Lecture - 18

Object Serialization

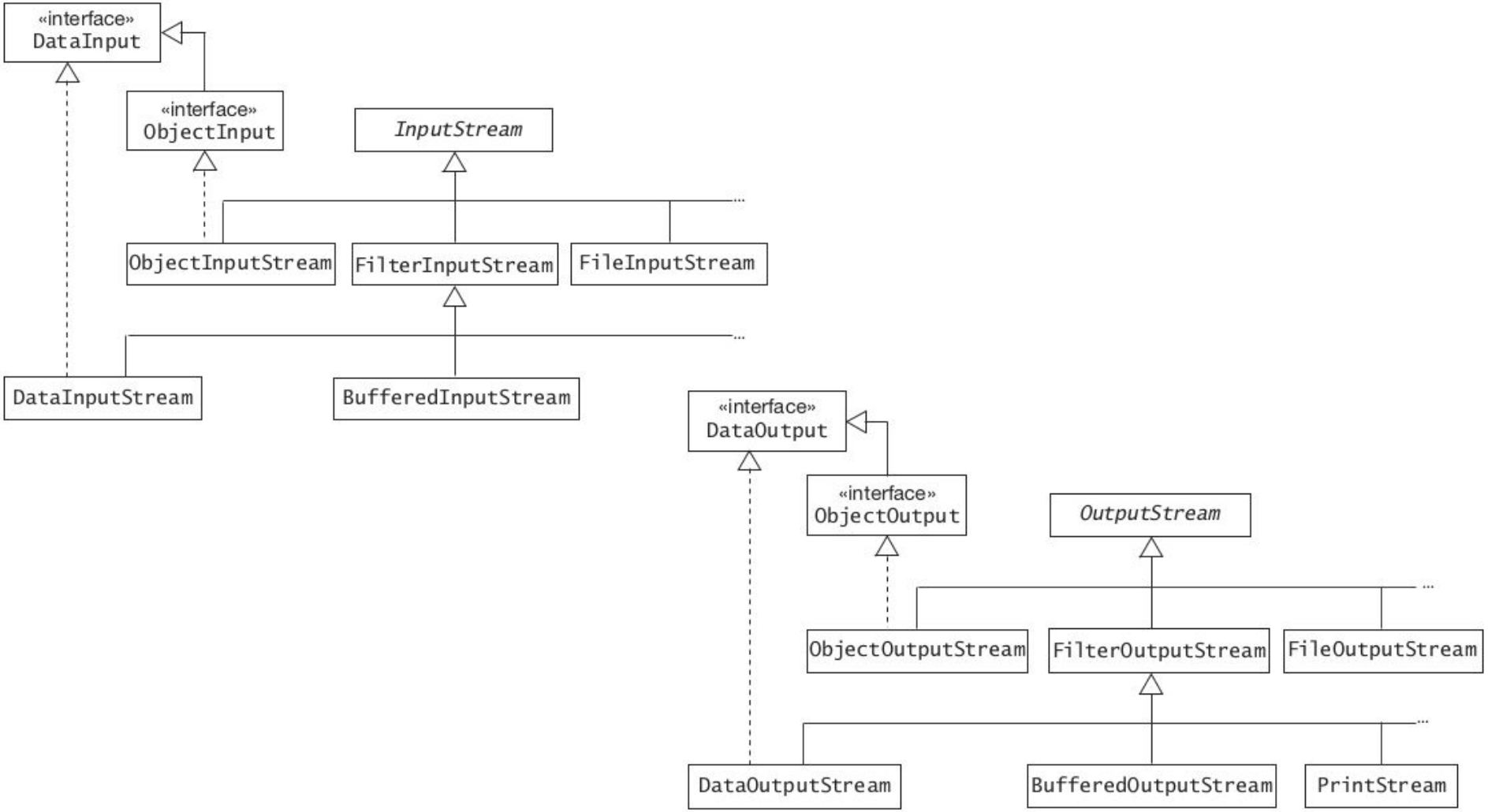
Arpit Rana

19th April 2022

Serialization

Object serialization allows an object to be transformed into a sequence of bytes that can later be re-created (deserialized) into the original object.

- After deserialization, the object has the same state as it had when it was serialized, barring any data members that were not serializable.
- This mechanism is generally known as *persistence*.

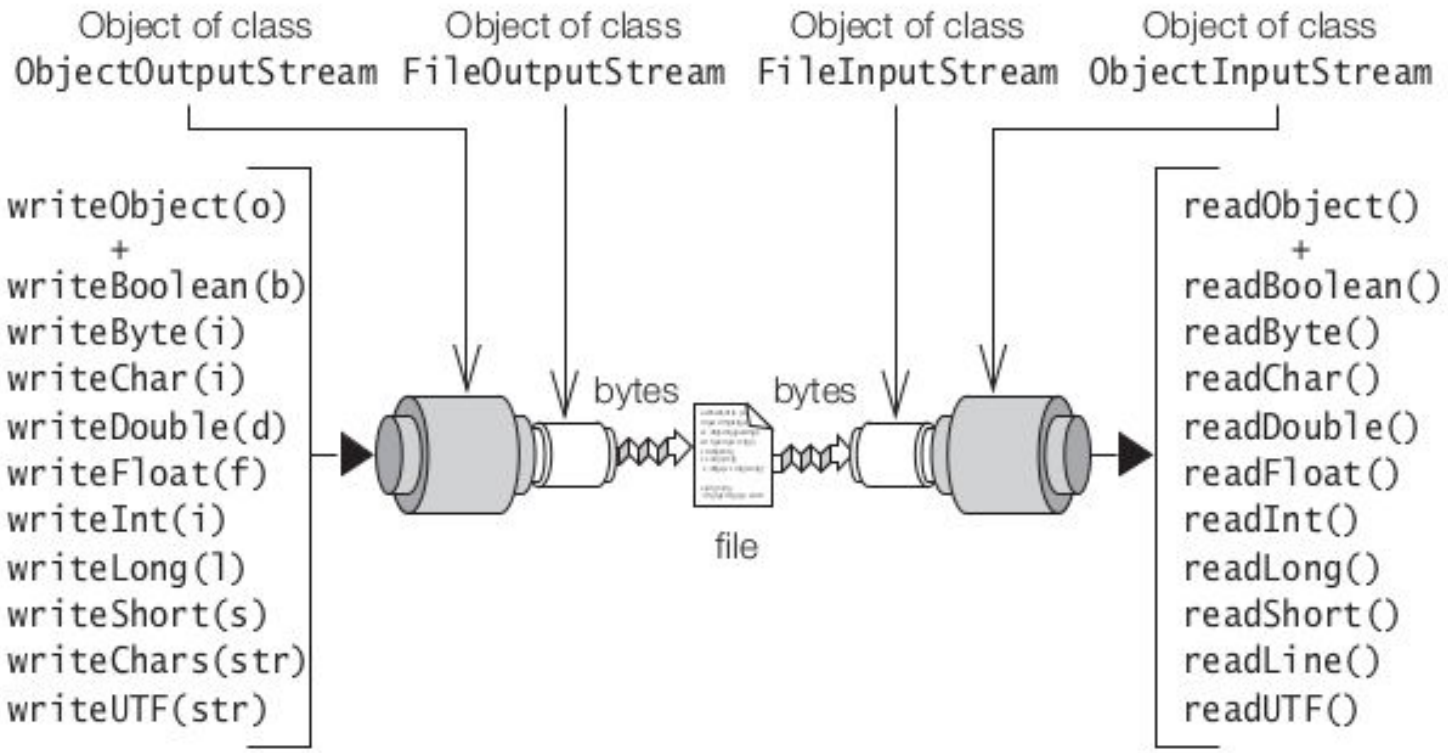

«interface»
DataInput
«interface»
ObjectInput
InputStream
...
ObjectInputStream
FilterInputStream
FileInputStream
...
DataInputStream
BufferedInputStream
«interface»
DataOutput
«interface»
ObjectOutput
OutputStream
...
ObjectOutputStream
FilterOutputStream
FileOutputStream
...
DataOutputStream
BufferedOutputStream
PrintStream

Serialization

Java provides this facility through the ObjectInput and ObjectOutput interfaces, which allow the reading and writing of objects from and to streams.

- These two interfaces extend the DataInput and DataOutput interfaces, respectively.
- The ObjectOutputStream class and the ObjectInputStream class implement the ObjectOutput interface and the ObjectInput interface, respectively.
- These classes provide methods to write and read binary representation of objects as well as Java primitive values.

Object Stream Chaining



The ObjectOutputStream Class

The class ObjectOutputStream can write objects to any stream that is a subclass of the OutputStream, e.g., to a file or a network connection (socket).

An Object OutputStream must be chained to an OutputStream using the following constructor:

```
ObjectOutputStream(OutputStream out) throws IOException
```

The ObjectOutputStream Class

The class ObjectOutputStream can write objects to any stream that is a subclass of the OutputStream, e.g., to a file or a network connection (socket).

In order to store objects in a file and thus provide persistent storage for objects, an ObjectOutputStream can be chained to a FileOutputStream:

```
FileOutputStream outputFile = new FileOutputStream("obj-storage.dat");
ObjectOutputStream outputStream = new ObjectOutputStream(outputFile);
```

The ObjectOutputStream Class

The class ObjectOutputStream can write objects to any stream that is a subclass of the OutputStream, e.g., to a file or a network connection (socket).

Objects can be written to the stream using the writeObject() method of the ObjectOutputStream class:

```
final void writeObject(Object obj) throws IOException
```

The ObjectOutputStream Class

The writeObject() method can be used to write any object to a stream, including strings and arrays, as long as the object implements the java.io.Serializable interface.

- This is a marker interface with no methods.
- The String class, the primitive wrapper classes and all array types implement the Serializable interface.
- The following information is included when an object is serialized:
  - the class information needed to reconstruct the object.
  - the values of all serializable non-transient and non-static members, including those that are inherited.

Note also that objects of subclasses that extend a serializable class are always serializable.

The ObjectInputStream Class

An ObjectInputStream is used to restore (deserialize) objects that have previously been serialized using an ObjectOutputStream.

An ObjectInputStream must be chained to an InputStream, using the following constructor:

```
ObjectInputStream(InputStream in) throws IOException,
                                         StreamCorruptedException
```

The ObjectInputStream Class

In order to restore objects from a file, an ObjectInputStream can be chained to a FileInputStream:

```
FileInputStream inputFile = new FileInputStream("obj-storage.dat");
ObjectInputStream inputStream = new ObjectInputStream(inputFile);
```

The method readObject() of the ObjectInputStream class is used to read an object from the stream:

```
final Object readObject() throws OptionalDataException,
                                 ClassNotFoundException, IOException
```

Object Serialization Example

```
//Reading and Writing Objects
import java.io.EOFException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Arrays;

public class ObjectSerializationDemo {
  void writeData() {                                    // (1)
    try {
      // Set up the output stream:
      FileOutputStream outputFile = new FileOutputStream("obj-storage.dat");
      ObjectOutputStream outputStream = new ObjectOutputStream(outputFile);

      // Write data:
      String[] strArray = {"Seven", "Eight", "Six"};
      long num = 2008;
      int[] intArray = {1, 3, 1949};
      String commonStr = strArray[2];                   // "Six"
      outputStream.writeObject(strArray);
      outputStream.writeLong(num);
      outputStream.writeObject(intArray);
      outputStream.writeObject(commonStr);

      // Flush and close the output stream:
      outputStream.flush();
      outputStream.close();
    } catch (FileNotFoundException e) {
      System.err.println("File not found: " + e);
    } catch (IOException e) {
      System.err.println("Write error: " + e);
    }
  }
```

```
void readData() {                                          // (2)
  try {
    // Set up the input stream:
    FileInputStream inputFile = new FileInputStream("obj-storage.dat");
    ObjectInputStream inputStream = new ObjectInputStream(inputFile);

    // Read the data:
    String[] strArray = (String[]) inputStream.readObject();
    long num = inputStream.readLong();
    int[] intArray = (int[]) inputStream.readObject();
    String commonStr = (String) inputStream.readObject();

    // Write data to the standard output stream:
    System.out.println(Arrays.toString(strArray));
    System.out.println(Arrays.toString(intArray));
    System.out.println(commonStr);

    // Close the stream:
    inputStream.close();
  } catch (FileNotFoundException e) {
    System.err.println("File not found: " + e);
  } catch (EOFException e) {
    System.err.println("End of stream: " + e);
  } catch (IOException e) {
    System.err.println("Read error: " + e);
  } catch (ClassNotFoundException e) {
    System.err.println("Class not found: " + e);
  }
}
```

```
  public static void main(String[] args) {
    ObjectSerializationDemo demo = new ObjectSerializationDemo();
    demo.writeData();
    demo.readData();
  }
}
```

Output from the program:

```
[Seven, Eight, Six]
[1, 3, 1949]
Six
```

Next lecture -

Customizing Object Serialization