1. Implement a client-server program where the server sends a message and the client receives it.

import java.io.\*;

import java.net.\*;

class Server

{

public static void main(String[] args)

{

try

{

System.out.println("Server Started");

ServerSocket srvr= new ServerSocket(1234);

Socket skt= srvr.accept();

System.out.println("Client-Server connection established\n");

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

PrintWriter pwsocket= new PrintWriter(skt.getOutputStream(),true);

System.out.print("Enter your message: ");

String omsg= brconsole.readLine();

pwsocket.println(omsg); //always use println() as the new line at end clears the buffer, print() does not

pwsocket.flush(); //flushing becomes optional when println() is used as the new line clears the buffer

pwsocket.close();

skt.close();

srvr.close();

}

catch (Exception e)

{

System.out.println("ERROR: "+e.getMessage());

}

}

}

class Client

{

public static void main(String[] args)

{

try

{

System.out.println("Client Started\n");

Socket skt= new Socket("localhost",1234);

BufferedReader brsocket= new BufferedReader(new InputStreamReader(skt.getInputStream()));

//while(!brsocket.ready()){} If multiple connections established, this loop waits until the listener is ready for hearing

String imsg= brsocket.readLine();

System.out.print("Message received: ");

System.out.println(imsg);

brsocket.close();

skt.close();

}

catch (Exception e)

{

System.out.println("ERROR: "+e.getMessage());

}

}

}

1. Implement a client-server program where the client sends a message and the server receives it.

import java.io.\*;

import java.net.\*;

class Server

{

public static void main(String[] args)

{

try

{

System.out.println("Server Started");

ServerSocket srvr= new ServerSocket(1234);

Socket skt= srvr.accept();

System.out.println("Client-Server connection established\n");

BufferedReader brsocket= new BufferedReader(new InputStreamReader(skt.getInputStream()));

//while(!brsocket.ready()){} If multiple connections established, this loop waits until the listener is ready for hearing

String imsg= brsocket.readLine();

System.out.print("Message received: ");

System.out.println(imsg);

brsocket.close();

skt.close();

srvr.close();

}

catch (Exception e)

{

System.out.println("ERROR: "+e.getMessage());

}

}

}

class Client

{

public static void main(String[] args)

{

try

{

System.out.println("Client Started\n");

Socket skt= new Socket("localhost",1234);

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

PrintWriter pwsocket= new PrintWriter(skt.getOutputStream(),true);

System.out.print("Enter your message: ");

String omsg= brconsole.readLine();

pwsocket.println(omsg); //always use println() as the new line at end clears the buffer, print() does not

pwsocket.flush(); //flushing becomes optional when println() is used as the new line clears the buffer, mandatory for print()

pwsocket.close();

skt.close();

}

catch (Exception e)

{

System.out.println("ERROR: "+e.getMessage());

}

}

}

1. Implement FileInputStream and FileOutputStream.

import java.io.\*;

public class Assignment2\_3

{

public static void main(String args[])

{

try

{

byte bWrite[]= {1,3,52,65,97};

OutputStream os= new FileOutputStream("test.txt");

for(int x=0; x<bWrite.length;x++)

{

os.write(bWrite[x]);

}

os.close();

InputStream is= new FileInputStream("test.txt");

int size= is.available();

for(int i=0; i<size; i++)

{

System.out.print((char)is.read()+" ");

}

is.close();

}

catch(IOException e)

{

System.out.println("Unsuccessful");

}

}

}

1. Convert the format of an image.

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

class Assignment2\_4

{

public static void main(String[] args)

{

File file= new File("input.png"); //if in another directory, provide full path in ""

BufferedImage image= null;

try

{

image= ImageIO.read(file);

ImageIO.write(image,"jpg",new File("output.jpg")); //if in another directory, provide full path in ""

ImageIO.write(image,"png",new File("output.png")); //if in another directory, provide full path in ""

ImageIO.write(image,"bmp",new File("output.bmp")); //if in another directory, provide full path in ""

}

catch(IOException e)

{

e.printStackTrace();

}

System.out.println("Done");

}

}