JAVA - FILES

JAVA FILEWRITER CLASS

- Java FileWriter class is used to write character-oriented data to a file.
- It is character-oriented class which is used for file handling in java.

Constructor	Description
FileWriter(String file)	Creates a new file. It gets file name in string.
FileWriter(File file)	Creates a new file. It gets file name in File object.

METHODS OF FILEWRITER CLASS

Method	Description
void write(String text)	It is used to write the string into FileWriter.
void write(char c)	It is used to write the char into FileWriter.
void write(char[] c)	It is used to write char array into FileWriter.
void flush()	It is used to flushes the data of FileWriter.
void close()	It is used to close the FileWriter.

JAVA FILEWRITER EXAMPLE

```
import java.io.FileWriter;
public class FileWriterExample {
    public static void main(String args[]){
        try{
          FileWriter fw=new FileWriter("D:\\testout.txt");
          fw.write("Welcome to javaTpoint.");
          fw.close();
        }catch(Exception e){System.out.println(e);}
        System.out.println("Success...");
    }
}
```

JAVA FILEREADER CLASS

- Java FileReader class is used to read data from the file. It returns data in byte format like FileInputStreamclass.

Constructor	Description
FileReader(String file)	It gets filename in string. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.
FileReader(File file)	It gets filename in file instance. It opens the given file in read mode. If file doesn't exist, it throws FileNotFoundException.

Method	Description
int read()	It is used to return a character in ASCII form. It returns -1 at the end of file.
void close()	It is used to close the FileReader class.

```
import java.io.FileReader;
public class FileReaderExample {
    public static void main(String args[])throws Exception{
        FileReader fr=new FileReader("D:\\testout.txt");
        int i;
        while((i=fr.read())!=-1)
        System.out.print((char)i);
        fr.close();
    }
}
```

SERIALIZATION AND DESERIALIZATION IN JAVA

- **Serialization in Java** is a mechanism of *writing the state of an object into a byte-stream*.
- The reverse operation of serialization is called *deserialization* where byte-stream is converted into an object.
- For serializing the object, we call the **writeObject()** method of *ObjectOutputStream* class, and for deserialization we call the **readObject()** method of *ObjectInputStream* class.
- ***We must have to implement the Serializable interface for serializing the object.***
- **Serializable** is a marker interface (has no data member and method).

STUDENT.JAVA

```
import java.io.Serializable;
public class Student implements Serializable{
 int id;
 String name;
 public Student(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

OBJECTOUTPUTSTREAM CLASS

- *The ObjectOutputStream class is used to write primitive data types, and Java objects to an OutputStream. Only objects that support the java.io.Serializable interface can be written to streams.*

public ObjectOutputStream(OutputStream out) throws IOException {}	It creates an ObjectOutputStream that writes to the specified OutputStream.

Method	Description
1) public final void writeObject(Object obj) throws IOException {}	It writes the specified object to the ObjectOutputStream.
2) public void flush() throws IOException {}	It flushes the current output stream.
3) public void close() throws IOException {}	It closes the current output stream.

OBJECTINPUTSTREAM CLASS

- An ObjectInputStream deserializes objects and primitive data written using an ObjectOutputStream.

1) public ObjectInputStream(InputStream in) throws IOException {}	It creates an ObjectInputStream that reads from the specified InputStream.

Method	Description
1) public final Object readObject() throws IOException, ClassNotFoundException{}	It reads an object from the input stream.
2) public void close() throws IOException {}	It closes ObjectInputStream.

EXAMPLE OF JAVA SERIALIZATION

```
import java.io.*;
class Persist{
 public static void main(String args[]){
  try{
  //Creating the object
  Student s1 =new Student(211,"ravi");
  //Creating stream and writing the object
  FileOutputStream fout=new FileOutputStream("f.txt");
  ObjectOutputStream out=new ObjectOutputStream(fout);
  out.writeObject(s1);
  out.flush();
  //closing the stream
  out.close();
  System.out.println("success");
  }catch(Exception e){System.out.println(e);}
 }
}
```

EXAMPLE OF JAVA DESERIALIZATION

```
import java.io.*;
class Depersist{
 public static void main(String args[]){
  try{
  //Creating stream to read the object
  ObjectInputStream in=new ObjectInputStream(new FileInputStream("f.txt"));
  Student s=(Student)in.readObject();
  //printing the data of the serialized object
  System.out.println(s.id+" "+s.name);
  //closing the stream
  in.close();
  }catch(Exception e){System.out.println(e);}
 }
}
```