CS6308- Java Programming

- V P Jayachitra
- Assistant Professor
- Department of Computer Technology
- MIT Campus
- Anna University

Module V

MODULE V I/O STREAMS	L	T	P	EL
	3	0	4	3
I/O Streams, binary I/O				
SUGGESTED ACTIVITIES : • Practical - binary streams, file streams • EL – Lambdas and Streams				
SUGGESTED EVALUATION METHODS: • Assignment problems • Quizzes				

Object Serialization

Outline

- What is Object Serialization?
- Why Object Serialization in Java?
- Package and class to support Serialization
- Package and class to support DeSerialization
- Serializable interface
- Example code

What is Object Serialization?

- Object serialization is the process of converting Object in to Byte Stream.
  - All primitive type fields
  - To exclude a field from serialization
    - static field, ***transient field***

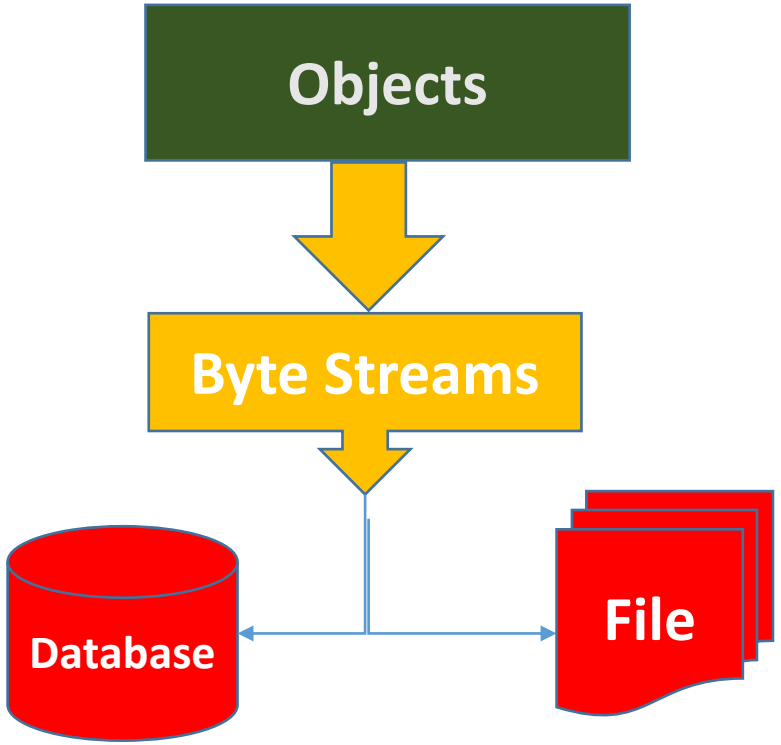
OBJECT---into-->BYTE STREAM!

- Object Deserialization is the process of reconstructing the Object from the Stream.

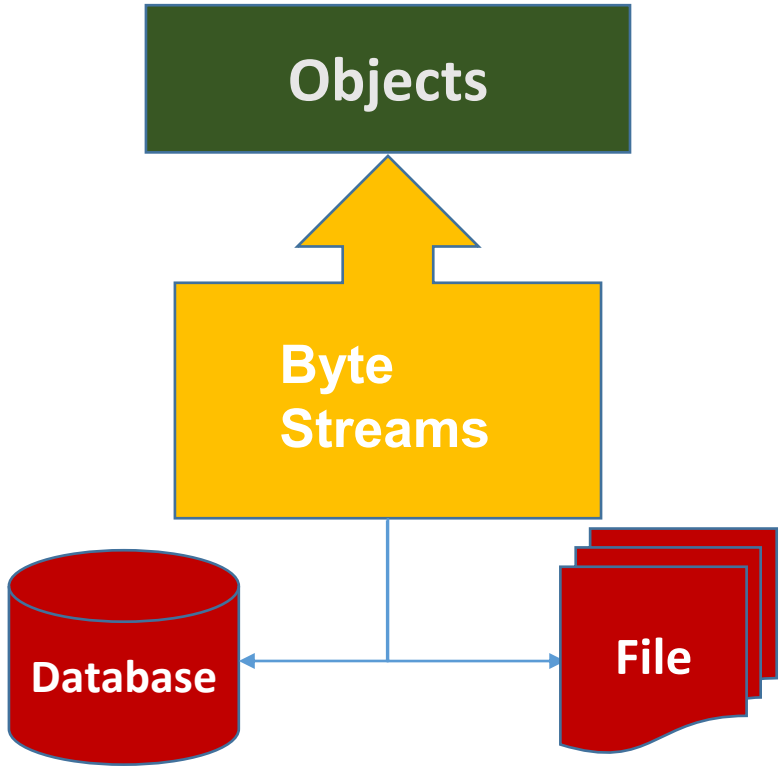
BYTE STREAM-----reconstructing---------->OBJECT

Serialization vs Deserialization

Serialization



Deserialization



Why Object Serialization in Java?

- To support persistence(permanent storage) of Java objects.
- To transmit objects over network.
- To create cloned copy/deep copy without clone class.

Package and class - Serialization

- **Class ObjectOutputStream**
  - java.lang.Object
    java.io. OutputStream
    java.io.ObjectOutputStream

Object → Byte Stream

  - public class **ObjectOutputStream** extends OutputStream
    - Only objects that implements Serializable interface can be written to streams.
  - **Constructor**
    - public ObjectOutputStream(OutputStream out) throws IOException
    - Creates an ObjectOutputStream that writes to the specified OutputStream.
  - Parameters
    - out - output stream to write in to…

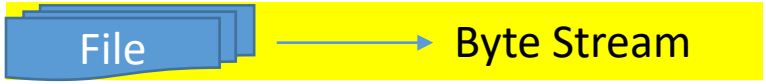
- **Class FileOutputStream**

Byte Stream → File

Package and class - DeSerialization

- **Class ObjectInputStream**
  - **Reconstruct Objects from Byte Stream**
  - java.lang.Object
    java.io. InputStream
    java.io.ObjectInputStream
  - public class **ObjectInputStream** extends InputStream
    - Only objects that implements Serializable interface can be read from streams.
- **Constructor**
  - public ObjectInputStream(InputStream int) throws IOException
    - Creates an ObjectInputStream that Read Object from the specified InputStream.
- Parameters
  - in - input stream to reade in to…
- **Class FileInputStream**
  - Reads Byte Stream from a file

Byte Stream → Object



Serialization and DeSerialization methods

- Special Method required for Serialization
  - private void writeObject(ObjectOutputStream out) throws IOException
    - To write the state of the specified Object (individual fields) to the ObjectOutputStream.

Writes object to the ObjectOutputStream

- Special Method required for DeSerialization
  - private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException
    - To read the state of the Object (individual fields) from the Stream.

ObjectInputStream

Read Object from Input stream

Serializable interface

public interface Serializable

- No methods and no fields

- To enable Serializability of a class
  - class must implement the java.io.Serializable interface.
- Otherwise Classes state cannot be serialized or deserialized.
  - ***<NotSerializableException>*** *i*s thrown

Writing to an Object Stream

1. FileOutputStream f = new FileOutputStream("File.txt");
Purpose: This line creates a FileOutputStream object f which is used to write raw data (in this case, serialized object data) to the file "File.txt".
What it does: The "File.txt" file is opened, and if it doesn't exist, it is created. The FileOutputStream provides a stream for output to this file.

2.ObjectOutputStream s = new ObjectOutputStream(f);
Purpose: This line creates an ObjectOutputStream named s. The ObjectOutputStream is used to write Java objects to the stream connected to the file.
What it does: It wraps the FileOutputStream f, allowing you to write objects (like your Student object) to the file using serialization.

Serialize a Student object to a file

Student object---------->byte stream---------->File

```
FileOutputStream f = new FileOutputStream("File.txt");
ObjectOutputStream s = new ObjectOutputStream(f);
s.writeObject("Student");
```

s.writeObject("Student");
Purpose: This line writes an object to the file. However, the string "Student" is being passed as the object to serialize, which is incorrect if the intention is to serialize an actual Student object.
What it does: In this case, it writes the string "Student" to the file, not a Student object. Since "Student" is a String object, which is serializable by default, the string will be written to the file.

```
import java.io.Serializable;

class Student implements Serializable {
    String name;
    int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

```
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            // Create a Student object
            Student student = new Student("John", 20);

            // Create a FileOutputStream for writing to a file
            FileOutputStream f = new FileOutputStream("File.txt");

            // Create an ObjectOutputStream for writing objects
            ObjectOutputStream s = new ObjectOutputStream(f);

            // Write the Student object to the file
            s.writeObject(student);

// Close the streams
            s.close();
            f.close();

            System.out.println("Student object serialized successfully!");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

What's Missing/Incorrect:
The object "Student" in the s.writeObject() call is just a string, not a Student object. If you want to serialize an actual Student object, you need to pass a Student instance.

Reading from an Object Stream

1. FileInputStream in = new FileInputStream("File.txt");
Purpose: This line opens the file "File.txt" that contains the serialized object data.
What it does: It creates a FileInputStream to read raw bytes from the file. The stream reads the file where the object was previously serialized and stored.

2. ObjectInputStream s = new ObjectInputStream(in);
Purpose: This line wraps the FileInputStream with an ObjectInputStream.
What it does: The ObjectInputStream converts the raw byte data (read from the file) back into actual Java objects.

Deserialize a Student object from a file.

- FileInputStream in = new FileInputStream(“File.txt");
- ObjectInputStream s = new ObjectInputStream(in);
- Student s = (Student)s.readObject();

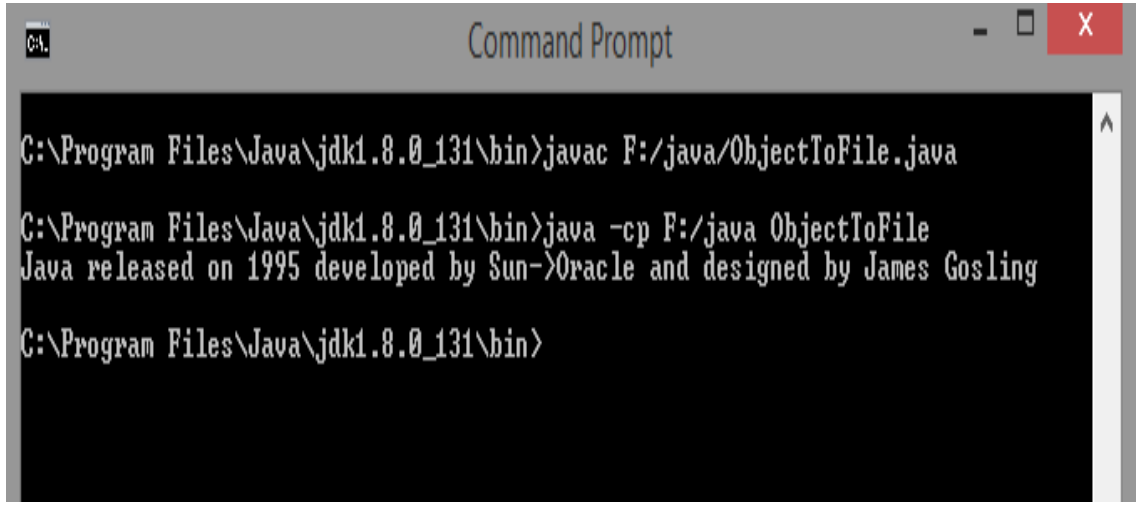
3.Student s = (Student) s.readObject();
Purpose: This line reads the serialized object from the file and casts it back into a Student object.
What it does: The readObject() method of ObjectInputStream reads the next object from the input stream (in this case, the serialized Student object), and the cast to (Student) converts it back into a Student type.

```
import java.io.Serializable;

class Student implements Serializable {
    String name;
    int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{name='" + name + "',
age=" + age + "}";
    }
}
```

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try {
            // Open the file input stream
            FileInputStream in = new FileInputStream("File.txt");

            // Create the ObjectInputStream to read objects from the file
            ObjectInputStream s = new ObjectInputStream(in);

            // Read the Student object from the file and cast it
            Student student = (Student) s.readObject();

            // Close the streams
            s.close();
            in.close();

            // Output the deserialized Student object
            System.out.println("Deserialized Student: " + student);

        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Implementation of serialisation and deseralisation in JAVA!

```
import java.io.*;

class ProgramLanguage implements Serializable{

        public String name;

        public int release;

        public String developer;

        public String designedBy;

}
```




```
class ObjectToFile{
public static void main(String args[]) throws Exception{
        ProgramLanguage PL;
        PL=new ProgramLanguage();
        PL.name="Java";
        PL.release=1995;
        PL.developer="Sun->Oracle";
        PL.designedBy="James Gosling";
        //Serialization

        String Filename="F:/java/File.txt";
        FileOutputStream fo=new   FileOutputStream(Filename);
        ObjectOutputStream os=new ObjectOutputStream(fo);
        os.writeObject(PL);
        //DeSerialization

        FileInputStream fi=new FileInputStream(Filename);
        ObjectInputStream is=new ObjectInputStream(fi);
        ProgramLanguage PL1=(ProgramLanguage)is.readObject();

        System.out.println(PL1.name + " released on " + PL1.release +
        " developed by "  + PL1.developer + " and designed by
        " +PL1.designedBy);

}
```

```java
import java.io.*;

class ProgramLanguage
implements Serializable{
        public String name;
        public int release;
        public String developer;
        public String designedBy;
}
```

```java
class SerialDeserial{
public static void main(String args[]) throws Exception{
        ProgramLanguage[] PL=new ProgramLanguage[2];
        PL[0]=new ProgramLanguage();
        PL[0].name="Java";
        PL[0].release=1995;
        PL[0].developer="Sun->Oracle";
        PL[0].designedBy="James Gosling";

        PL[1]=new ProgramLanguage();
        PL[1].name="GO";
        PL[1].release=2009;
        PL[1].developer="Google";
        PL[1].designedBy="Robert Griesemer";
ObjectOutputStream os=new ObjectOutputStream(new FileOutputStream("F:/java/object.txt“ )
        os.writeObject(PL);
ObjectInputStream is=new ObjectInputStream(new FileInputStream("F:/java/object.txt"));
        ProgramLanguage[] PL1=(ProgramLanguage[])is.readObject();

        System.out.println(PL1[0].name + " released on " + PL1[0].release + " developed
        by "  + PL1[0].developer + " and designed by " +PL1[0].designedBy);
        System.out.println(PL1[1].name + " released on " + PL1[1].release + " developed
        by "  + PL1[1].developer + " and designed by " +PL1[1].designedBy);
        }
```

```
C:\Program Files\Java\jdk1.8.0_131\bin>javac F:/java/SerialDeserial.java

C:\Program Files\Java\jdk1.8.0_131\bin>java -cp F:/java SerialDeserial
Java released on 1995 developed by Sun->Oracle and designed by James Gos
GO released on 2009 developed by Google and designed by Robert Griesemer
```

Other example code

```
//to read N objects from user
ObjectOutputStream os=new ObjectOutputStream(new FileOutputStream("F:/java/objects.txt“ ));
Scanner input=new Scanner(System.in);
ProgramLanguage PL;
while(input.hasNext())
{
name=input.next();
release=input.nextInt();
developer=input.nextLine();
designedBy=input.nextLine();
PL=new ProgramLanguage(name, release,developer, designedBy);
os.writeObject(PL);
}
```