retrieve objects from files, send them over a network, or transfer them between systems.



Steps for reading and writing objects using serialization

1. Implement the Serializable interface:

- ✓ The class of any object you want to serialize must implement the java.io.Serializable interface.
- ✓ Serializable is a marker interface, meaning it has no methods to implement. It simply marks the class as eligible for serialization.
- ✓ If a superclass is serializable, all its subclasses are automatically serializable as well.

2. Define a serialVersionUID

- ✓ Add a private static final long serialVersionUID field to your class.
- ✓ This version number is used to verify that the sender and receiver of a serialized object have compatible class definitions.
- ✓ If you change your class and the serialVersionUID is different, a java.io.InvalidClassException is thrown during deserialization.

3. Mark fields as transient

- ✓ Use the transient keyword to exclude specific fields from the serialization process.

- ✓ This is useful for sensitive data (like passwords) or temporary fields that do not need to be saved to the file.
- ✓ Static fields are not part of an object's state, so they are not serialized by default. The transient keyword has no effect on them.

Example:

```
import java.io.*;
class Student implements Serializable
{
   String name;
   int age;
   Student(String name, int age)
   {
      this.name = name;
      this.age = age;
   }
}
public class SerializeDemo
{
   public static void main(String[] args)
   {
      try          // SERIALIZATION (Writing object to file)
      {
            Student s1 = new Student("John", 1);
            FileOutputStream fos = new FileOutputStream("student.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(s1); // Write object
            oos.close();
            fos.close();
            System.out.println("Object serialized and saved to student.ser");
      }
      catch (Exception e)
      {
            System.out.println(e);
      }
      try          // DESERIALIZATION (Reading object from file)
      {
            FileInputStream fis = new FileInputStream("student.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Student s2 = (Student) ois.readObject(); // Read object
            ois.close();
            fis.close();
```

```
                System.out.println("Object deserialized:");
                System.out.println("Name: " + s2.name);
                System.out.println("Age: " + s2.age);
            }
            catch (Exception e)
            {
                System.out.println(e);
            }
        }
    }
```

OUTPUT: