Dt : 29/11/2022

IO Streams and Files(Conclusion):

Classes Related to Character Stream:

1.FileWriter

2.FileReader

3.BufferedReader

1.FileWriter:

=>FileWriter Class is from java.io package and which is used to create new file and opens the file to write character Stream.

syntax:

```
FileWriter fw = new FileWriter("fPath/fName");
```

2.FileReader:

=>FileReader class is from java.io package and which is used to find the file and opens the file to read the character stream.

syntax:

```
FileReader fr = new FileReader("fPath/fName");
```

3.BufferedReader:

=>BufferedReader classes is from java.io package and which is used to read character Stream into JavaProgram.

syntax:

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

Dt : 30/11/2022

Ex:(demonstrating reading data from console)

DemoFile2.java

```
package maccess;
import java.io.*;
public class DemoFile2 {
    public static void main(String[] args) {
        try {
         InputStreamReader is = new
InputStreamReader(System.in);
         BufferedReader br = new BufferedReader(is);
         String path = "D:\\Images\\Text.txt";//File
location
         FileWriter fw = new FileWriter(path);//Creates
the file
         System.out.println("Enter the data:(@ at
end)");
         char ch1;
         while((ch1=(char)br.read())!='@')
         {
             fw.write(ch1);
         }//end of loop
         System.out.println("Data stored
Successfully...");
         fw.close();
         System.out.println("=====Display data from
File=====");
```

```
            FileReader fr = new FileReader(path);//Opens
the file
            int k;
            while((k=fr.read())!=-1)
            {
                 System.out.print((char)k);
            }//end of loop
            fr.close();
           }catch(Exception e) {e.printStackTrace();}
      }

}
```

o/p:

Enter the data:(@ at end)

java is simple

thread

task

progra

job

@

Data stored Successfully...

=====Display data from File=====

java is simple

thread

task

progra

job

==========

faq:

define "File"?

=>"File" is a class from java.io package and which is used to find the properties of file like filePath,fileName,fileLength,...

syntax:

File ob = new File("fPath&fName");

==========

Note:

=>length() method is used ti find the length of String and which is also used to find the length of file.

==========

**imp*

Socket programming in Java(Network programming in java):

define Computer N/W?

=>The inter connection of autonomous computers is known as Computer N/W.

=>Based on number of nodes in the N/W,the N/Ws are categorized in to the following:

(1)LAN - Local Area N/W

(2)MAN - Metropolitan Area N/W

(3)WAN - Wide Area N/W

(4)WWW - World Wide Web

define WWW?

=>WWW is a UnLimited N/W holding UnLimited Nodes.

=>The Computers in the N/w are categorized into two types:

(1)Server Computers

(2)Client Computers

(1)Server Computers:

=>The computers which are holding Server Applications are known as Server Computers.

=>These Server Computers will accept the request and generate response.

(2)Client Computers:

=>The computers which are holding client applications are known as Client Computers

=>These Client Computers will generate request to Servers.

----

define N/W protocol?:

=>The set-of-rules used by computers in the N/W is known as N/W protocol.

(1)Connection oriented protocols

(2)Connection less Protocols

(1)Connection oriented protocols:

=>In Connection Oriented Protocols the Client will receive ack from Server.

Ex:

TCP/IP

(2)Connection less Protocols:

=>In Connection less protocols the client will not receive ack from Server.

Ex:

UDP

---

define IP Address?

=>The Unique identification number used by computer in the N/W.

=>we use this IP Address to identify the computer in the N/W.

=>Based on the range of IP Addresses the N/Ws are Classified into the following:

class A - 1.0.0.0 to 126.255.255.254 (16 million)

class B - 128.1.0.1 to 191.255.255.254 (65000)

class C - 192.0.1.1 to 223.255.254.254 (254)

class D - 224.0.0.0 to 239.255.255.255(multicast)

class E - 240.0.0.0 to 254.255.255.255(future)

note:

127.0.0.0 loopback network

255.255.255.255 - default network

**imp*

define Socket?

=>The logical connection established for communication is known as Socket.

=>we use port number for Socket Connection.

Ex:

portNo : 0 to 65535

The following are the reserved port numbers:

13 - date and time services

21 - FTP which transfers files

23 - Telnet,which provides remote login

25 - SMTP,which delivers mails

80 - HTTP,which transfers web pages

109 - POP,which access mail boxes

The following are the network classes from "java.net" package:

(1)Socket,ServerSocket - used for TCP/IP connection

(2)DatagramPacket,DatagramSocket - used for UDP connection

(3)URL,URIConnection - used for read-write data from the internet

(4)InetAddress - this class is used to get the IP Address and hostname of the computer.

Note:

The communication b/w two Java Appls running on two diff JVMs can be established using 'Socket' and 'ServerSocket' classes.

=>The JVMs can be same ComputerSystem or different ComputerSystems.

(1)Socket,ServerSocket Classes:

methods of Socket class:

1. InputStream getInputStream()

2. OutputStream getOutputStream()

3. synchronized void close()

methods of ServerSocket class:

1. Socket accept()

2. synchronized void close()

Dt : 1/12/2022

Variables - Methods

Method - Methods

Classes - Classes

packages - packages

JavaProgram - File Storage

JavaProgram - N/w

Server.java

Program:

Server.java

```
import java.io.*;
import java.net.*;
class Server
{
```

```
public static void main(String args[])
                throws IOException
{
ServerSocket ss=new ServerSocket(888);
Socket s=ss.accept();
System.out.println("connection established");
PrintStream ps=new PrintStream
      (s.getOutputStream());
DataInputStream br=new DataInputStream
            (s.getInputStream());
DataInputStream kb=
new DataInputStream(System.in);
while(true)
{
String str,str1;
while((str=br.readLine())!=null)
{
System.out.println(str);
str1=kb.readLine();
ps.println(str1);
}
ps.close();
```

```
br.close();
kb.close();
ss.close();
s.close();
System.exit(0);
}
}
}
```

Client.java

```
import java.io.*;
import java.net.*;
class Client
{
public static void main(String args[])
        throws IOException
{
Socket s=new Socket("localhost",888);
DataOutputStream dos=new DataOutputStream
        (s.getOutputStream());
DataInputStream br=new DataInputStream
                (s.getInputStream());
```

```
DataInputStream kb=new DataInputStream
                (System.in);
String str,str1;
while(!(str=kb.readLine()).equals("exit"))
{
dos.writeBytes(str+"\n");
str1=br.readLine();
System.out.println(str1);
}
dos.close();
br.close();
kb.close();
s.close();
}
}
```

==========

Note:

=>Execute above two programs in two differnt CommandPrompts.

==========

Summary:

1.Socket Programming

2.RPC/RMI

3.CORBA

4.WebServices

==========

1.Programming Components(Java Alphabets)

(a)Variables

1.Primitive DataType variables(Values)

(i)Static

(ii)NonStatic

=>Instance

=>Local

2.NonPrimitive DataType variables(Object references)

(i)Static

(ii)NonStatic

=>Instance

=>Local

(b)Methods

1.Static methods

(i)pre-defined methods

(ii)User defined methods

2.Non-Static methods(Instance methods)

(i)pre-defined methods

(ii)User defined methods

(c)Blocks

1.Static blocks

2.NonStatic blocks(Instance blocks)

(d)Constructors

=>NonStatic Constructors

(e)Classes

1.static classes(Only InnerClasses)

2.NonStatic classes

(f)Interfaces

1.static Interfaces(Only InnerInterfaces)

2.NonStatic Interfaces

(g)AbstractClasses

1.static abstract classes(Only InnerAbstractClasses)

2.NonStatic abstract classes

==========

2.Programming Concepts

(a)Object Oriented Programming

=>Constructing Applications using Class-Object Concept

=>Object definition

=>Object Creation

=>Object Location

=>Object Components

=>Object Types

(i)User Defined Class Objects

(ii)String Objects

(iii)WrapperClass Objects

(iV)Array Objects

(v)Collection<E> Objects

(vi)Map<K,V> Objects

(vii)Enum<E> Objects

=>Object Serialization

=>Object Collection

=>Object Locking

=>Object Cloning

=>Object Sorting

(b)Exception Handling process

=>Error Vs Exception

=>Exception Handling process

=>try Vs catch Vs finally

=>throw Vs throws

=>Exception re-throwing process

=>Checked Exceptions Vs NonChecked Exceptions

(c)Multi-Threading process(Level-1)

=>Thread Definition

=>Thread Creation

=>Thread Location

=>Thread Behaviour

=>Thread synchronization

(1)Mutual Exclusion process

(i)synchronized block

(ii)synchronized method

(iii)static synchronization

(2)Thread Commmunication process

=>wait() Vs sleep()

=>notify()

=>notifyAll()

=>Thread-Life-Cycle

(d)Java Collection Framework(JCF)

(Data Structures in Java)

=>Array

=>Set<E>

=>List<E>

=>Queue<E>

=>Map<k,V>

=>Enum<E>

(e)IOStreams and Files in Java

=>Stream

=>Types of Streams

=>Byte Stream Vs Character Stream

=>FileInputStream Vs FileOutputStream

=>ObjectInputStream Vs ObjectOutputStream

=>Serialization Vs DeSerialization

=>FileReader Vs FileWriter

=>File->JavaProgram->File

=>Console->JavaProgram->File

=>File->JavaProgram->Console

(f)Networking in Java

(Communication with TCP/IP)

=>Network Definition

=>Server Vs Client

=>IP Address

=>Socket

=>PortNo

=>ServerSocket Vs Socket

==========

3.Object Oriented Programming features

(a)Class

=>Complete Structure of class is Constructed

(b)Object

=>Object is a storage related to class holding Instance members of Class

(c)Abstraction

=>The process of hiding the background implementations which are not needed by the end-user is known as Abstraction process.

=>we use Interfaces and Abstract classes to construct Abstraction process.

(d)Encapsulation

=>The process of binding all the programming components into a Single unit class is known as Encapsulation process.

(e)PolyMorphism

=>Definition

=>Dynamic PolyMorphism Vs Static PolyMorphism

=>static Vs private Vs final

=>SingleTon Classes

=>SingleTon class design pattern

=>Mutable Objects Vs Immutable Objects

(f)Inheritance

=>Extraction features from one component to another Component is known as Inheritance

=>Types:

**Single Inheritance*

**Multiple Inheritance(Using Interfaces)*

**Multi-Level Inheritance*

**Hierarchal Inheritance*

**Hybrid Inheritance*

=>According Realtime(Types)

**Single Inheritance*

**Multiple Inheritance(Using Interfaces)*

==========

define StandAlone Application?

=>The application which is installed in one computer and perform actions in the same computer are knoen as StandAlone Application.

=>Based on User interaction the StandAlone applications are categorized into two types:

(a)CUI Applications

(b)GUI Applications

(a)CUI Applications:

=>The applications in which the user interacts through Console are known as CUI Applications.(CUI - Console User Interface)

(b)GUI Applications:

=>The applications in which the user interacts through GUI Components are known as GUI Applications.(GUI - Graphical User Interface)

=>To design GUI components we use the following:

(i)AWT

(ii)Swing

(iii)JavaFx

(i)AWT:

=>AWT stands for "Abstract Window Toolkit" and which is used to design GUI components.

Dis-Advantage:

AWT will not Support MVC(Model View Controller).

(ii)Swing:

=>Swing also used to develop GUI Components and which support MVC.

(iii)JavaFx:(Java8)

=>JavaFx introduced by Java8 version and which ai also used to design GUI components

Advantage:

InBuilt rich UI controls

==========

Studend2.java (StandAlone Application)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
public class Student2
extends JFrame implements ActionListener
{
        String str1,str2=null,str3,str4;
        JLabel lb1;
        JLabel lb2;
        JLabel lb3;
        JLabel lb4;
        JLabel lb5;
```

```
JLabel lb6;

JLabel lb7;

JLabel lb8;

JComboBox jc;

JTextField t1;

JTextField t2;

JTextField t3;

JTextField t4;

JTextField t5;

JTextField t6;

JButton b1;

JButton b2;


Student2() //constructor

{

        Container c=this.getContentPane();

        String str1[]=

        {"ECE","CSE","EEE","MECH","CIVIL"};

        jc= new JComboBox(str1);

        c.setLayout(null);

        c.setBackground(Color.yellow);

Font f1=new Font("dialog",Font.BOLD,30);
```

```
        lb1=new JLabel("Student Data");
        lb1.setFont(f1);
        lb1.setBounds(450,50,500,50);
        lb1.setForeground(Color.red);
Font f=new Font("dialog",Font.BOLD,20);
lb3= new JLabel("BRANCH");
        lb3.setFont(f);
        lb3.setBounds(450,100,500,50);
        lb3.setForeground(Color.red);
              jc.setFont(f);
              jc.setBounds(550,100,150,50);
              jc.setForeground(Color.GREEN);
        lb2=new JLabel("NAME");
        lb2.setFont(f);
        lb2.setBounds(50,100,500,50);
        lb2.setForeground(Color.red);
                    t1=new JTextField(50);
              t1.setBounds(200,100,200,50);
              lb4=new JLabel("RNO");
        lb4.setFont(f);
        lb4.setBounds(50,180,500,50);
        lb4.setForeground(Color.red);
```

```
        t2=new JTextField(50);
        t2.setBounds(200,180,200,50);
lb5=new JLabel("6 SUB MARKS");
lb5.setFont(f);
lb5.setBounds(50,260,500,50);
lb5.setForeground(Color.red);
        t3=new JTextField(50);
        t3.setBounds(200,260,300,50);
lb6=new JLabel("TOTAL");
lb6.setFont(f);
lb6.setBounds(50,340,500,50);
lb6.setForeground(Color.red);
        t4=new JTextField(50);
        t4.setBounds(200,340,150,50);
lb7=new JLabel("PERCENTAGE");
lb7.setFont(f);
lb7.setBounds(450,340,500,50);
lb7.setForeground(Color.red);
        t5=new JTextField(50);
        t5.setBounds(600,340,150,50);
        lb8=new JLabel("RESULT");
lb8.setFont(f);
```

```
lb8.setBounds(50,420,500,50);
lb8.setForeground(Color.red);
        t6=new JTextField(50);
        t6.setBounds(200,420,150,50);
        b1=new JButton("Calculate");
b1.setBounds(300,500,100,50);
        b2=new JButton("Clear");
b2.setBounds(500,500,100,50);
        c.add(lb1);
c.add(lb2);
c.add(t1);
c.add(lb3);
c.add(jc);
c.add(lb4);
c.add(t2);
c.add(lb5);
c.add(t3);
c.add(lb6);
c.add(t4);
c.add(lb7);
c.add(t5);
c.add(lb8);
```

```
            c.add(t6);
            c.add(b1);
            c.add(b2);
            b1.addActionListener(this);
            b2.addActionListener(this);
        }
        public static void main(String[] args)
        {
            Student2 obj1=new Student2();
            obj1.setTitle("Student Details");
            obj1.setSize(800,600);
            obj1.setVisible(true);
obj1.setDefaultCloseOperation
   (JFrame.EXIT_ON_CLOSE); // close window
        }
public void actionPerformed(ActionEvent arg)
        {
            str1=arg.getActionCommand();
            if(str1.equals("Calculate"))
            {
                str2=t1.getText();
                str3=t2.getText();
```

```
try
{
int len=str3.length();
if(len==10)
        {
        try
                {
String s11=str3.substring(7,8);
        Choice2 c1=new Choice2();
        String bb=c1.valid(s11);
        boolean br1=bb.equals("1");
        boolean br2=bb.equals("2");
        boolean br3=bb.equals("3");
        boolean br4=bb.equals("4");
        boolean br5=bb.equals("5");
        String ss=null;
        if(br1)
                ss="CIVIL";
        else if(br2)
                ss="EEE";
        else if(br3)
                ss="mech";
```

```
                    else if(br4)
                            ss="ECE";
                    else if(br5)
                            ss="CSE";
        if(((jc.getSelectedItem().toString())
                                .equals(ss)))
                    {
                            try
                            {
                            str4=t3.getText();
StringTokenizer st=
                new StringTokenizer(str4," ");
                            int a,b,c,d,e,f;
                            String s1=st.nextToken();
                            String s2=st.nextToken();
                            String s3=st.nextToken();
                            String s4=st.nextToken();
                            String s5=st.nextToken();
                            String s6=st.nextToken();
                    a=Integer.parseInt(s1);
                    b=Integer.parseInt(s2);
                    c=Integer.parseInt(s3);
```

```
                d=Integer.parseInt(s4);
                e=Integer.parseInt(s5);
                f=Integer.parseInt(s6);
        if(!((a<0 || a>100) || (b<0 || b>100) ||
            (c<0 || c>100)
    || (d<0 || d>100) || (e<0 || e>100) ||
            (f<0 || f>100)))
                        {
                int total=a+b+c+d+e+f;
                t4.setText(" "+total);
                float per=total/6;
                t5.setText(" "+per);
if((a<35 || b<35 || c<35 || d<35 || e<35 ||
                f<35))
                        {
                t6.setText("fail");
                        }
                        else
                        {
                t6.setText("pass");
                        }
                }
```

```
                                        else
                                        {
JOptionPane.showMessageDialog
  (this,"values between 0 to 100");
                                        }
                                    }
                            catch(NumberFormatException nfe)
                                    {
JOptionPane.showMessageDialog
(this,"only enter the number in marks");
                                    }
                                }
                                else
                                {
JOptionPane.showMessageDialog
(this,"mismatch of rno and branch");
                                }
                                    }
                        catch(NullPointerException npe)
                                {
        JOptionPane.showMessageDialog
            (this,"invalid rno");
```

```
                    }
                    }
                else
                    {
        JOptionPane.showMessageDialog
        (this,"rno must be 10 digits");
                    }
                }
        catch(NoSuchElementException nsee)
                {
JOptionPane.showMessageDialog
(this," plz enter 6 sub marks");
                }
        }
            else
            {
                t1.setText("");
                t2.setText("");
                t3.setText("");
                t4.setText("");
                t5.setText("");
                t6.setText("");
```

```
                }
        }
}
class Choice2
{
  String b;
  String valid(String s1)
  {
    switch(s1)
        {
          case "1":
                b="1";
                break;
          case "2":
                b="2";
                break;
          case "3":
                b="3";
                break;
         case "4":
                b="4";
                break;
```

```
        case "5":
                b="5";
                break;
        }
 return b;
  }
}
```