OOPJ Notes Day-12 Session-2 Date: 10/05/2023

Java I/O (File Handling in Java)

- variable
  - t is temporary conrainter which is used to store record in primary memory.
- File
  - It is permenant conrainter which is used to store record on secondary memory.
  - Types of Files
    - Text Files
      - Examples: .c, .cpp, .java. .cs, .html, css. .js, .txt, .doc, .docs, .xml, .json etc.
      - We can read text file using any text editor.
      - It requires more processing than binary file hence it is slower in performance.
      - If we want to save data in human readable format then we should create text file.
    - Binary Files
      - Examples: .jpg, .jpeg, .bmp, .gif, .mp3, .mp4, .obj, .class etc.
      - To read binary file, we must use specific program.
      - It requires less processing than text file hence it is faster in performance.
      - If we dont want to save data in human readable format then we should create binary file.
- Stream
  - It is an sequence of bits which either readed or written from source to destination.
  - Stream is always associated with resource.
  - Standard stream instances of Java programming languages which are associated with
    - Console(Keyboard / Monitor):
      - System.in
      - System.out
      - System.err
  - If we want to save data in file the we should use types declared in java.io package.
  - java.io.File class reprsents Physical file on secondary memory.
- Exploration of java.io.File class
  - Creating a File/Directory using java.io.File

```
class FileClassDemo {

    public static void main(String[] args) {

        String path="abc.txt";

        File f=new File(path);

        try {
            if(f.exists())
            {
                System.out.println("File already exists");
            }
            else
            {
                f.createNewFile();
                System.out.println("File Created Successfully");
            }

        } catch (IOException e) {
            System.out.println("File can be created");
        }

    }

}
```

```
- Removing a File/Directory using java.io.File
- Reading MetaData of File/Directory using java.io.file
```

- Reading and Writing data using FileInputStream and FileOutputStream
- Reading and Writing data using BufferedInputStream and BufferedOutputStream
- Reading and Writing data using DataInputStream and DataOutputStream
- Serialization and Dersialization using ObjectInputStream and ObjectOutputStream

```
import java.io.File;
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.io.Serializable;

class St implements Serializable
{
    public int RollNo;
    public String Name;
    public St(int r, String n)
    {
        RollNo=r;
        Name=n;
    }

}
class OOSDemo
{
    public static void main(String[] args) {
        St s1=new St(1001,"Malkeet");

        String path="abc.txt";

        FileOutputStream fout=null;
        FileInputStream fin=null;
        ObjectOutputStream oout=null;
        ObjectInputStream oin=null;
        try {
            fout=new FileOutputStream(path);
            fin=new FileInputStream(path);
        } catch (FileNotFoundException e) {
            System.out.println("Can't Read and Write to file");
        }

        File f=new File(path);
        try {
            oout=new ObjectOutputStream(fout);
              oout.writeObject(s1);
        } catch (IOException e) {
            System.out.println("Cant serilalize");
        }

        try {
            oin=new ObjectInputStream(fin);

            try {

                St obj;
                obj=(St)oin.readObject();

                System.out.println("Roll No: "+obj.RollNo+" Name: "+obj.Name);

            } catch (ClassNotFoundException e) {
                System.out.println("Can't Deserliaze");
            }



        } catch (IOException e) {
            System.out.println("Can Read File");
        }



    }
}
```

- Serializable interface
- Restricting serialization using transient modifier
- Text file manipulation using Reader and Writer

try with resource

- External resources like File, Database con and N/W con to be in try block before its use.

```
class FileClassDemo {


    public static void main(String[] args) {
        //String pd="C:Users\\hp\\Desktop\\Dir\\File\\";
        String path="abc.txt";


        File f=new File(path);


        try{ //Try with resources
            f.createNewFile();
            System.out.println("File Created");
        }
        catch(IOException ex)
        {
            System.out.println("File not created"+ex.getMessage());
        }


    }
}
```