Socket programming

1. Example Program to Send and Receive a Message using Connectionless Sockets

```
import java.net.*;
import java.io.*;
class datagramReceiver{
 public static void main(String[ ] args){
 try{
 int MAX_LEN = 40;
 int localPortNum = Integer.parseInt(args[0]);
 DatagramSocket mySocket = new DatagramSocket(localPortNum);
 byte[] buffer = new byte[MAX_LEN];
 DatagramPacket packet = new DatagramPacket(buffer, MAX_LEN);
 mySocket.receive(packet);
 String message = new String(buffer);
 System.out.println(message);
 mySocket.close( );
 }
 catch(Exception e){e.printStackTrace( );}
 }
}
```

---

2. Program to Send a Single Datagram Packet

```
import java.net.*;
import java.io.*;

class datagramSender{
 public static void main(String[ ] args){
 try{
 InetAddress receiverHost = InetAddress.getByName(args[0]);
```

```
int receiverPort = Integer.parseInt(args[1]);
String message = args[2];
DatagramSocket mySocket = new DatagramSocket( );
byte[] buffer = message.getBytes( );
DatagramPacket packet = new DatagramPacket(buffer, buffer.length, receiverHost,
receiverPort);
mySocket.send(packet);
mySocket.close( );
}
catch(Exception e){ e.printStackTrace( ); }
}
```

3.Example Program to Send and Receive a Message in both Directions (Duplex Communication) using Connectionless Sockets

```
import java.net.*;
import java.io.*;
class datagramSenderReceiver{
 public static void main(String[ ] args){
 try{
 InetAddress receiverHost = InetAddress.getByName(args[0]);
 int receiverPort = Integer.parseInt(args[1]);
 String message = args[2];
 DatagramSocket mySocket = new DatagramSocket( );
 byte[] sendBuffer = message.getBytes( );
 DatagramPacket packet = new DatagramPacket(sendBuffer, sendBuffer.length,
 receiverHost, receiverPort);
 mySocket.send(packet);
 // to receive a message
 int MESSAGE_LEN = 60;
```

```
byte[ ] recvBuffer = new byte[MESSAGE_LEN];
DatagramPacket datagram = new DatagramPacket(recvBuffer, MESSAGE_LEN);
mySocket.receive(datagram);
String recvdString = new String(recvBuffer);
System.out.println("\n"+recvdString);

mySocket.close( );
}
catch(Exception e){ e.printStackTrace( ); }
}
}
```

---

3. Datagram Sender and Receiver (Sends First, Receives Next) Program

```
import java.net.*;
import java.io.*;
class datagramReceiverSender{
public static void main(String[ ] args){
try{
int MAX_LEN = 60;
int localPortNum = Integer.parseInt(args[0]);
DatagramSocket mySocket = new DatagramSocket(Integer.parseInt(localPortNum);
byte[ ] recvBuffer = new byte[MAX_LEN];
DatagramPacket packet = new DatagramPacket(recvBuffer, MAX_LEN);
mySocket.receive(packet);
String message = new String(recvBuffer);
System.out.println("\n"+message);
// to reply back to sender
InetAddress senderAddress = packet.getAddress( );
int senderPort = packet.getPort( );
```

```
// String messageToSend = args[1];
Scanner inputScanner = new Scanner(System.in);
String messageToSend = inputScanner.nextLine();
byte[ ] sendBuffer = messageToSend.getBytes();

DatagramPacket datagram = new DatagramPacket(sendBuffer, sendBuffer.length,
senderAddress, senderPort);
mySocket.send(datagram);
mySocket.close( );
}
catch(Exception e){e.printStackTrace( );}
}
}
```

Example Program to Send Objects of User-defined Classes using Stream-mode Sockets

```
import java.io.*;
class EmployeeData implements Serializable{
int empID;
String empName;
double empSalary;
void setID(int id){ empID = id; }
void setName(String name){ empName = name; }
void setSalary(double salary){ empSalary = salary; }
int getID( ){ return empID; }
String getName( ){ return empName; }
double getSalary( ){ return empSalary; }
}
```

Client Program to Send an Object of User-defined Class across Stream-mode Socket

```
import java.io.*;
import java.net.*;
import java.util.*;
class connectionClient{
 public static void main(String[] args){
 try{
 InetAddress serverHost = InetAddress.getByName(args[0]);
 int serverPortNum = Integer.parseInt(args[1]);
 Socket clientSocket = new Socket(serverHost, serverPortNum);
 EmployeeData empData = new EmployeeData( );
 Scanner input = new Scanner(System.in);
 System.out.print("Enter employee id: ");
 int id = input.nextInt( );
 input.nextLine( );
 System.out.print("Enter employee name: ");
 String name = input.nextLine( );
 System.out.print("Enter employee salary: ");
 double salary = input.nextDouble( );
 empData.setID(id);
 empData.setName(name);
 empData.setSalary(salary);
 ObjectOutputStream oos = new ObjectOutputStream(clientSocket.getOutputStream( ));
 oos.writeObject(empData);
 oos.flush( );
 clientSocket.close( );
 }
 catch(Exception e){e.printStackTrace( );}
```

```
 }
}
```

Server to Receive an Object of User-defined Class across a Stream-mode Sock

```
import java.io.*;
import java.net.*;
class connectionServer{
 public static void main(String[] args){
 try{
 int serverListenPortNum = Integer.parseInt(args[0]);
 ServerSocket connectionSocket = new ServerSocket(serverListenPortNum);
 Socket dataSocket = connectionSocket.accept( );
 ObjectInputStream ois = new ObjectInputStream(dataSocket.getInputStream( ));
 EmployeeData eData = (EmployeeData) ois.readObject( );
 System.out.println("Employee id : "+eData.getID( ));
 System.out.println("Employee name : "+eData.getName( ));
 System.out.println("Employee salary : "+eData.getSalary( ));
 dataSocket.close( );
 connectionSocket.close( );
 }
 catch(Exception e){e.printStackTrace( );}
 }
}
```