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**

till now we looked byte stram and character stream

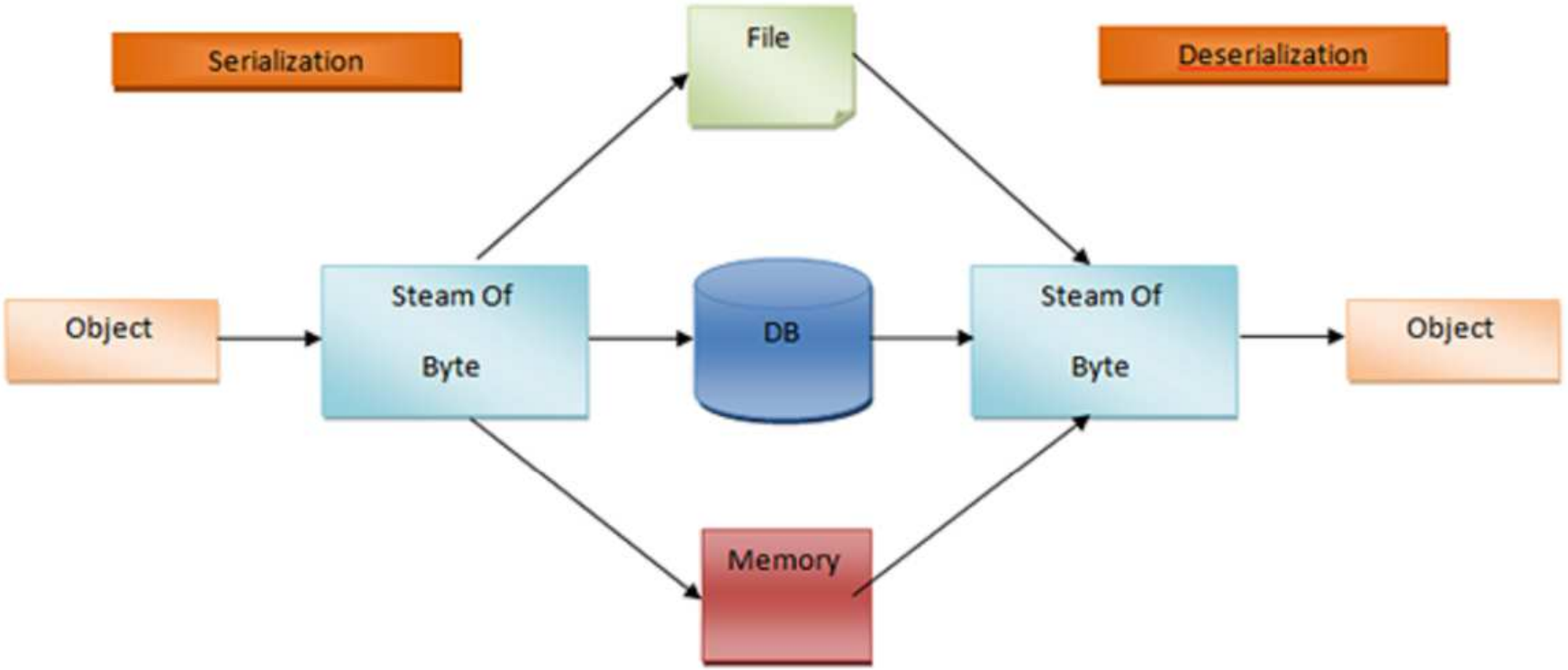
  - **ObjectOutputStream**

Serialization

- **Serialization** is the process of writing(converting) the state of an object to a byte stream.

  kore object ondu athil oronnilum data ondu ee data a object inte lifetimeil matre nammaku access cheyan pattu eethine overcome cheyan vendiyanu object inte state ne nammal streamilekku ezuthunne so that we could use some methods and store it on storage places

  - 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 c**onverts byte streams into object.


Serialization
File
Deserialization
Object
Steam Of
Byte
DB
Steam Of
Byte
Object
Memory

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

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

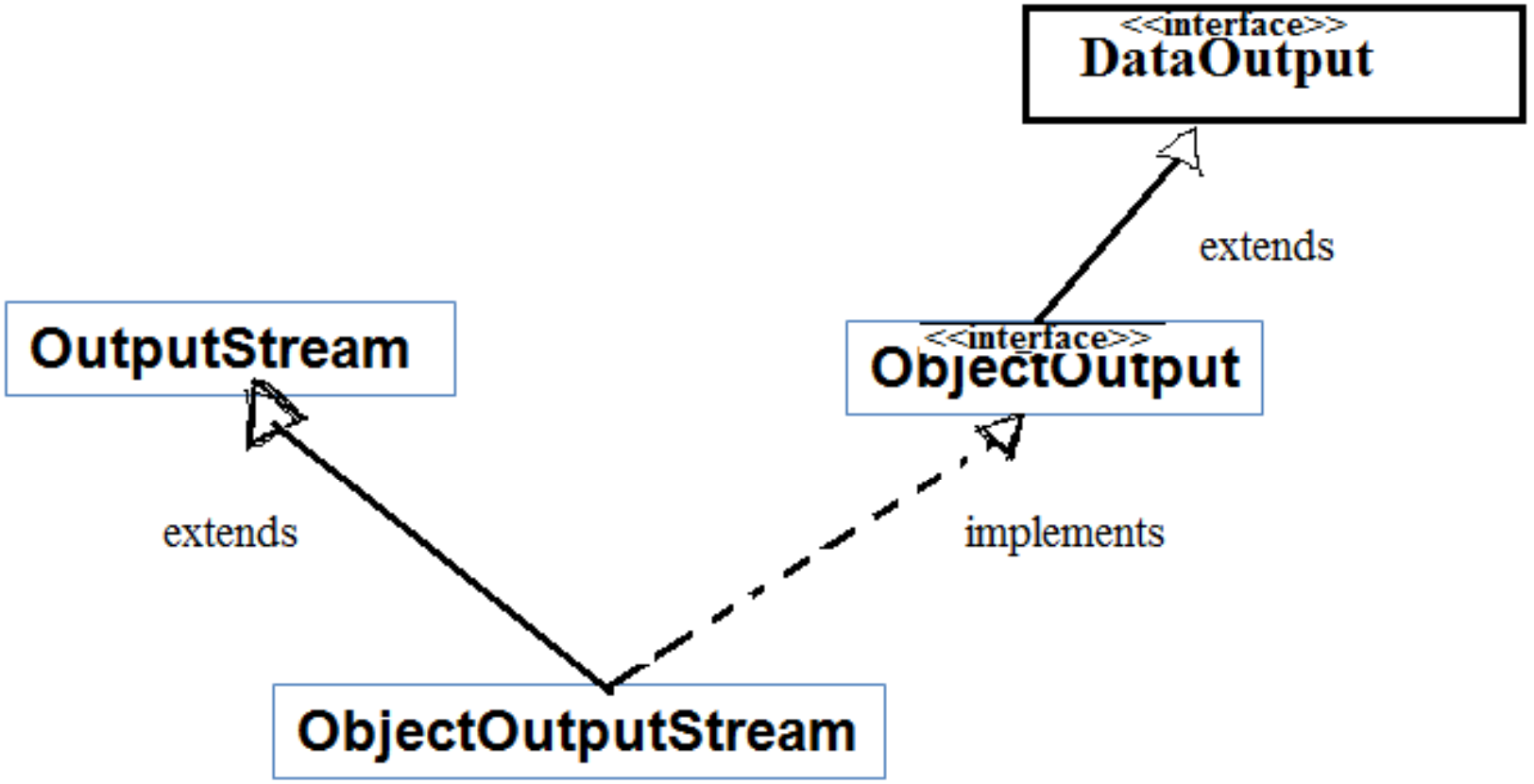
The Methods Defined by **ObjectOutput**

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 *outStream) throws IOException*

  - The argument outStream 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 **ObjectOuputStream** called **PutField.**
  - It facilitates the writing of persistent fields.


<<interface>>
DataOutput
extends
OutputStream
<<interface>>
ObjectOutput
extends
implements
ObjectOutputStream

ObjectOutputStream-methods

Method	Description
void close( )	Closes the invoking stream. Further write attempts will generate an **IOException**.
void flush( )	Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
void write(byte *buffer*[ ])	Writes an array of bytes to the invoking stream.
void write(byte *buffer*[ ], int *offset*, int *numBytes*)	Writes a subrange of *numBytes* bytes from the array *buffer*, beginning at *buffer*[*offset*].
void write(int *b*)	Writes a single **byte** to the invoking stream. The byte written is the low-order byte of *b*.
void writeBoolean(boolean *b*)	Writes a **boolean** to the invoking stream.
void writeByte(int *b*)	Writes a **byte** to the invoking stream. The byte written is the low-order byte of *b*.
void writeBytes(String *str*)	Writes the bytes representing *str* to the invoking stream.
void writeChar(int *c*)	Writes a **char** to the invoking stream.
void writeChars(String *str*)	Writes the characters in *str* to the invoking stream.
void writeDouble(double *d*)	Writes a **double** to the invoking stream.
void writeFloat(float *f*)	Writes a **float** to the invoking stream.
void writeInt(int *i*)	Writes an **int** to the invoking stream.
void writeLong(long *l*)	Writes a **long** to the invoking stream.
final void writeObject(Object *obj*)	Writes *obj* to the invoking stream.
void writeShort(int *i*)	Writes a **short** 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
int available( )	Returns the number of bytes that are now available in the input buffer.
void close( )	Closes the invoking stream. Further read attempts will generate an **IOException**.
int read( )	Returns an integer representation of the next available byte of input. –1 is returned when the end of the file is encountered.
int read(byte *buffer*[ ])	Attempts to read up to *buffer.length* bytes into *buffer*, returning the number of bytes that were successfully read. –1 is returned when the end of the file is encountered.
int read(byte *buffer*[ ], int *offset*, int *numBytes*)	Attempts to read up to *numBytes* bytes into *buffer* starting at *buffer*[*offset*], returning the number of bytes that were successfully read. –1 is returned when the end of the file is encountered.
Object readObject( )	Reads an object from the invoking stream.
long skip(long *numBytes*)	Ignores (that is, skips) *numBytes* bytes in the invoking stream, returning the number of bytes actually ignored.

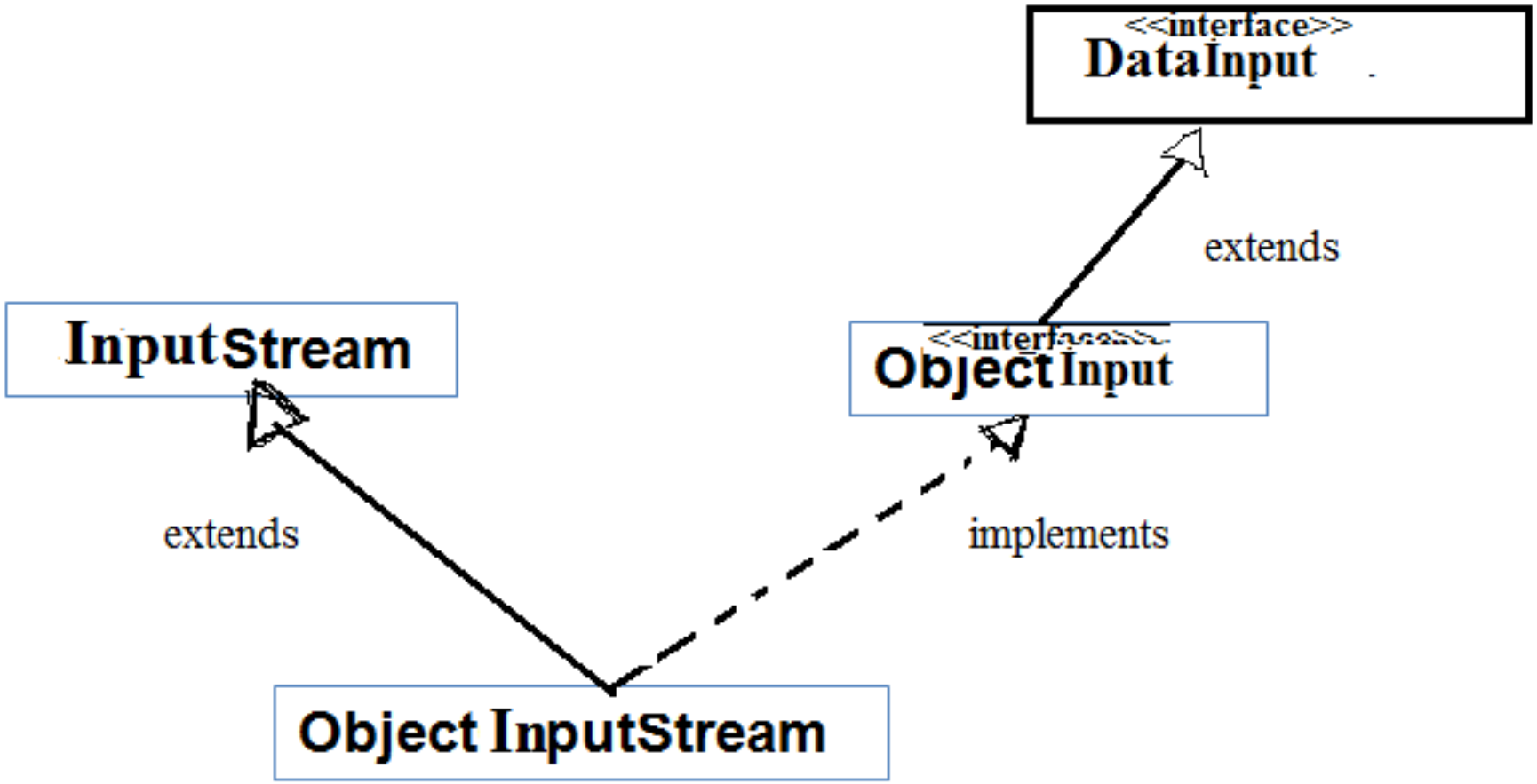
ObjectInputStream

- 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


<<interface>>
DataInput
extends
InputStream
<<interface>>
ObjectInput
extends
implements
Object InputStream

ObjectInputStream-methods

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

Reference

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