**Midterm test**

**Due: Monday 3/28/2016, 11:59 P.M.**

**No delay is accepted whatsoever. After the due date I grade the test for 0**

**Drop the test as a zipped folder in *Midterm Exam on blackboard.***

**I cannot answer the questions to help you, but if you like to clarify anything about the questions email me at** [mbadii@pace.edu](mailto:mbadii@pace.edu)**. In this case let me know which part of a question is not clear for you.**

**Note: This is an individual test. You must not communicate with each other for this test.**

**Do not delete anything from this file.**

**Each question is for 20 points.**

**Copy/paste your answer to questions 1-5 under the word: *Answer* in this document.**

**Question 1**: Add code to the following class: MyClient to impose a timeout of 5 seconds. If a timeout does occur, a message “timed out on receiver” should be displayed and the client should be terminated.

Note: This question is similar (not the same) as exercise 4 in chapter 4 of your book.

Note: Only copy/paste the class: MyClient under the word: **Answer**.

Hints:

1. You need to call the java method: setSoTimeout(5000) of class: DatagramSocket.
2. You need an extra catch-block with the exception class: SocketTimeoutException.

**Client**:

**import** java.io.\*;

**import** java.net.\*;

**public** **class** MyClient {

**public** **static** **void** main(String[] args) **throws** IOException {

**final** **int** MAX\_LEN = 100;

**byte**[] buffer = **new** **byte**[MAX\_LEN];

**int** port = 16790;

DatagramPacket datagram = **new** DatagramPacket(buffer, buffer.length);

DatagramSocket clientSocket = **new** DatagramSocket(port);

**try** {

clientSocket.receive(datagram);

String message = **new** String(buffer);

System.***out***.println(message);

} **catch** (IOException e) {

System.***out***.println("Error: " + e);

System.*exit*(0);

} **finally** {

clientSocket.close();

}

System.***out***.println("The program terminated with no error and no exception");

}

}

**Server**:

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**public** **class** MyServer {

**public** **static** **void** main(String[] args) **throws** IOException {

**try** {

Scanner keyboard = **new** Scanner(System.***in***);

System.***out***.print("Enter a message to send: ");

String message = keyboard.nextLine();

**byte**[] buffer = message.getBytes();

**int** port = 16790;

InetAddress host = InetAddress.*getByName*("localhost");

DatagramSocket serverSocket = **new** DatagramSocket();

DatagramPacket datagram = **new** DatagramPacket(buffer, buffer.length, host, port);

serverSocket.send(datagram);

serverSocket.close();

} **catch** (IOException e) {

System.***out***.println("Error: " + e);

System.*exit*(0);

}

}

}

**Answer**:

**MyClient.java**

import java.io.\*;

import java.net.\*;

public class MyClient {

public static void main(String[] args) throws IOException {

final int MAX\_LEN = 100;

byte[] buffer = new byte[MAX\_LEN];

int port = 16790;

DatagramPacket datagram = new DatagramPacket(buffer, buffer.length);

DatagramSocket clientSocket = new DatagramSocket(port);

clientSocket.setSoTimeout(5000);

try {

clientSocket.receive(datagram);

String message = new String(buffer);

System.out.println(message);

}

catch (SocketTimeoutException ex) {

System.out.println("timed out on receiver");

System.exit(0);

}

catch (IOException e) {

System.out.println("Error: " + e);

System.exit(0);

}

catch (Exception e) {

System.out.println("Error: " + e);

System.exit(0);

}

finally {

clientSocket.close();

}

System.out.println("The program terminated with no error and no exception");

}

}

**Question 2**: In a client-serve program we normally run the server first and then the client. If we run the client first we get error message. We like to run the client first and then the server. Change the the following class: MyClient to avoid any exception error. In other words as long as the server is not connected the client stays on with no error.

Note: Only copy/paste the class: MyClient under the word: **Answer**.

Hints: The client has to loop around until the server gets connect. You may need to include more try-catch block(s) or/and more specific exception class(s). Run the client many times to make sure you do not get any type of exception errors.

**Server**:

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**public** **class** MyServer {

**public** **static** **void** main(String[] args) **throws** IOException {

ServerSocket serverSocket = **null**;

Socket clientSocket = **null**;

PrintWriter out = **null**;

System.***out***.println("Daytime server ready.");

**try** {

serverSocket = **new** ServerSocket(4321);

**while**(**true**){

clientSocket = serverSocket.accept();

System.***out***.println("Request received.");

out = **new** PrintWriter(clientSocket.getOutputStream(), **true**);

Date timestamp = **new** Date ();

out.println(timestamp.toString());

}

} **catch** (IOException e) {

System.***out***.println("Error: " + e);

System.*exit*(0);

}

out.close();

clientSocket.close();

serverSocket.close();

}

}

**Client**:

**import** java.io.\*;

**import** java.net.ConnectException;

**import** java.net.Socket;

**public** **class** MyClient {

**public** **static** **void** main(String[] args) **throws** IOException {

Socket clientSocket = **null**;

BufferedReader in = **null**;

System.***out***.println("Welcome to the Daytime client.");

**try** {

clientSocket = **new** Socket("localhost", 4321);

in = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));

String s = in.readLine();

System.***out***.println("Here is the timestamp received from the server: "+s);

in.close();

clientSocket.close();

clientSocket.close();

}**catch** (IOException e) {

System.***out***.println("Error: " + e);

System.*exit*(0);

}

System.***out***.println("The program terminated with no error and no exception");

}

}

**Answer**:

import java.io.\*;

import java.net.ConnectException;

import java.net.Socket;

public class MyClient {

public static void main(String[] args) throws IOException {

Socket clientSocket = null;

BufferedReader in = null;

System.out.println("Welcome to the Daytime client.");

while(true)

{

try {

clientSocket = new Socket("localhost", 4321);

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

String s = in.readLine();

System.out.println("Here is the timestamp received from the server: "+s);

in.close();

clientSocket.close();

break;

}catch(ConnectException ct)

{

System.out.println("No Connection is made!Waiting to Connect to server!");

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

catch (IOException e) {

System.out.println("Error: " + e);

System.exit(0);

}

catch(Exception ex)

{

System.out.println("Error: " + ex);

}

}

System.out.println("The program terminated with no error and no exception");

}

}

**Question 3**

Using RMI, write an application for a prototype opinion poll system. Assume that only one issue is being polled. Respondents may choose *yes,* or *no*. Write a client-server application to accept the votes, keep the tally (in memory), and provide the current counts to those who are interested.

Note: Copy/paste the classes: (all java files) under the word: **Answer**.

Note: This question is a shorten version of exercise: 9 on page 230 of your book.

Note: You do not need to make a number of client programs. Make one server and one client programs. Keep the server running and run the client several times. For example I run the client three times as follows:

Enter port: What is your vote?[yes|no] : yes

Yes = 1; No = 0

Enter port: What is your vote?[yes|no] : no

Yes = 1; No = 1

Enter port: What is your vote?[yes|no] : no

Yes = 1; No = 2

**Server Side**:

**import** java.rmi.Remote;

**public** **interface** VoteInterface **extends** Remote{

//This interface is complete. Do not change it.

**public** String castVote(**char** vote )**throws** java.rmi.RemoteException;

}

**import** java.rmi.\*;

**import** java.rmi.server.\*;

**public** **class** VoteServerImpl **extends** UnicastRemoteObject **implements** VoteInterface {

//Complete this method.

}

**import** java.rmi.\*;

**import** java.rmi.server.\*;

**import** java.rmi.registry.Registry;

**import** java.rmi.registry.LocateRegistry;

**import** java.net.\*;

**public** **class** MyServer{

**public** **static** **void** main(String args[]){

//Complete this method.

}

}

**Client Side:**

**import** java.rmi.Remote;

**public** **interface** VoteInterface **extends** Remote{

//This interface is complete. Do not change it.

**public** String castVote(**char** vote )**throws** java.rmi.RemoteException;

}

**import** java.rmi.\*;

**import** java.util.\*;

**public** **class** MyClient{

**public** **static** **void** main(String args[]){

//Complete this method.

}

}

**Answer:**

**Server Side**:

**MyServer.java**

import java.rmi.\*;

import java.rmi.server.\*;

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.net.\*;

public class MyServer{

public static void main(String args[]){

try{

int port = 16790;

String host = "localhost";

VoteServerImpl exportedObj = new VoteServerImpl();

startRegistry(port);

String registryURL = "rmi://" + host + ":" + port + "/poll";

Naming.rebind(registryURL, exportedObj);

System.out.println("Hello Server ready.");

}catch (Exception e){

e.printStackTrace();

}

}

private static void startRegistry(int port)throws RemoteException, NotBoundException{

try{

Registry registry = LocateRegistry.getRegistry(port);

registry.list( ); // This call will throw an exception if the registry does not already running

}catch (RemoteException e){

System.out.println("RMI registry cannot be located at port "+port);

LocateRegistry.createRegistry(port);

System.out.println("RMI registry created at port " + port);

}

}

}

**VoteInterface.java**

import java.rmi.Remote;

public interface VoteInterface extends Remote{

//This interface is complete. Do not change it.

public String castVote(char vote )throws java.rmi.RemoteException;

}

**VoteServerImpl.java**

import java.rmi.\*;

import java.rmi.server.\*;

import java.util.ArrayList;

import java.util.List;

public class VoteServerImpl extends UnicastRemoteObject implements VoteInterface {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

List<Character> list = new ArrayList<Character>();

protected VoteServerImpl() throws RemoteException {

super();

}

@Override

public String castVote(char vote) throws RemoteException {

int yes =0,no=0;

String yCount = null;

String nCount = null;

list.add(vote);

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

{

char count= list.get(i);

if(count == 'y' || count =='Y')

{

yes = yes+1;

yCount = Integer.toString(yes);

}

else if(count == 'n' || count =='N')

{

no = no+1;

nCount = Integer.toString(no);

}

}

if(yCount == null)

{

yCount = "0";

}

else if(nCount== null)

{

nCount ="0";

}

return "Yes = "+yCount+", No = "+nCount;

}

}

**Client Side**:

**MyClient.java**

import java.rmi.\*;

import java.util.\*;

public class MyClient{

public static void main(String args[]){

Scanner keyboard = new Scanner(System.in);

try{

int port = 16790;

String host = "localhost";

String registryURL = "rmi://" + host + ":" + port + "/poll";

VoteInterface h = (VoteInterface)Naming.lookup(registryURL);

System.out.println("Enter poll: What is your vote?[yes|no] : ");

String msg = keyboard.nextLine();

System.out.println("Lookup completed " );

char vote = msg.charAt(0);

String message = h.castVote(vote);

System.out.println("Vote Count: " + message);

}

catch (Exception e){

e.printStackTrace();

}

}

}

**VoteInterface.java**

import java.rmi.Remote;

public interface VoteInterface extends Remote{

//This interface is complete. Do not change it.

public String castVote(char vote )throws java.rmi.RemoteException;

}

**Question 4**: Use cookies (and not MySQL) to accomplish the following:

1. Make a folder in drive C named: **Temp**.

2. Copy the following line to notepad editor (or any other text editor) and save the editor with file name: **names.txt** in the folder **Temp** (On each line there are two words separated by a single space):

Jane Doe

John Doe

Adam Smith

Ann Smith

Joe Bean

Mike Smith

Sara Doe

Rose Nigel

William Smith

David Depaoli

Thomas Iacovelli

Dominick Olsen

Danielle Williams

George Perez

Elaine Silva

Angela Dawnell

3. Make a servlet named: AddCookies and copy the following over it.

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Scanner;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/AddCookies")

**public** **class** AddCookies **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

//This servlet is complete. Do not change it.

**public** AddCookies() {

**super**();

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

addCookies(response);

System.***out***.println("The program terminated with no error.");

}

**private** **void** addCookies(HttpServletResponse response){

Scanner inputStream = **null**;

**try**{

inputStream = **new** Scanner(**new** File("C:\\Temp\\names.txt"));

**while**(**true**){

String name = inputStream.next();

String value = inputStream.next();

Cookie c = **new** Cookie(name, value);

c.setMaxAge(365 \* 24\*60\*60);

response.addCookie(c);

}

}**catch**(Exception e){

}

inputStream.close();

}

}

4. Load XAMPP and start: Apache.

5. Run the servlet: AddCookies. This servlet reads the file: names.txt one lien at a time to make cookies. The servlet makes a cookie for each line. On each line there are two words. The first word would be the name of the cookie and the second word would be the value of the cookie.

6. Make an html document and copy following over it:

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Q 4</title>

</head>

<body>

<form action=*"http://localhost:8080/Project/Main"*>

<center>

<pre>

User Name:<input name=*"name"* type=*"text"* value=*""*><br>

Password: <input name=*"password"* type=*"text"* value=*""*><br><br>

Changing Password<br>

New Password: <input name=*"newPassword"* type=*"text"* value=*""*><br>

Re enter The New Password:<input name=*"reEnterPassword"* type=*"text"* value=*""*><br>

<input name=*"changePassword"* type=*"Submit"* value=*"Change the Password"*>

</pre>

</center>

</form>

</body>

</html>

7. Make a servlet named: Main and copy the following over it:

/\*

1. If we make a cookie with a name and password and make a

second cookie with the same name and different password IE

replaces the first one.

2. For the sake of simplicity assume the user enters strings in all

the input boxes.

\*/

**import** java.io.\*;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.Cookie;

**import** java.util.Scanner;

@WebServlet("/Main")

**public** **class** Main **extends** javax.servlet.http.HttpServlet **implements** javax.servlet.Servlet {

**private** String head, tail;

**public** Main() {

**super**();

head = "<html><title>Not In The List</title><body><font color = blue><h1>";

tail = "</h1></font></body></html>";

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

//This method is complete. Do not change it.

String userName = request.getParameter("name");

String password = request.getParameter("password");

String newPassword = request.getParameter("newPassword");

String reEnteredPassword = request.getParameter("reEnterPassword");

displayCookies(request);

**if**(isIn(request, userName, password))

**if**(isTheSame(newPassword, reEnteredPassword))

replace(request, response, userName, password, newPassword);

**else**

isNotTheSame(response, newPassword, reEnteredPassword);

**else**

isNotInTheList(response, userName, password);

displayCookies(request);

}

**private** **boolean** isIn(HttpServletRequest request, String userName, String password){

//Complete this method.

//Check the cookies. If one of them has the name: userName and the value: password

//return false

**return** **false**; //Once this method is complete remove this line.

}

**private** **boolean** isTheSame(String newPassword, String reEnteredPassword){

//Complete this method

//Make sure the user entered the same new password twice. If both newPassword and

//reEnteredPassword have the same string return true otherwise return false.

**return** **false**; //Once this method is complete remove this line.

}

**private** **void** replace(HttpServletRequest request, HttpServletResponse response,

String userName, String password, String newPassword){

//Complete this method

// This is the case that the program is sure there is a cookie with the name:

// username and the value: password. Replace the value of this cookie by the

// value of: newPassword.

}

**private** **void** isNotInTheList(HttpServletResponse response, String userName, String password){

//Complete this method.

//Make an html document to display the content of the console.

System.***out***.println(userName + " with the password: " + password + " is not in the list");

}

**private** **void** isNotTheSame(HttpServletResponse response, String newPassword,

String reEnteredPassword){

//Complete this method.

//Make an html document to display the content of the console.

System.***out***.println(newPassword + " must be the same as " + reEnteredPassword);

}

**private** **void** success(HttpServletResponse response){

//Complete this method.

//Make an html document to display the content of the console.

System.***out***.println("The password is changed successfully.");

}

**private** **void** displayCookies(HttpServletRequest request){

//This method is complete. Do not change it.

Cookie[] cookies = request.getCookies();

**if** (cookies != **null** && cookies.length > 0)

**for**(**int** i=0; i<cookies.length; i++)

System.***out***.println(cookies[i].getName() + ", " + cookies[i].getValue());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

**Example 1**:

User enters a user name and password that has no cookie:



The response would be:



**Example 2**: The new password and reentered password are not the same:



The response would be:



**Example 3**: The user name and password has a cookie and want to change the password:



The response would be:



**Answer**

**Only copy the class: Main.java**

**Main.java**

import java.io.\*;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.eclipse.jdt.internal.compiler.lookup.SyntheticFactoryMethodBinding;

import javax.servlet.http.Cookie;

import java.util.Scanner;

@WebServlet("/Main")

public class Main extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private String head, tail;

public Main() {

super();

head = "<html><title>Not In The List</title><body><font color = blue><h1>";

tail = "</h1></font></body></html>";

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String userName = request.getParameter("name");

String password = request.getParameter("password");

String newPassword = request.getParameter("newPassword");

String reEnteredPassword = request.getParameter("reEnterPassword");

displayCookies(request);

if(isIn(request, userName, password))

if(isTheSame(newPassword, reEnteredPassword))

replace(request, response, userName, password, newPassword);

else

isNotTheSame(response, newPassword, reEnteredPassword);

else

isNotInTheList(response, userName, password);

displayCookies(request);

}

private boolean isIn(HttpServletRequest request, String userName, String password){

Cookie[] cookies = request.getCookies();

boolean flag = false;

for(int i=0; i<cookies.length;i++) {

if ((cookies[i].getName().equalsIgnoreCase(userName)) && cookies[i].getValue().equalsIgnoreCase(password))

{

flag= true;

break;

}

else

{

flag = false;

}

}

return flag;

}

private boolean isTheSame(String newPassword, String reEnteredPassword){

boolean flag = false;

if(newPassword.equalsIgnoreCase(reEnteredPassword))

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

private void replace(HttpServletRequest request, HttpServletResponse response,

String userName, String password, String newPassword){

Cookie[] cookies = request.getCookies();

Cookie c = null ;

for(int i=0; i<cookies.length;i++) {

c = cookies[i];

if(c.getName().equalsIgnoreCase(userName) && c.getValue().equalsIgnoreCase(password))

{

if(!newPassword.equalsIgnoreCase(null)&& !newPassword.isEmpty() && !newPassword.equalsIgnoreCase(""))

{

if(!c.getValue().equalsIgnoreCase(newPassword))

{

c.setValue(newPassword);

response.addCookie(c);

success(response);

}

else

{

response.setContentType("text/html");

PrintWriter out = null;

try {

out = response.getWriter();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

out.println("<html>\n" +

"<head><title>Processing Cookies</title></head>\n" +

"<body bgcolor=\"#fdf5e6\">\n" +

"<h1 align=\"center\">Old and New Password cannot be same</h1>\n" +

"</body></html>");

}

}

else

{

response.setContentType("text/html");

PrintWriter out = null;

try {

out = response.getWriter();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

out.println("<html>\n" +

"<head><title>Processing Cookies</title></head>\n" +

"<body bgcolor=\"#fdf5e6\">\n" +

"<h1 align=\"center\">Old or New Password cannot be empty</h1>\n" +

"</body></html>");

}

}

}

}

private void isNotInTheList(HttpServletResponse response, String userName, String password){

response.setContentType("text/html");

PrintWriter out = null;

try {

out = response.getWriter();

} catch (IOException e) {

e.printStackTrace();

}

out.println("<html>\n" +

"<head><title>Processing Cookies</title></head>\n" +

"<body bgcolor=\"#fdf5e6\">\n" +

"<h1 align=\"center\">"+userName+" with the password:"+password+" is not in the list</h1>\n" +

"</body></html>");

System.out.println(userName + " with the password: " + password + " is not in the list");

}

private void isNotTheSame(HttpServletResponse response, String newPassword,

String reEnteredPassword){

response.setContentType("text/html");

PrintWriter out = null;

try {

out = response.getWriter();

} catch (IOException e) {

e.printStackTrace();

}

out.println("<html>\n" +

"<head><title>Processing Cookies</title></head>\n" +

"<body bgcolor=\"#fdf5e6\">\n" +

"<h1 align=\"center\">"+newPassword + " must be the same as " + reEnteredPassword+"</h1>\n" +

"</body></html>");

System.out.println(newPassword + " must be the same as " + reEnteredPassword);

}

private void success(HttpServletResponse response){

response.setContentType("text/html");

PrintWriter out = null;

try {

out = response.getWriter();

} catch (IOException e) {

e.printStackTrace();

}

out.println("<html>\n" +

"<head><title>Processing Cookies</title></head>\n" +

"<body bgcolor=\"#fdf5e6\">\n" +

"<h1 align=\"center\">The password is changed successfully.</h1>\n" +

"</body></html>");

System.out.println("The password is changed successfully.");

}

private void displayCookies(HttpServletRequest request){

Cookie[] cookies = request.getCookies();

if (cookies != null && cookies.length > 0)

for(int i=0; i<cookies.length; i++)

System.out.println(cookies[i].getName() + ", " + cookies[i].getValue());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

5. Complete the following client-server program to do the following:

a) The server sends 1, or 2 to the client.

b) If the client receives 1 draws a rectangle. If it receives 2 draws a circle.

Note: The sizes of the rectangle and the circle is optional.

Note: Assume the user enters 1 or 2.

Note: You must use Integer.parseInt to convert a string received from the server to an integer.

Hints: Look at example 9 of lecture 6.

**Server**:

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.Scanner;

**public** **class** MyServer {

**public** **static** **void** main(String[] args){

//Complete this method.

}

}

**Client**:

**import** java.applet.Applet;

**import** java.awt.\*;

**import** java.io.\*;

**import** java.net.\*;

**public** **class** MyClient **extends** Applet {

**private** **int** s;

**public** **void** init(){

//Complete this method.

setSize(400, 400);

**try**{

//You may need repaint();

repaint();

}**catch**(Exception e){

System.***out***.println("Error: " + e);

System.*exit*(0);

}

}

**public** **void** paint(Graphics page){

//Complete this method.

}

}

**Example 1**:

The user has not entered 1 or 2:



**Server side**:

Enter 1, or 2: 1

**Client side**:



**Server side**:

Enter 1, or 2: 2

**Client side**:



**Answer**:

**Server**:

**MyServer.java**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class MyServer {

public static void main(String[] args){

try{

ServerSocket serverSocket = null;

serverSocket = new ServerSocket(16790);

Socket clientSocket = null;

clientSocket = serverSocket.accept();

Scanner keyboard = new Scanner(System.in);

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

System.out.print("Enter 1 or 2: ");

String a = keyboard.nextLine();

out.println(a);

out.close();

serverSocket.close();

clientSocket.close();

}catch(Exception e){

System.out.println("Error: " + e);

}

}

}

**Client**:

**MyClient.java**

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Scanner;

public class MyClient extends Applet {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private int s =1;

ServerSocket serverSocket;

Socket clientSocket;

public void init(){

setSize(400, 400);

try {

InetAddress host = InetAddress.getByName("localhost");

clientSocket =new Socket(host, 16790);

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

String input = in.readLine();

s = Integer.parseInt(input);

repaint();

} catch (IOException e) {

System.out.println("Error: " + e);

System.exit(0);

}

}

public void paint(Graphics page){

if(s == 1)

{

page.drawRect(0, 0,200,100);

}

else if(s== 2)

{

page.drawOval(0, 0,200,200);

}

}

public void windowClosing(java.awt.event.WindowEvent e) {

try {

clientSocket.close();

serverSocket.close();

} catch (IOException e1) {

e1.printStackTrace();

}

System.exit(0);

}

}