Serialisation and Deserialisation

Introduction

Serialization in Java is the process of converting an object's state into a byte stream, enabling the object to be easily saved to a file, sent over a network, or stored in a database. This byte stream can be reverted back into a copy of the object through deserialization.

Benefits

The scope of serialization and deserialization in Java provides several benefits that can be leveraged in various applications, particularly in areas involving object persistence, data interchange, and distributed computing.

Persistence:

- **Long-Term Storage**: Objects can be serialized to files or databases, allowing their state to be saved and restored later. This is useful for applications that need to save user settings, application state, or other data between sessions.

Communication:

- **Network Transmission**: Serialized objects can be sent over a network, enabling distributed applications to communicate complex data structures easily. This is

essential for remote method invocation (RMI), web services, and other network-based applications.

Caching:

- **Temporary Storage**: Objects can be serialized to memory (using byte arrays) for caching purposes, improving performance by avoiding repeated computations or data retrieval operations.

Deep Copy:

- **Cloning Objects**: Serialization and deserialization can be used to create deep copies of objects, ensuring that the copied object is entirely independent of the original, with no shared references.

Cross-Platform Interoperability:

- **Language Agnostic Data Exchange**: Serialized objects can be converted into a platform-independent format (e.g., JSON or XML), facilitating data exchange between applications written in different programming languages.

Session Management:

- **Web Applications**: Serialized objects can store user session data, allowing web applications to maintain state across multiple requests and sessions.

Logging and Debugging:

- **State Capture**: Serialization can capture the state of an object at a specific point in time, aiding in logging and debugging complex applications by preserving the exact state of objects for later analysis.

Mechanism of Serialization

Serialization is the process of converting an object's state into a byte stream so it can be easily saved to a file, sent over a network, or stored in a database.

Making a Class Serializable:

- Implement the `Serializable` interface in the class. This interface doesn't contain any methods.

```
class MyObject implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    private int value;
}
```

Writing an Object to a File:

- Use `ObjectOutputStream` to serialize an object and write it to a file.

```
FileOutputStream fileOut = new FileOutputStream("object.ser");
ObjectOutputStream out = new ObjectOutputStream(fileOut);
MyObject myObj = new MyObject("example", 123);
out.writeObject(myObj);
out.close();
fileOut.close();
```

Mechanism of Deserialization

Deserialization is the reverse process of converting a byte stream back into an object.

Reading an Object from a File:

- Use `ObjectInputStream` to read the serialized object from a file and convert it back into an object.

FileInputStream fileIn = new FileInputStream("object.ser");

ObjectInputStream in = new ObjectInputStream(fileIn);

MyObject deserializedObj = (MyObject) in.readObject();

in.close();

fileIn.close();

`serialVersionUID`

`serialVersionUID` is a unique identifier for each serializable class. It ensures that the class definition matches during serialization and deserialization.

1. **Purpose:**
    - **It helps maintain compatibility between different versions of a class. If a class changes, the `serialVersionUID` ensures that the old version of the class can still be deserialized.**
2. **Declaring `serialVersionUID`:**
    - **It is good practice to explicitly declare `serialVersionUID` in your class.**

private static final long serialVersionUID = 1L;

Refer to this repo for test project that explains the above code:

https://github.com/ShenanVindinu/serialization_and_deserialization.git