



# **CS205 Object Oriented Programming in Java**

## **Module 3 - More features of Java (Part 4)**

**Prepared by**

**Renetha J.B.**

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

# Topics



- **More features of Java :**
  - **Input / Output:**
    - ✓ Object Streams and Serialization

# Object Streams and Serialization

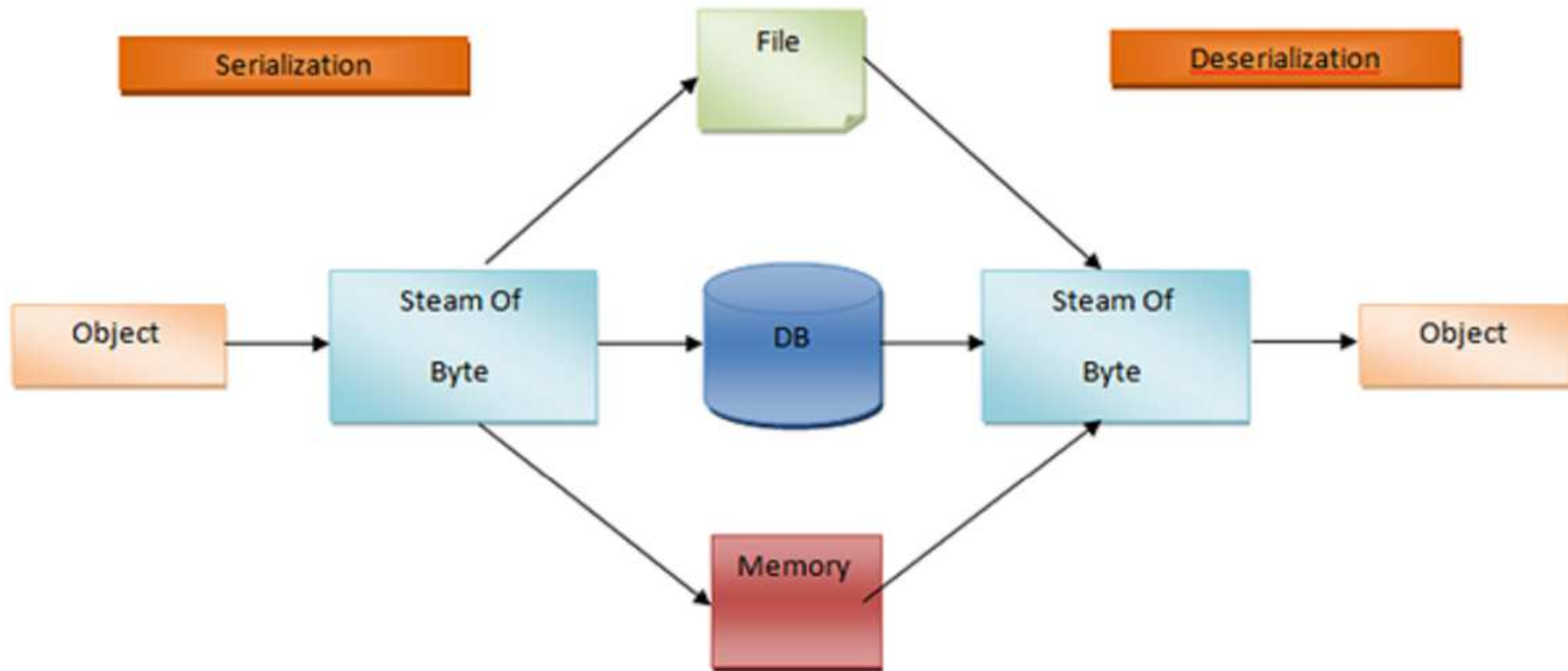


- **Object streams** support I/O(input-output) of objects
- The object stream classes are
  - **ObjectInputStream**
  - **ObjectOutputStream**

# Serialization



- **Serialization** is the process of writing(converting) the state of an object to a byte stream.
  - This is useful when we want to
    - save(store) the state of the program to a *persistent(permanent)* storage area, such as a file or
    - when we want to send it over network.
- Later we can restore these objects by using the process of *deserialization*.
  - **Deserialization** converts byte streams into object.



# Serialization(contd.)



- Serialization is also needed to implement Remote Method Invocation (RMI).
  - RMI allows a Java object on one machine to invoke a method of a Java object on a different machine.
  - The *sending machine* serializes the object and transmits it. The *receiving machine* deserializes it.

# Serialization(contd.)



- If we attempt to serialize an object at the top of an object graph,
  - all of the other referenced objects are recursively located and serialized.
- Similarly, during the process of deserialization, all of these objects and their references are correctly restored when deserialization is done at the top.

# Serialization(contd.)



- Interfaces and classes that support serialization are:
  - **Serializable**
  - **Externalizable**



# Serialization(contd.)



- **Serializable**
  - Only an **object that implements** the **Serializable** interface can be **saved and restored** by the serialization facilities.
  - The **Serializable** interface defines no members.
  - It is simply used to indicate that a class may be serialized.
  - If a class is serializable, all of its subclasses are also serializable.
  - Variables that are declared as **transient** and **static variables** are not saved by the serialization facilities.

# Externalizable



- Much of the work to save and restore the state of an object occurs automatically.
  - The programmer may need to have control over these processes.
  - it may be desirable to use compression or encryption techniques.
- The **Externalizable** interface is designed for these situations.
- The **Externalizable** interface defines two methods:

void **readExternal**(ObjectInput *inStream*) throws IOException,  
ClassNotFoundException

void **writeExternal**(ObjectOutput *outStream*) throws IOException

- *inStream* is the byte stream from which the object is to be read
- *outStream* is the byte stream to which the object is to be written

*Prepared by Renetha J.B*

# ObjectOutput



- The **ObjectOutput** interface **extends** the **DataOutput** interface and supports object serialization.
- It defines the methods such as **writeObject( )**
- **writeObject( )** method is called to **serialize** an object.
- All of these methods will throw an **IOException** on error conditions

# ObjectOutput-methods



Method	Description
<code>void close( )</code>	Closes the invoking stream. Further write attempts will generate an <b>IOException</b> .
<code>void flush( )</code>	Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
<code>void write(byte <i>buffer</i>[ ])</code>	Writes an array of bytes to the invoking stream.
<code>void write(byte <i>buffer</i>[ ], int <i>offset</i>, int <i>numBytes</i>)</code>	Writes a subrange of <i>numBytes</i> bytes from the array <i>buffer</i> , beginning at <i>buffer[offset]</i> .
<code>void write(int <i>b</i>)</code>	Writes a single byte to the invoking stream. The byte written is the low-order byte of <i>b</i> .
<code>void writeObject(Object <i>obj</i>)</code>	Writes object <i>obj</i> to the invoking stream.

---

The Methods Defined by **ObjectOutput**

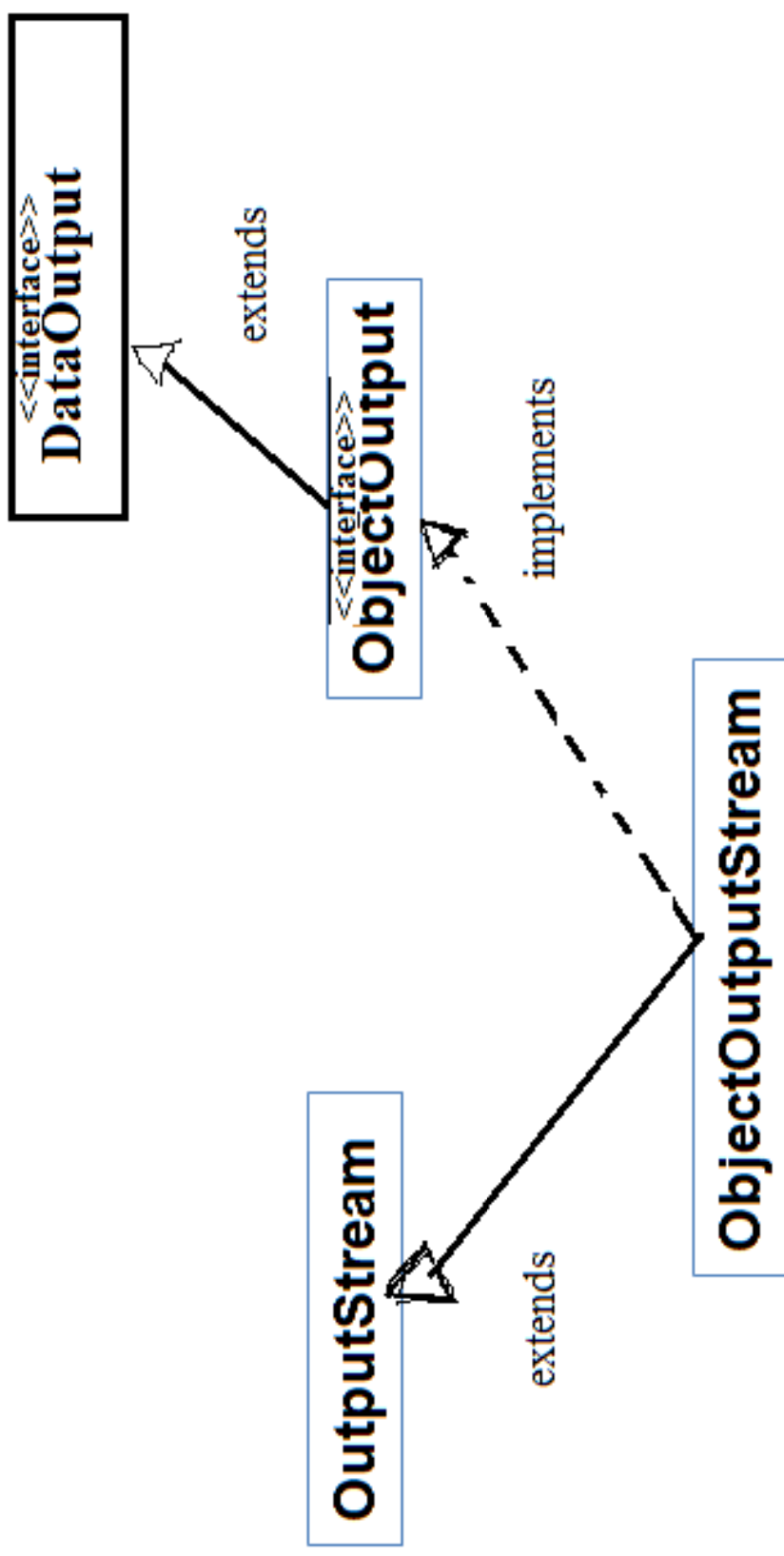
# ObjectOutputStream



- The **ObjectOutputStream** class **extends** the **OutputStream** class and **implements** the **ObjectOutput** interface.
- It is responsible for **writing objects to a stream**.
- A constructor of this class is

**ObjectOutputStream**(*OutputStream outputStream*) *throws IOException*

- The argument *outputStream* is the output stream to which serialized objects will be written.
- Methods in this class will throw an **IOException** on error conditions.
- There is also an inner class to **ObjectOutputStream** called **PutField**.
  - It facilitates the writing of persistent fields.



# ObjectOutputStream-methods

Method	Description
<code>void close( )</code>	Closes the invoking stream. Further write attempts will generate an <b>IOException</b> .
<code>void flush( )</code>	Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
<code>void write(byte <i>buffer</i> [ ])</code>	Writes an array of bytes to the invoking stream.
<code>void write(byte <i>buffer</i> [ ], int <i>offset</i>, int <i>numBytes</i>)</code>	Writes a subrange of <i>numBytes</i> bytes from the array <i>buffer</i> , beginning at <i>buffer[offset]</i> .
<code>void write(int <i>b</i>)</code>	Writes a single <b>byte</b> to the invoking stream. The byte written is the low-order byte of <i>b</i> .
<code>void writeBoolean(boolean <i>b</i>)</code>	Writes a <b>boolean</b> to the invoking stream.
<code>void writeByte(int <i>b</i>)</code>	Writes a <b>byte</b> to the invoking stream. The byte written is the low-order byte of <i>b</i> .
<code>void writeBytes(String <i>str</i>)</code>	Writes the bytes representing <i>str</i> to the invoking stream.
<code>void writeChar(int <i>c</i>)</code>	Writes a <b>char</b> to the invoking stream.
<code>void writeChars(String <i>str</i>)</code>	Writes the characters in <i>str</i> to the invoking stream.
<code>void writeDouble(double <i>d</i>)</code>	Writes a <b>double</b> to the invoking stream.
<code>void writeFloat(float <i>f</i> )</code>	Writes a <b>float</b> to the invoking stream.
<code>void writeInt(int <i>i</i>)</code>	Writes an <b>int</b> to the invoking stream.
<code>void writeLong(long <i>l</i>)</code>	Writes a <b>long</b> to the invoking stream.
<code>final void writeObject(Object <i>obj</i>)</code>	Writes <i>obj</i> to the invoking stream.
<code>void writeShort(int <i>i</i>)</code>	Writes a <b>short</b> to the invoking stream.

# ObjectInput



- The **ObjectInput** interface **extends** the **DataInput** interface and defines the method such as **readObject( )** method.
- This is called to **deserialize** an object.
- All of these methods will throw an **IOException** on error conditions.
- The **readObject( )** method can also throw **ClassNotFoundException**



# ObjectInput-methods



Method	Description
<code>int available( )</code>	Returns the number of bytes that are now available in the input buffer.
<code>void close( )</code>	Closes the invoking stream. Further read attempts will generate an <b>IOException</b> .
<code>int read( )</code>	Returns an integer representation of the next available byte of input. -1 is returned when the end of the file is encountered.
<code>int read(byte <i>buffer</i>[ ])</code>	Attempts to read up to <i>buffer.length</i> bytes into <i>buffer</i> , returning the number of bytes that were successfully read. -1 is returned when the end of the file is encountered.
<code>int read(byte <i>buffer</i>[ ], int <i>offset</i>, int <i>numBytes</i>)</code>	Attempts to read up to <i>numBytes</i> bytes into <i>buffer</i> starting at <i>buffer[offset]</i> , returning the number of bytes that were successfully read. -1 is returned when the end of the file is encountered.
<code>Object readObject( )</code>	Reads an object from the invoking stream.
<code>long skip(long <i>numBytes</i>)</code>	Ignores (that is, skips) <i>numBytes</i> bytes in the invoking stream, returning the number of bytes actually ignored.

# ObjectInputStream

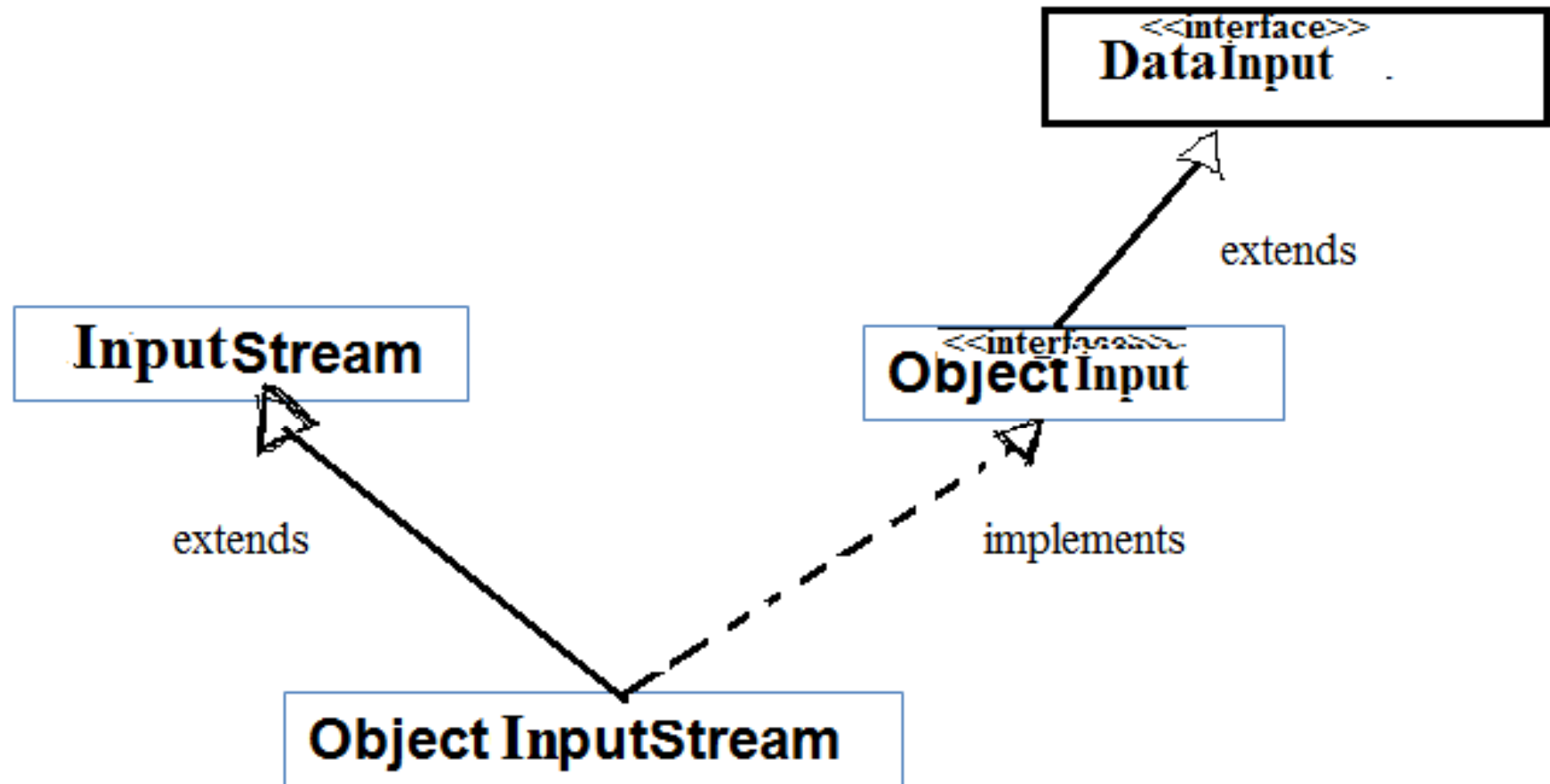


- The **ObjectInputStream** class **extends** the **InputStream** class and **implements** the **ObjectInput** interface.
- **ObjectInputStream** is responsible for **reading objects from a stream**.
- A constructor of this class is

**ObjectInputStream**(**InputStream** *inStream*) throws **IOException**

- The argument *inStream* is the input stream from which serialized objects should be read.

- The methods will throw an **IOException** on error conditions.
- The **readObject()** method can also throw **ClassNotFoundException**.
- There is also an inner class to **ObjectInputStream** called **GetField**. It facilitates the reading of persistent fields



# ObjectInputStream-methods



Method	Description
<code>int available( )</code>	Returns the number of bytes that are now available in the input buffer.
<code>void close( )</code>	Closes the invoking stream. Further read attempts will generate an <b>IOException</b> .
<code>int read( )</code>	Returns an integer representation of the next available byte of input. -1 is returned when the end of the file is encountered.
<code>int read(byte <i>buffer</i>[ ], int <i>offset</i>, int <i>numBytes</i>)</code>	Attempts to read up to <i>numBytes</i> bytes into <i>buffer</i> starting at <i>buffer[offset]</i> , returning the number of bytes successfully read. -1 is returned when the end of the file is encountered.
<code>boolean readBoolean( )</code>	Reads and returns a <b>boolean</b> from the invoking stream.
<code>byte readByte( )</code>	Reads and returns a <b>byte</b> from the invoking stream.
<code>char readChar( )</code>	Reads and returns a <b>char</b> from the invoking stream.
<code>double readDouble( )</code>	Reads and returns a <b>double</b> from the invoking stream.
<code>float readFloat( )</code>	Reads and returns a <b>float</b> from the invoking stream.
<code>void readFully(byte <i>buffer</i>[ ])</code>	Reads <i>buffer.length</i> bytes into <i>buffer</i> . Returns only when all bytes have been read.
<code>void readFully(byte <i>buffer</i>[ ], int <i>offset</i>, int <i>numBytes</i>)</code>	Reads <i>numBytes</i> bytes into <i>buffer</i> starting at <i>buffer[offset]</i> . Returns only when <i>numBytes</i> have been read.
<code>int readInt( )</code>	Reads and returns an <b>int</b> from the invoking stream.
<code>long readLong( )</code>	Reads and returns a <b>long</b> from the invoking stream.
<code>final Object readObject( )</code>	Reads and returns an object from the invoking stream.
<code>short readShort( )</code>	Reads and returns a <b>short</b> from the invoking stream.
<code>int readUnsignedByte( )</code>	Reads and returns an unsigned <b>byte</b> from the invoking stream.
<code>int readUnsignedShort( )</code>	Reads and returns an unsigned <b>short</b> from the invoking stream.

Figure 48-2 Commonly Used Methods Defined by ObjectInputStream

# Reference



- **Herbert Schildt, Java: The Complete Reference, 8/e, Tata McGraw Hill, 2011.**