

USOCK

Uygulama Bağımsız Mesajlaşma Altyapısı

Ege Üniversitesi Bilgisayar Mühendisliği

Sunucu Yazılım Teknolojileri

Proje-1

Umut BENZER 05-06-7670 <http://www.ubenzer.com/>

Savaş YILDIZ 05-07-8569



İçindekiler

İçindekiler	2
Tasarım Aşaması.....	4
Genel Tanıtım	4
Platform ve Dil.....	4
Raporun Kapsamı	4
Sık Kullanılacak Terimler.....	4
USock.....	5
Paketler ve Sınıflar.....	5
com.ubenzer.usock.classes	5
com.ubenzer.usock.debug	6
com.ubenzer.usock.interfaces	6
Sınıf Diyagramı.....	7
USOCK Kütüphanesi Kullanım Kılavuzu	8
Adım 1: usock.jar paketini programınızın classpath'ına ekleyin	8
Adım 2: USOCK'ı başlatın.....	8
Adım 3: Veri Gönderin	9
Adım 4: Veri alın	12
Genel Çalışma Prensibi.....	14
UMesen	15
Giriş.....	15
Kullanım Kılavuzu	15
Programı Başlatmak	15
Programın Kullanımı	15
Programın Testi ve Diğer Notlar	15
Kullanıcı Arayüzü	16
Kaynak Kodlar	17
UMESEN.....	17
AboutBox.java	17
MainWindow.java	20
PortAndNickSelectionBox.java	29

UMESENApp.java.....	32
FileDataPackage.java.....	32
StringDataPackage.java.....	33
DefaultADP.java.....	34
USOCK.....	35
Host.java.....	35
IncomingMessagePasser.java.....	37
IncomingMessageProcessor.java.....	38
OutgoingMessageProcessor.java.....	40
Server.java.....	41
USock.java.....	42
Debug.java.....	45
ArrivedDataProcessor.java.....	46
IHost.java.....	46

Tasarım Aşaması

Genel Tanıtım

Uygulama bağımsız bir şekilde iletişim altyapısı gerçekleştirimi. Bu bağlamda, sunulmak üzere iki proje geliştirilmiştir. Bunlardan bir tanesi bir **JAVA Class Library** olan ve **jar** dosyası şeklinde dağıtılan, tüm JAVA projelerinde kolaylıkla referans gösterilerek kullanılacak **USOCK** mesajlaşma altyapısı ve diğeri de **USOCK** mesajlaşma altyapısını kullanarak makineler arası metin ve dosya transferine izin veren bir iletişim programı **UMESEN**'dir.

Platform ve Dil

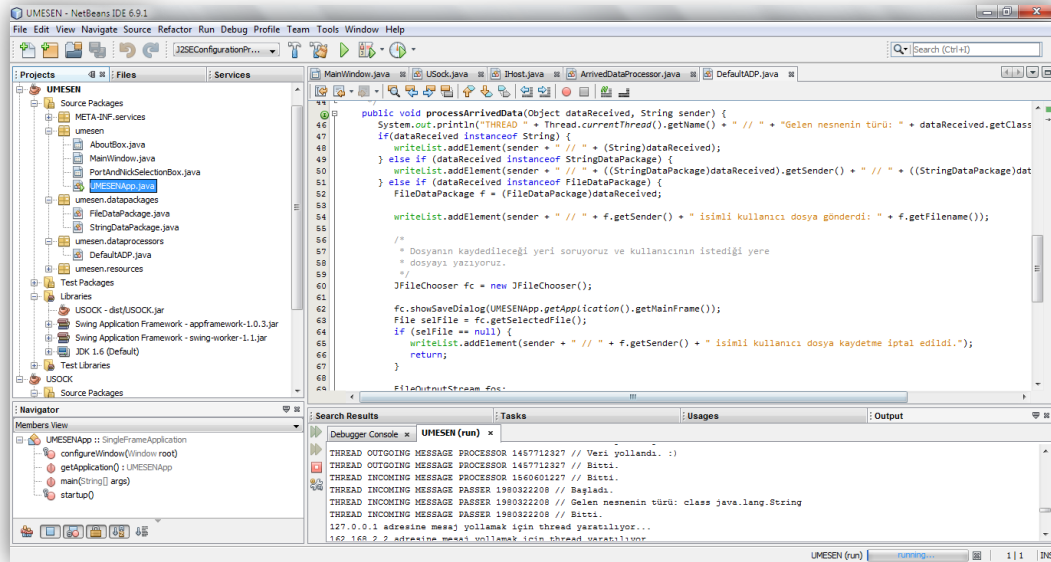
Proje NetBeans ortamında geliştirilmiştir. Arayüz NetBeans'ın sunduğu görsel editörden yararlanarak oluşturulmuştur.

Raporun Kapsamı

UMESEN iletişim altyapısı değil, bunu kullanan bir uygulama olduğundan ve yorum satırları ile tüm sınıflar ayrıntılı bir şekilde açıklandığından bu konu üzerinde çok durulmayacak, **USOCK** ise ayrıntısıyla anlatılacaktır.

Sık Kullanılacak Terimler

Raporda sıkça **KULLANICI** terimi geçecektir. **KULLANICI**, **USOCK** altyapısını kullanan herhangi bir yazılım anlamına gelmekte olup, size verilen örnekte bu uygulama **UMESEN**'dir.



USock

Paketler ve Sınıflar

Projede toplam üç adet paket kullanılmıştır. Paketler ve paketlerde bulunan sınıflar aşağıda belirtilmiştir. Sınıfların ne işe yaradığı ve metotlarının neler yaptığı konusundaki ayrıntılı bilgiler JavaDoc olarak kaynak kodu ile beraber sunulduğundan buraya bir kopyası daha eklenmemiştir.

com.ubenzer.usock.classes

Bu paket USOCK iletişim altyapısını oluşturan sınıfları içermektedir.

Host

Host sınıfı “kendisiyle iletişim kurulabilecek” daha net bir anlamda “kendisine veri aktarılabilir” adresi ve port numarası olan bir uzak bilgisayarı temsil etmektedir.

IncomingMessagePasser (Thread)

Bu sınıf uzak makineden alınmış olan bir mesajın işlenmek üzere KULLANICI programına iletilmesinden sorumludur.

IncomingMessageProcessor (Thread)

Bu sınıf bağlantı isteğinde bulunan uzak makinenin bağlantı isteğinin kabul edilmesi, gelen verinin alınması ve bu verinin IncomingMessagePasser sınıfından bir süreç yaratılarak buna geçirilmesinden sorumludur.

OutgoingMessageProcessor (Thread)

Bu sınıf uzaktaki bir makineye (kendisi Host sınıfı ve IHost arayüzü ile temsil edilir) bağlantı kurmak ve aktarılacak olan veriyi yollamakla mükelleftir.

Server (Thread)

Sürekli olarak USock iletişim altyapısı kullanılmak üzere yapılandırılırken belirtilmiş olan portu gelen istekler için dinlemekle sorumlu olan sınıftır. Bir veri aktarım isteği geldiğinde bu sınıf yeni bir IncomingMessageProcessor yaratarak işi bu sürece atmakta, dinlemeye devam etmektedir.

USock

USOCK altyapısının ilk yapılandırmasını da sağlayan, IHost’ların bir listesinin tutulabileceği, gelen isteklerin dinlenmesini başlatan ana sınıftır.

com.ubenzer.usock.debug

Bu paket yazılan JAVA kütüphanesinde kolaylıkla hata ayıklamak, önemli bilgileri konsol ekranına yazdırmak ve gerektiğinde de bu özelliği kolayca kapatmak üzere tasarlanmış bir pakettir. Bu paket yazılım geliştirmeye yönelik olup içindekilerin çalışan bir uygulamada yaptığı herhangi bir iş bulunmamaktadır.

Debug

Gönderilen bilgilerin el ile hata ayıklama ve bilgilendirilme amaçlı olarak konsol ekranına yazıldığı basit bir sınıftır. System.out.println yerine bilgilerin bu sınıfa gösterilmesinin başlıca iki amacı vardır:

- Sınıfın içeriği kolayca değiştirilebilir, böylece ileride bilgiler konsola yazdırılmayabilir.
- Uzun uzun System.out.println yerine Debug.log yazmak daha kolay ve hızlıdır.

com.ubenzer.usock.interfaces

Bu paket KULLANICI sınıfı tarafından kullanılacak ve kullanılmak zorunda olan bir dizi interface barındırmaktadır.

ArrivedDataProcessor

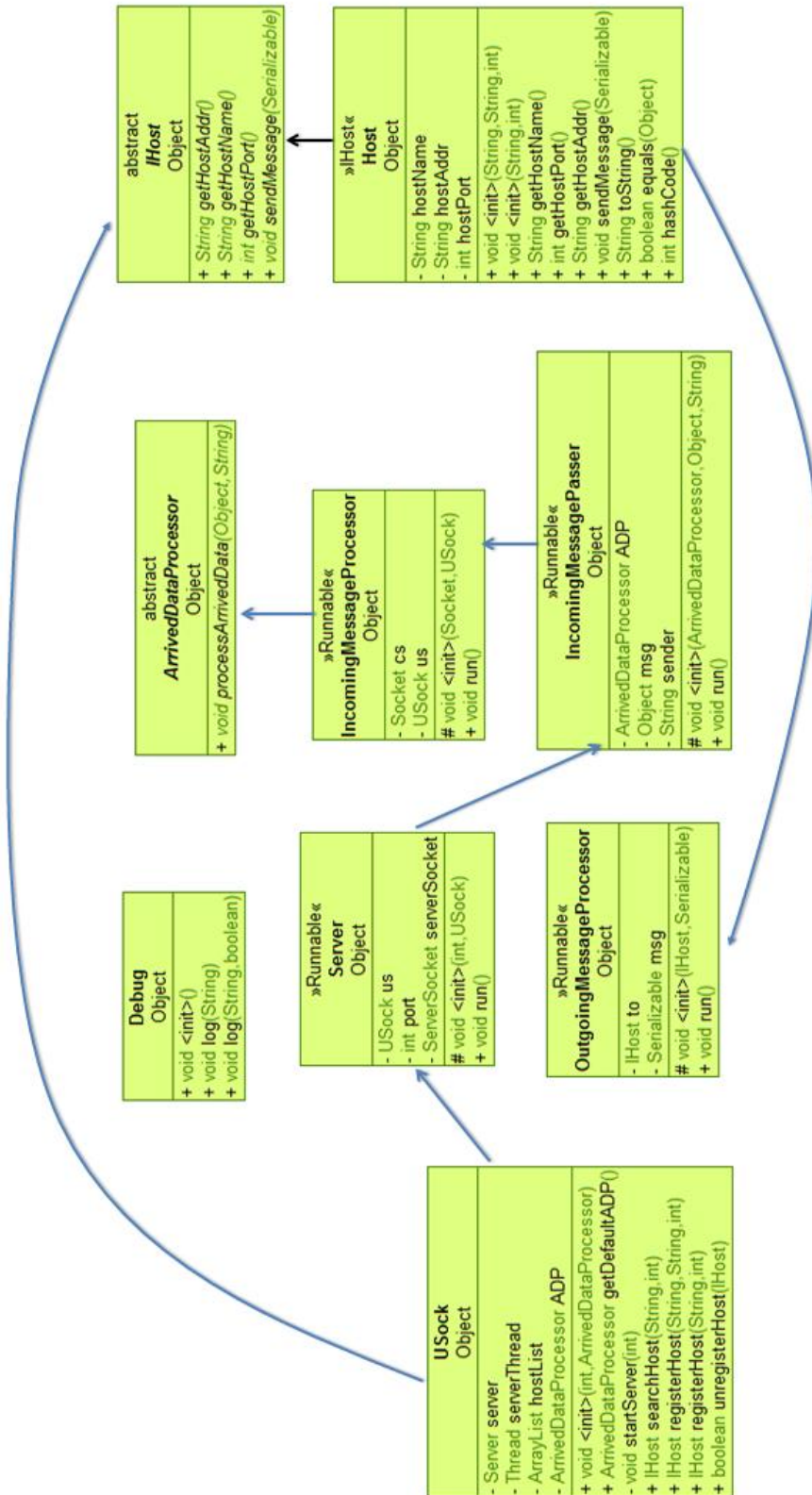
Bu interface gelen verinin nasıl işleneceğini belirler. Uzak bilgisayardan gönderilmiş herhangi bir veri işlenmesi için USOCK'tan KULLANICI'ya geçirilmelidir. Gelen veriyi alacak olan kullanıcı sınıfı mutlaka bu interfacei implement etmelidir.

IHost

USOCK altyapısının bir "Host"u kullanabilmesi için hostun bu interfacei implement etmesi gerekmektedir.

Sınıf Diyagramı

Yukarıdaki sınıflar kullanıldığında aşağıdaki UML Class Diyagram ortaya çıkmaktadır. Bu diyagramdaki siyah oklar "implements" anlamına gelirken mavi oklar ise sınıfların hangilerinin birbirleriyle bağlantılı olduğunu/birbirini çağırıldığını göstermektedir.

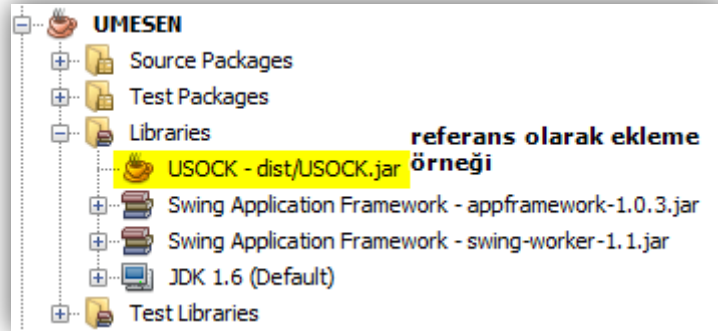


USOCK Kütüphanesi Kullanım Kılavuzu

Unutulmamalıdır ki USOCK Kütüphanesi yazılan tüm JAVA uygulamalarında kolaylıkla kullanılabilir. Yazının bu kısmında sıfırdan geliştirilen bir projede USOCK'un nasıl kullanılacağı anlatılmaktadır.

Adım 1: usock.jar paketini programınızın classpath'ına ekleyin

USOCK kütüphanesini kullanabilmek için öncelikle bu kütüphaneyi projenize referans olarak eklemeniz gerekmektedir. Bunu yapmak her IDE'de farklı bir adım gerektirmekte olduğundan ayrıntılı bilgi için kullandığınız IDE'nin yardım dokümanına bakmanız gerekmektedir.



Adım 2: USOCK'ı başlatın

USOCK kütüphanesini bu kütüphane aracılığı ile veri almaya başlamadan önce yapılandırmanız. Eğer sadece veri gönderecekseniz bu adımı atlayabilirsiniz.

USOCK kütüphanesini başlatmak için, bir USOCK nesnesi yaratmanız yeterlidir. Bu, gelen istekleri dinlemek üzere arka planda bir thread başlatacak, programınızın işleyişini etkilemeyecektir.

Dilerseniz birden fazla USOCK nesnesi yaratabilir, hepsi ile farklı birer portu dinleyebilirsiniz.

USOCK sınıfını kullanabilmek için aşağıdaki "import"ları yapmanız gerekmektedir:

```
import com.ubenzer.usock.classes.USock;  
import com.ubenzer.usock.interfaces.ArrivedDataProcessor;
```

Bu sınıfın constructoru aşağıdaki gibidir:

```
/**
 * Yeni bir USock iletişim altyapısı yaratılır. Bu yapı iki yönlüdür.
 * Her USock nesnesi, gelen istekleri alabilmek adına bir port dinleme ihtiyacı
 * duyar. ArrivedDataProcessor ise, bu USock nesnesine uzaktan gelen isteklerin
 * hangi kullanıcı sınıfına aktarılacağını belirtir.
 *
 * KULLANICI: USock altyapısını kullanan program.
 * USOCK: Bizim altyapımız.
 *
 * @param Gelen istekler için dinlenecek olan port numarası
 * @param Gelen verilerin hangi kullanıcı classına devredileceği
 * @throws Hatalı verileri sevmeyiz.
 */
public USock(int portToBeListened, ArrivedDataProcessor ADP) throws Exception {
    if(portToBeListened < 0) throw new Exception("Port 0'dan küçük olamaz.");
    if(portToBeListened > 65535) throw new Exception("Port 65535'ten büyük olamaz.");

    this.ADP = ADP;

    this.startServer(portToBeListened);
}
```

Burada ilk parametre “bu bilgisayar”a gelecek veri aktarımı istekleri için hangi portun dinleneceğidir. Unutulmamalıdır ki, başka bir uygulamanın zaten dinlemekte olduğu bir portu USOCK kütüphanesi dinleyemez. Böyle bir durumda IOException atılacaktır.

İkinci parametre ise “uzaktaki bilgisayardan gelen bilgilerin” hangi nesneye geçirileceğinin belirlenmesini sağlar. Uzaktan bir veri geldiğinde ve başarılı bir şekilde indirildiğinde işlenmesi için sizin uygulamanıza bu nesne aracılığı ile geçirilecektir. Bu nesne mutlaka ama mutlaka ArrivedDataProcessor arayüzünü implement etmelidir.

Aşağıda USOCK nesnesinin örnek bir yapılandırılması görüntülenmektedir:

```
try {
    /* Bu satır aracılığı ile iletişim alt yapımızı başlatıyoruz. */
    usock = new USock(port, (ArrivedDataProcessor) (new DefaultADP(this.mesajListModel)));
} catch (Exception ex) {
    /* Eğer port zaten kullanımda ise Exception atacaktır. O zaman yeniden port soruyoruz. */
    startServer();
}
```

ArrivedDataProcessor arayüzünün içerisinde neler olduğu ileriki kısımlarda anlatılacaktır.

Adım 3: Veri Gönderin

USock ile veri göndermek çok kolaydır. Veri göndermek için bir uzak bilgisayara ihtiyacınız vardır. Uzak bilgisayarlar bu programda IHost arayüzü ile temsil edilmektedir.

Bu yüzden öncelikle veri göndereceğiniz bilgisayar için bir Host nesnesi oluşturmanız gerekir. IHost ara yüzünü kullanabilmek için aşağıdaki “import”u yapmış olmanız gerekmektedir:

```
import com.ubbenzer.usock.interfaces.IHost;
```

Yeni bir “Host” eklemek için USock sınıfındaki registerHost metodu çağırılmalıdır. Bu metot geriye bizim “Hostumuzu” döndürecektir:

```
/**
 * USock altyapısına daha sonra hızlıca erişmek için yeni bir host kaydet.
 *
 * @param Hostun adı (tamamen görsel amaçlı)
 * @param Hostun adresi (IP)
 * @param Hostun portu (Uzak makine hangi portu dinliyor?)
 * @return Oluşturulan ve register edilen IHost nesnesi.
 */
public IHost registerHost(String hostName, String hostAddress, int port) {
    IHost host = new Host(hostName,hostAddress,port);
    for(IHost h:hostList) {
        if(h.equals(host)) return h;
    }
    hostList.add(host);
    return host;
}
/**
 * USock altyapısına daha sonra hızlıca erişmek için yeni bir host kaydet.
 *
 * @param Hostun adresi (IP)
 * @param Hostun portu (Uzak makine hangi portu dinliyor?)
 * @return Oluşturulan ve register edilen IHost nesnesi.
 */
public IHost registerHost(String hostAddress, int port) {
    IHost host = new Host(hostAddress,port);
    for(IHost h:hostList) {
        if(h.equals(host)) return h;
    }
    hostList.add(host);
    return host;
}
```

Bu metoda isteğe bağlı bir host adı ve zorunlu adres ile port bilgileri verilmelidir. Host adı sadece bilgi amaçlıdır, başka bir kullanımı yoktur. Adres, “ulaşılabilen” herhangi bir adres olabilir. Bu bir ağ adresi veya bir IP adresi olabilir. Port bilgisi ile karşı bilgisayarın gelen istekler için dinlediği port numarasıdır.

Örnek kullanım aşağıda yer almaktadır. Bu, kendi kendimize mesaj atabilmemiz için 127.0.0.1’i kendimize eklemektedir.

```
IHost h = usock.registerHost("Kendiniz","127.0.0.1",port);
```

Bir defa “Host” yarattıktan sonra mesaj yollamak çok kolaydır. IHost.sendMessage metodu ile bu gerçekleştirilebilir.

```

/**
 * Hosta bir mesaj yollar.
 *
 * @param Herhangi bir Serializable sınıf.
 */
public void sendMessage(Serializable msg);

```

Metodun açıklamasından da görülebileceği gibi, gönderilen mesaj JAVA'nın Serializable sınıfını implement ettiği sürece her türlü veri nesnesi olabilir. Dilerseniz siz kendi sınıflarınızı geliştirebileceğiniz gibi JAVA'nın kendi sınıflarının da büyük bir çoğunluğunu kullanabilirsiniz.

Bu metot veri gönderimini ayrı bir iş parçacığında yapar. Böylece bu metodu çalıştırdığınız halde programınız kilitlenmez.

Verinin aktarılamaması durumunda oluşan Exception konsol ekranına yazılır. İleriki sürümlerde daha ayrıntılı bir hata ayıklama mekanizması geliştirilmesi planlanmaktadır.

Aşağıda değişik veri paketleri için bazı örnekler gösterilmiştir:

```

// Seçili hostların listesini alalım.
int[] selectedIx = jHostList.getSelectedIndices();

// Bu hostlara teker teker gönderilecek veriyi yollayalım.
/* Gönderme işlemi threaded olduğu için bir diğeri için
 * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
 */
for (int i=0; i<selectedIx.length; i++) {
    IHost sel = (IHost) jHostList.getModel().getElementAt(selectedIx[i]);
    sel.sendMessage(jText.getText());
}

```

**JAVA STRING YOLLAMA
ÖRNEĞİ**

```

// Seçili hostların listesini alalım.
int[] selectedIx = jHostList.getSelectedIndices();

// Bu hostlara teker teker gönderilecek veriyi yollayalım.
/* Gönderme işlemi threaded olduğu için bir diğeri için
 * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
 */
for (int i=0; i<selectedIx.length; i++) {
    IHost sel = (IHost) jHostList.getModel().getElementAt(selectedIx[i]);
    sel.sendMessage(new StringDataPackage(jText.getText(),this.nickname));
}

```

**Kullanıcının kendi yarattığı bir veri
paketini yollama örneği**

```

JFileChooser fc = new JFileChooser();

// Dosya açma dialogunu göster
fc.showOpenDialog(UMESENApp.getApplication().getMainFrame());
File selFile = fc.getSelectedFile();
if (selFile == null) return; // Cancel

// Seçili hostların listesini alalım.
int[] selectedIx = jHostList.getSelectedIndices();

// Bu hostlara teker teker gönderilecek veriyi yollayalım.
/* Gönderme işlemi threaded olduğu için bir diğeri için
 * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
 */
for (int i=0; i<selectedIx.length; i++) {
    IHost sel = (IHost) jHostList.getModel().getElementAt(selectedIx[i]);

    byte [] fileByte = new byte [(int)selFile.length()];
    /* Dosyayı okuyoruz */
    FileInputStream fis;
    try {
        fis = new FileInputStream(selFile);
        fis.read(fileByte,0,fileByte.length);
        FileDataPackage fp = new FileDataPackage(selFile.getName(), fileByte, this.nickname);
        sel.sendMessage(fp);
    } catch (FileNotFoundException ex) {
        System.out.println("FileNotFoundException yedik: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("IOException yedik: " + ex.getMessage());
    }
}
}

```

Kısacası yollayabileceğiniz veriler hayal gücünüz ve JAVA'nın yetenekleri ile sınırlıdır.

Adım 4: Veri alın

USOCK altyapısı gelen istekleri otomatik olarak kabul eder ve dosya aktarımını gerçekleştirir. Bunlar arka planda çalışan süreçler ile yürütüldüğünden KULLANICI yazılımını etkilememektedir. Alınan veriler en son aşamada USock nesnesi yaratılırken kendisine parametre geçirilmiş ArrivedDataProcessor'u kullanarak bu bilgiyi KULLANICI yazılımına geçirir.

ArrivedDataProcessor arayüzü aşağıdaki gibidir:

```

package com.ubenzer.usock.interfaces;

/**
 * Gelen verileri işleyecek KULLANICI sınıfı mutlaka bu
 * arayüzü implement etmelidir.
 *
 * Zaten implement etmesi çok karışık bir şey de değildir. :)
 *
 * @author UB
 */
public interface ArrivedDataProcessor {

    /**
     * Gelen veriyi işler.
     *
     * @param Nesne halinde gelen veri
     * @param Yollayanın adresi
     */
    public void processArrivedData(Object dataReceived,String sender);

}

```

Yeni bir nesne gelince ArrivedDataProcessor'un processArrivedData metodu çağırılır. Bu metotta Object gelen veridir. KULLANICI sınıfı bu nesneyi alıp işlemeli ve gerekeni yapmalıdır. String ise verinin geldiği istemcinin adresidir.

Aşağıda örnek bir ArrivedDataProcessor görülmektedir:

```

public void processArrivedData(Object dataReceived, String sender) {
    System.out.println("THREAD " + Thread.currentThread().getName() + " // " + "Gelen nesnenin türü: " + c
if(dataReceived instanceof String) {
    writeList.addElement(sender + " // " + (String)dataReceived);
} else if (dataReceived instanceof StringDataPackage) {
    writeList.addElement(sender + " // " + ((StringDataPackage)dataReceived).getSender() + " // " + ((S
} else if (dataReceived instanceof FileDataPackage) {
    FileDataPackage f = (FileDataPackage)dataReceived;

    writeList.addElement(sender + " // " + f.getSender() + " isimli kullanıcı dosya gönderdi: " + f.get

    /*
     * Dosyanın kaydedileceği yeri soruyoruz ve kullanıcının istediği yere
     * dosyayı yazıyoruz.
     */
    JFileChooser fc = new JFileChooser();

    fc.showSaveDialog(UMESENApp.getApplication().getMainFrame());
    File selFile = fc.getSelectedFile();
    if (selFile == null) {
        writeList.addElement(sender + " // " + f.getSender() + " isimli kullanıcı dosya kaydetme iptal e
        return;
    }

    FileOutputStream fos;
    try {
        fos = new FileOutputStream(selFile);
        fos.write(f.getFile());
        fos.close();
    }
}

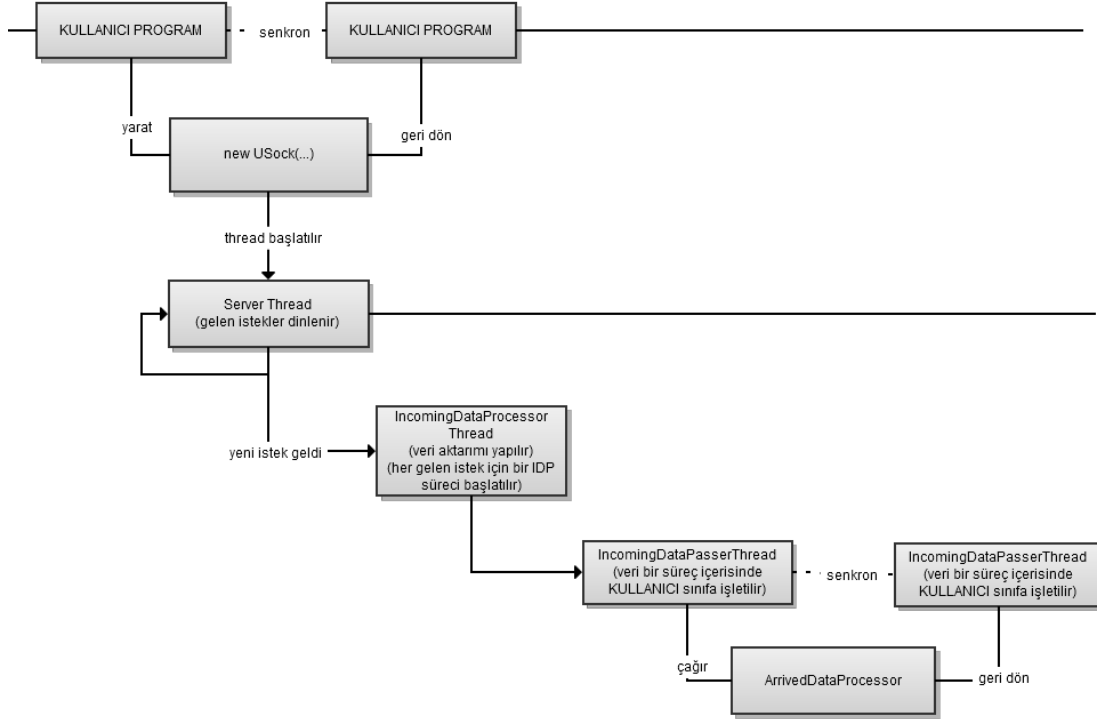
```

Bu kolar sadece fikir vermek amaçlı olarak konulmuştur ve tam değildir. Tam kodlar için kaynak kodları kısmına bakabilirsiniz. Bu kısmı KULLANICI yazılımında istediğiniz ihtiyaçlarınız için istediğiniz gibi yazabilirsiniz.

Genel Çalışma Prensibi

Aşağıdaki şema USOCK'un nasıl çalıştığı konusunda daha iyi bir fikir edinmenizi sağlayacaktır. Bu şema USOCK'un gelen mesajları işleme çizelgesidir. Düz çizgiler asenkron olup beraber çalışanları, kesikli çizgiler ise kilitlenen metotları göstermektedir.

Veri gönderme tamamen ayrı olup kendi sürecinde çalıştığından şemaya alınmamıştır.



UMesen

Giriş

UMesen, USOCK kütüphanesini iletişim altyapısı olarak kullanan örnek bir programdır. Bu program örnek olması açısından birisi JAVA'nın String sınıfı olmak üzere toplamda üç farklı sınıfı veri paketi olarak kullanmakta ve bunları USOCK ile uzak bilgisayara yollamaktadır.

Program UMESENAApp.java main sınıfı ile başlamaktadır. Projenin derlenmesi, bir ortamda çalıştırılması gibi konular IDE bağımlı olduğundan bu konulara girilmeyecektir. CD içerisindeki kodlar NetBeans projesi olarak açılabilir.

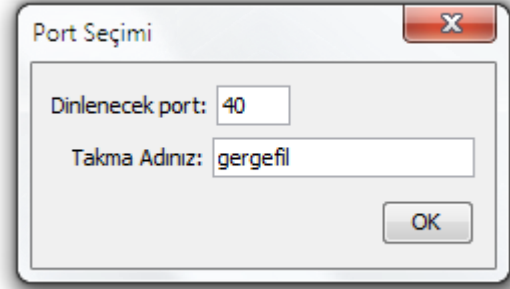
Eğer derlenmiş halini açmak isterseniz aşağıdaki kodu kullanabilirsiniz:

```
java -jar UMESEN.jar
```

Kullanım Kılavuzu

Programı Başlatmak

Program başladığında sizden takma ad ve dinlenecek port numarasını isteyecektir. **Varsayılan 40. porttur.** Eğer 40.port kullanımda ise buradan elle başka port girebilirsiniz. Eğer kullanımda iken bu portu seçmek isterseniz size tekrar aynı kutucuk gelecektir. **Eğer bu programın iki kopyasını aynı bilgisayarda açarsanız ilk program 40.portu kapa- cağından ikinci kopyaya başka bir port numarası atamalısınız.**



Programın Kullanımı

Programın ara yüzünde neyin ne olduğu bir sonraki sayfada bulunan şekilde açıklanmıştır. **Gönderilen veriler, gelen istekler, süreçlerin çalışmaya başlaması ve sonlanması, veri gönderiminde ve alın- da oluşan Exception'lar ve buna benzer bilgileri konsol ekranından takip edebilirsiniz.**

Programın Testi ve Diğer Notlar

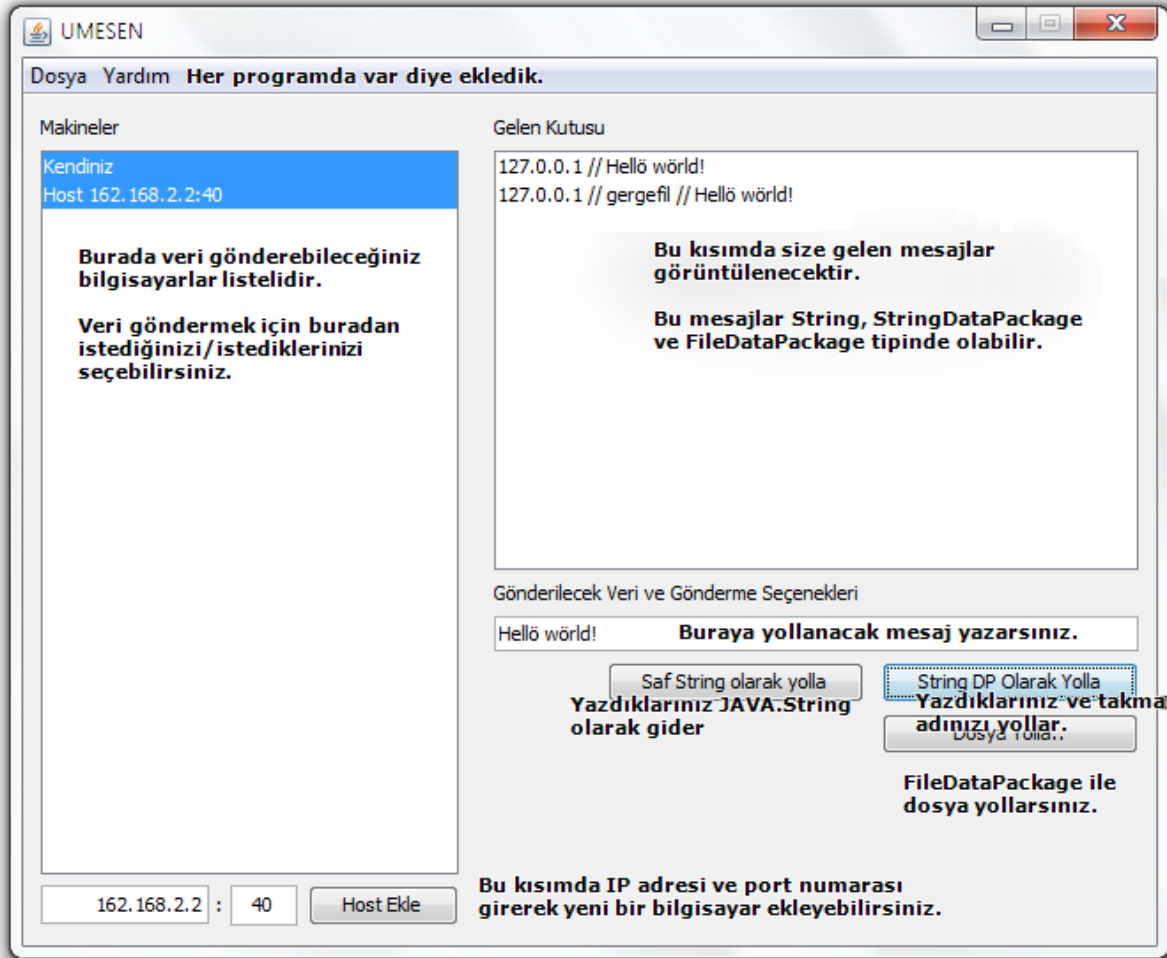
Program yazıldıktan sonra hem mesajlaşmada hem de çok yüksek boyutlu dosyaların aktarılmasında test edilmiştir.

Aynı anda birden fazla veri aktarılması test edilmiştir. Bu verilerin çift yönlülüğü test edilmiştir. Bu program aracılığı ile yerel ağdaki iki bilgisayar arasında The Simpsons dizisinin yaklaşık 150MB'lık bir bölümü transfer edilmiştir.

LÜTFEN DİKKAT: Bu raporda yer alan kaynak kodları ve ekran görüntüleri projenin son halini tam yansıtmayabilir. Bu kodlar ve ekran görüntüleri otomatik güncellenmediğinden rapor yazıldıktan sona

yakalanan bir hata sonrası kodların değiştirilmesi durumunda bu rapora yansımayacaktır.

Kullanıcı Arayüzü



Kaynak Kodlar

UMESEN

AboutBox.java

```
1 /*
2  * Hakkında kutusunu görüntülemek dışında
3  * hiçbir şey yapmaz.
4  *
5  */
6
7 package umesen;
8
9 import org.jdesktop.application.Action;
10
11 public class AboutBox extends javax.swing.JDialog {
12
13     public AboutBox(java.awt.Frame parent) {
14         super(parent);
15         initComponents();
16         getRootPane().setDefaultButton(closeButton);
17     }
18
19     @Action public void closeAboutBox() {
20         dispose();
21     }
22
23     // <editor-fold defaultstate="collapsed" desc="Generated Code">
24     private void initComponents() {
25
26         closeButton = new javax.swing.JButton();
27         javax.swing.JLabel appTitleLabel = new javax.swing.JLabel();
28         javax.swing.JLabel versionLabel = new javax.swing.JLabel();
29         javax.swing.JLabel appVersionLabel = new javax.swing.JLabel();
30         javax.swing.JLabel vendorLabel = new javax.swing.JLabel();
31         javax.swing.JLabel appVendorLabel = new javax.swing.JLabel();
32         javax.swing.JLabel homepageLabel = new javax.swing.JLabel();
33         javax.swing.JLabel appHomepageLabel = new javax.swing.JLabel();
34         javax.swing.JLabel appDescLabel = new javax.swing.JLabel();
35         javax.swing.JLabel imageLabel = new javax.swing.JLabel();
36
37         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
38         org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(umesen.UMESENAApp.class).getContext().getResourceMap(AboutBox.class);
39         setTitle(resourceMap.getString("title")); // NOI18N
40         setModal(true);
41         setName("aboutBox"); // NOI18N
42         setResizable(false);
43
44         javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(umesen.UMESENAApp.class).getContext().getActionMap(AboutBox.class, this);
```

```

45     closeButton.setAction(actionMap.get("closeAboutBox")); // NOI18N
46     closeButton.setName("closeButton"); // NOI18N
47
48     appTitleLa-
bel.setFont(appTitleLabel.getFont().deriveFont(appTitleLabel.getFont().getStyle()
| java.awt.Font.BOLD, appTitleLabel.getFont().getSize()+4));
49     appTitleLabel.setText(resourceMap.getString("Application.title")); //
NOI18N
50     appTitleLabel.setName("appTitleLabel"); // NOI18N
51
52     versionLa-
bel.setFont(versionLabel.getFont().deriveFont(versionLabel.getFont().getStyle() |
java.awt.Font.BOLD));
53     versionLabel.setText(resourceMap.getString("versionLabel.text")); //
NOI18N
54     versionLabel.setName("versionLabel"); // NOI18N
55
56     appVersionLabel.setText(resourceMap.getString("Application.version"));
// NOI18N
57     appVersionLabel.setName("appVersionLabel"); // NOI18N
58
59     vendorLa-
bel.setFont(vendorLabel.getFont().deriveFont(vendorLabel.getFont().getStyle() |
java.awt.Font.BOLD));
60     vendorLabel.setText(resourceMap.getString("vendorLabel.text")); //
NOI18N
61     vendorLabel.setName("vendorLabel"); // NOI18N
62
63     appVendorLabel.setText(resourceMap.getString("Application.vendor")); //
NOI18N
64     appVendorLabel.setName("appVendorLabel"); // NOI18N
65
66     homepageLa-
bel.setFont(homepageLabel.getFont().deriveFont(homepageLabel.getFont().getStyle()
| java.awt.Font.BOLD));
67     homepageLabel.setText(resourceMap.getString("homepageLabel.text")); //
NOI18N
68     homepageLabel.setName("homepageLabel"); // NOI18N
69
70     appHomepageLabel.setText(resourceMap.getString("Application.homepage"));
// NOI18N
71     appHomepageLabel.setName("appHomepageLabel"); // NOI18N
72
73     appDescLabel.setText(resourceMap.getString("appDescLabel.text")); //
NOI18N
74     appDescLabel.setName("appDescLabel"); // NOI18N
75
76     imageLabel.setIcon(resourceMap.getIcon("imageLabel.icon")); // NOI18N
77     imageLabel.setName("imageLabel"); // NOI18N
78
79     javax.swing.GroupLayout layout = new ja-
vax.swing.GroupLayout(getContentPane());
80     getContentPane().setLayout(layout);
81     layout.setHorizontalGroup(
82         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
83             .addGroup(layout.createSequentialGroup()
84                 .addComponent(imageLabel)

```

```

85         .addGap(18, 18, 18)
86
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
87         .addGroup(javax.swing.GroupLayout.Alignment.LEADING, la-
yout.createSequentialGroup())
88
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
89         .addComponent(versionLabel)
90         .addComponent(vendorLabel)
91         .addComponent(homepageLabel))
92
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
93
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
94         .addComponent(appVersionLabel)
95         .addComponent(appVendorLabel)
96         .addComponent(appHomepageLabel)))
97         .addComponent(appTitleLabel, ja-
vax.swing.GroupLayout.Alignment.LEADING)
98         .addComponent(appDescLabel, ja-
vax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
266, Short.MAX_VALUE)
99         .addComponent(closeButton))
100        .addContainerGap())
101    );
102    layout.setVerticalGroup(
103        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
104        .addComponent(imageLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
105        .addGroup(layout.createSequentialGroup()
106            .addComponent(appTitleLabel)
107            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
108            .addComponent(appDescLabel, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, ja-
vax.swing.GroupLayout.PREFERRED_SIZE)
109            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
110            .addGroup(layout.createSequentialGroup()
111                .addComponent(versionLabel)
112                .addComponent(appVersionLabel))
113            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
114            .addGroup(layout.createSequentialGroup()
115                .addComponent(vendorLabel)
116                .addComponent(appVendorLabel))
117            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
118            .addGroup(layout.createSequentialGroup()
119                .addComponent(homepageLabel)
120                .addComponent(appHomepageLabel))
121            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 33,

```

```

Short.MAX_VALUE)
123         .addComponent(closeButton)
124         .addContainerGap())
125     );
126
127     pack();
128 }// </editor-fold>
129
130 // Variables declaration - do not modify
131 private javax.swing.JButton closeButton;
132 // End of variables declaration
133
134 }
135
136

```

MainWindow.java

```

1 package umesen;
2
3 import umesen.dataprocessors.DefaultADP;
4 import java.io.FileNotFoundException;
5 import umesen.datapackages.StringDataPackage;
6 import com.ubenzer.usock.classes.USock;
7 import com.ubenzer.usock.interfaces.ArrivedDataProcessor;
8 import com.ubenzer.usock.interfaces.IHost;
9 import java.io.File;
10 import java.io.FileInputStream;
11 import java.io.IOException;
12 import javax.swing.DefaultListModel;
13 import org.jdesktop.application.Action;
14 import org.jdesktop.application.SingleFrameApplication;
15 import org.jdesktop.application.FrameView;
16 import javax.swing.JDialog;
17 import javax.swing.JFileChooser;
18 import javax.swing.JFrame;
19 import umesen.datapackages.FileDataPackage;
20
21 /**
22  * Verilerin kullanıcıdan alınarak iletildiği ve gelen
23  * verilerin UI'de gösterildiği ana formdur.
24  *
25  * @author UB
26  */
27 public class MainWindow extends FrameView {
28
29     public USock usock; // bağlantı kütüphanemiz
30     int port = 0; //bağlantı kurulacak port
31     String nickname = ""; // nickname
32     DefaultListModel hostListModel = new DefaultListModel(); // listeleri tut-
mak için gerekli modeller
33     DefaultListModel mesajListModel = new DefaultListModel();
34
35     public MainWindow(SingleFrameApplication app) {
36         super(app);
37         initComponents();
38         getFrame().setResizable(false); // Bu satırı bulmak samimi söylüyorum 3

```

saatimi aldı!

```
39
40     startServer(); /* Dinlemeye başlama hazırlıkları */
41 }
42
43 private void startServer() {
44     /* Port ve Takma isim alınmak üzere form işlemleri yapılır. */
45     port = -1;
46     if (portSelectionBox == null) {
47         JFrame mainFrame = UMESENAApp.getApplication().getMainFrame();
48         portSelectionBox = new PortAndNickSelectionBox(mainFrame, this);
49         portSelectionBox.setLocationRelativeTo(mainFrame);
50     }
51     UMESENAApp.getApplication().show(portSelectionBox);
52
53     if(port == -1) System.exit(0);
54
55     try {
56         /* Bu satır aracılığı ile iletişim alt yapımızı başlatıyoruz. */
57         usock = new USock(port, (ArrivedDataProcessor) (new DefaultADP(this.mesajListModel)));
58     } catch (Exception ex) {
59         /* Eğer port zaten kullanımda ise Exception atacaktır. O zaman yeniden port soruyoruz. */
60         startServer();
61     }
62
63     /* Kolaylık olması amacı ile kendimizi listeye ekleyelim. */
64     // Bu adım mecburi değildir.
65     IHost h = usock.registerHost("Kendiniz", "127.0.0.1", port);
66     if(!hostListModel.contains(h)) {
67         hostListModel.addElement(h);
68     }
69 }
70 public void setPort(int port) {
71     this.port = port;
72 }
73
74 @Action
75 public void showAboutBox() {
76     if (aboutBox == null) {
77         JFrame mainFrame = UMESENAApp.getApplication().getMainFrame();
78         aboutBox = new AboutBox(mainFrame);
79         aboutBox.setLocationRelativeTo(mainFrame);
80     }
81     UMESENAApp.getApplication().show(aboutBox);
82 }
83
84 @SuppressWarnings("unchecked")
85 // <editor-fold defaultstate="collapsed" desc="Generated Code">
86 private void initComponents() {
87
88     mainPanel = new javax.swing.JPanel();
89     jSafString = new javax.swing.JButton();
90     jScrollPane1 = new javax.swing.JScrollPane();
91     jHostList = new javax.swing.JList();
92     jLabel1 = new javax.swing.JLabel();
```

```

93     jHostIP = new javax.swing.JTextField();
94     jLabel3 = new javax.swing.JLabel();
95     jHostPort = new javax.swing.JTextField();
96     jHostEkle = new javax.swing.JButton();
97     jLabel2 = new javax.swing.JLabel();
98     jScrollPane2 = new javax.swing.JScrollPane();
99     jList2 = new javax.swing.JList();
100    jLabel4 = new javax.swing.JLabel();
101    jText = new javax.swing.JTextField();
102    jStringDP = new javax.swing.JButton();
103    jDosyaYolla = new javax.swing.JButton();
104    menuBar = new javax.swing.JMenuBar();
105    javax.swing.JMenu fileMenu = new javax.swing.JMenu();
106    javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
107    javax.swing.JMenu helpMenu = new javax.swing.JMenu();
108    javax.swing.JMenuItem aboutMenuItem = new javax.swing.JMenuItem();
109    jDialog1 = new javax.swing.JDialog();
110    jDialog2 = new javax.swing.JDialog();
111    jDialog3 = new javax.swing.JDialog();
112
113    mainPanel.setMaximumSize(new java.awt.Dimension(1024, 768));
114    mainPanel.setMinimumSize(new java.awt.Dimension(640, 480));
115    mainPanel.setName("mainPanel"); // NOI18N
116    mainPanel.setPreferredSize(new java.awt.Dimension(640, 480));
117
118    org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(umesen.UMESENApp.class).getContex
t().getResourceMap(MainWindow.class);
119    jSafString.setText(resourceMap.getString("jSafString.text")); // NOI18N
120    jSafString.setName("jSafString"); // NOI18N
121    jSafString.addActionListener(new java.awt.event.ActionListener() {
122        public void actionPerformed(java.awt.event.ActionEvent evt) {
123            jSafStringActionPerformed(evt);
124        }
125    });
126
127    jScrollPane1.setName("jScrollPane1"); // NOI18N
128
129    jHostList.setModel(hostListModel);
130    jHostList.setName("jHostList"); // NOI18N
131    jScrollPane1.setViewportView(jHostList);
132
133    jLabel1.setText(resourceMap.getString("jLabel1.text")); // NOI18N
134    jLabel1.setName("jLabel1"); // NOI18N
135
136    jHostIP.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
137    jHostIP.setText(resourceMap.getString("jHostIP.text")); // NOI18N
138    jHostIP.setName("jHostIP"); // NOI18N
139
140    jLabel3.setText(resourceMap.getString("jLabel3.text")); // NOI18N
141    jLabel3.setName("jLabel3"); // NOI18N
142
143    jHostPort.setHorizontalAlignment(javax.swing.JTextField.CENTER);
144    jHostPort.setText(resourceMap.getString("jHostPort.text")); // NOI18N
145    jHostPort.setName("jHostPort"); // NOI18N
146
147    jHostEkle.setText(resourceMap.getString("jHostEkle.text")); // NOI18N

```

```

148     jHostEkle.setName("jHostEkle"); // NOI18N
149     jHostEkle.addActionListener(new java.awt.event.ActionListener() {
150         public void actionPerformed(java.awt.event.ActionEvent evt) {
151             jHostEkleActionPerformed(evt);
152         }
153     });
154
155     jLabel2.setText(resourceMap.getString("jLabel2.text")); // NOI18N
156     jLabel2.setName("jLabel2"); // NOI18N
157
158     jScrollPane2.setName("jScrollPane2"); // NOI18N
159
160     jList2.setModel(mesajListModel);
161     jList2.setName("jList2"); // NOI18N
162     jScrollPane2.setViewportView(jList2);
163
164     jLabel4.setText(resourceMap.getString("jLabel4.text")); // NOI18N
165     jLabel4.setName("jLabel4"); // NOI18N
166
167     jText.setText(resourceMap.getString("jText.text")); // NOI18N
168     jText.setName("jText"); // NOI18N
169
170     jStringDP.setText(resourceMap.getString("jStringDP.text")); // NOI18N
171     jStringDP.setName("jStringDP"); // NOI18N
172     jStringDP.addActionListener(new java.awt.event.ActionListener() {
173         public void actionPerformed(java.awt.event.ActionEvent evt) {
174             jStringDPActionPerformed(evt);
175         }
176     });
177
178     jDosyaYolla.setText(resourceMap.getString("jDosyaYolla.text")); //
NOI18N
179     jDosyaYolla.setName("jDosyaYolla"); // NOI18N
180     jDosyaYolla.addActionListener(new java.awt.event.ActionListener() {
181         public void actionPerformed(java.awt.event.ActionEvent evt) {
182             jDosyaYollaActionPerformed(evt);
183         }
184     });
185
186     javax.swing.GroupLayout mainPanelLayout = new ja-
vax.swing.GroupLayout(mainPanel);
187     mainPanel.setLayout(mainPanelLayout);
188     mainPanelLayout.setHorizontalGroup(
189         mainPanelLa-
yout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190         .addGroup(mainPanelLayout.createSequentialGroup())
191         .addContainerGap()
192         .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
193         .addComponent(jScrollPane1, ja-
vax.swing.GroupLayout.DEFAULT_SIZE, 249, Short.MAX_VALUE)
194         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, mainPa-
nellayout.createSequentialGroup())
195         .addComponent(jHostIP, javax.swing.GroupLayout.DEFAULT_SIZE,
108, Short.MAX_VALUE)
196

```

```

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
197         .addComponent(jLabel13, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 4, javax.swing.GroupLayout.PREFERRED_SIZE)
198
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
199         .addComponent(jHostPort, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
200
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
201         .addComponent(jHostEkle, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE))
202         .addComponent(jLabel11))
203         .addGap(10, 10, 10)
204
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
205         .addGroup(mainPanelLayout.createSequentialGroup()
206                 .addGap(10, 10, 10)
207
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
208                 .addComponent(jLabel14)
209                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
210                         .addComponent(jLabel12)
211                         .addComponent(jScrollPane2, ja-
vax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
364, Short.MAX_VALUE)
212                         .addComponent(jText))))
213                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, mainPa-
nellLayout.createSequentialGroup()
214                         .addComponent(jSafString, ja-
vax.swing.GroupLayout.PREFERRED_SIZE, 145, javax.swing.GroupLayout.PREFERRED_SIZE)
215
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
216
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
217                 .addComponent(jDosyaYolla, ja-
vax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
218                 .addComponent(jStringDP, ja-
vax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
145, Short.MAX_VALUE))))
219         .addContainerGap())
220     );
221     mainPanelLayout.setVerticalGroup(
222         mainPanelLa-
yout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
223         .addGroup(mainPanelLayout.createSequentialGroup()
224                 .addContainerGap()
225
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
226                 .addComponent(jLabel11)
227                 .addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE,

```

```

14, javax.swing.GroupLayout.PREFERRED_SIZE))
228
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
229
.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
230
    .addGroup(mainPanelLayout.createSequentialGroup()
231
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 237, javax.swing.GroupLayout.PREFERRED_SIZE)
232
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
233
        .addComponent(jLabel4)
234
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
235
        .addComponent(jText, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
236
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
237
        .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
238
            .addComponent(jStringDP)
239
            .addComponent(jSafString))
240
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
241
        .addComponent(jDosyaYolla))
242
        .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 391, Short.MAX_VALUE))
243
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
244
        .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
245
            .addComponent(jHostEkle, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
246
            .addComponent(jHostIP, javax.swing.GroupLayout.PREFERRED_SIZE, 22, javax.swing.GroupLayout.PREFERRED_SIZE)
247
            .addComponent(jHostPort, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
248
            .addComponent(jLabel13))
249
            .addContainerGap())
250
    );
251
    menuBar.setName("menuBar"); // NOI18N
252
    fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
253
    fileMenu.setName("fileMenu"); // NOI18N
254
    javax.swing.ActionMap actionMap =
255
    org.jdesktop.application.Application.getInstance(umesen.UMESENApp.class).getContext().getActionMap(MainWindow.class, this);
256
    exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
257
    exitMenuItem.setText(resourceMap.getString("exitMenuItem.text")); //
NOI18N
258
    exitMenuItem.setName("exitMenuItem"); // NOI18N
259
    fileMenu.add(exitMenuItem);
260
261
262

```

```

263     menuBar.add(fileMenu);
264
265     helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
266     helpMenu.setName("helpMenu"); // NOI18N
267
268     aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
269     aboutMenuItem.setText(resourceMap.getString("aboutMenuItem.text")); //
NOI18N
270     aboutMenuItem.setName("aboutMenuItem"); // NOI18N
271     helpMenu.add(aboutMenuItem);
272
273     menuBar.add(helpMenu);
274
275     jDialog1.setName("jDialog1"); // NOI18N
276
277     javax.swing.GroupLayout jDialog1Layout = new ja-
vax.swing.GroupLayout(jDialog1.getContentPane());
278     jDialog1.getContentPane().setLayout(jDialog1Layout);
279     jDialog1Layout.setHorizontalGroup(
280         jDia-
log1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
281         .addGap(0, 400, Short.MAX_VALUE)
282     );
283     jDialog1Layout.setVerticalGroup(
284         jDia-
log1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
285         .addGap(0, 300, Short.MAX_VALUE)
286     );
287
288     jDialog2.setName("jDialog2"); // NOI18N
289
290     javax.swing.GroupLayout jDialog2Layout = new ja-
vax.swing.GroupLayout(jDialog2.getContentPane());
291     jDialog2.getContentPane().setLayout(jDialog2Layout);
292     jDialog2Layout.setHorizontalGroup(
293         jDia-
log2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
294         .addGap(0, 400, Short.MAX_VALUE)
295     );
296     jDialog2Layout.setVerticalGroup(
297         jDia-
log2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
298         .addGap(0, 300, Short.MAX_VALUE)
299     );
300
301     jDialog3.setName("jDialog3"); // NOI18N
302
303     javax.swing.GroupLayout jDialog3Layout = new ja-
vax.swing.GroupLayout(jDialog3.getContentPane());
304     jDialog3.getContentPane().setLayout(jDialog3Layout);
305     jDialog3Layout.setHorizontalGroup(
306         jDia-
log3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
307         .addGap(0, 400, Short.MAX_VALUE)
308     );
309     jDialog3Layout.setVerticalGroup(
310         jDia-

```

```

log3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
311     .addGap(0, 300, Short.MAX_VALUE)
312     );
313
314     setComponent(mainPanel);
315     setMenuBar(menuBar);
316 }// </editor-fold>
317
318 /**
319  * Bu buton String tipinde bir veriyi listede seçili tüm hostlara yollar.
320  * Burada gösterilmek istenen Serializeable olan tüm JAVA Classlarının
doğrudan
321  * kullanılabileceği, kullanıcının yeni sınıf bile yazmaya ihtiyaç duyma-
yacağıdır.
322  *
323  * @param JAVA ile ilgili bir şeyler
324  */
325 private void jSafStringActionPerformed(java.awt.event.ActionEvent evt) {
326
327     // Seçili hostların listesini alalım.
328     int[] selectedIx = jHostList.getSelectedIndices();
329
330     // Bu hostlara teker teker gönderilecek veriyi yollayalım.
331     /* Gönderme işlemi threaded olduğu için bir diğeri için
332     * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
333     */
334     for (int i=0; i<selectedIx.length; i++) {
335         IHost sel = (IHost) jHost-
List.getModel().getElementAt(selectedIx[i]);
336         sel.sendMessage(jText.getText());
337     }
338 }
339
340 /**
341  * Sisteme ve listeye yeni bir Host ekler.
342  *
343  * @param JAVA ile ilgili bir şeyler
344  */
345 private void jHostEkleActionPerformed(java.awt.event.ActionEvent evt) {
346     String portS = jHostPort.getText();
347     int portGelen = 0;
348     try {
349         portGelen = Integer.parseInt(portS);
350     } catch (NumberFormatException numberFormatException) {
351         System.out.println("Port sayı değil.");
352     }
353     IHost h = usock.registerHost(jHostIP.getText(), portGelen);
354     if(!hostListModel.contains(h)) {
355         hostListModel.addElement(h);
356     }
357 }
358
359 /**
360  * Bu buton StringDataPackage tipinde bir veriyi listede seçili tüm host-
lara yollar.
361  * Burada gösterilmek istenen Serializeable olduktan sonra kullanıcının
kendi

```

```

362     * veri yapısını kolaylıkla oluşturabileceğidir.
363     *
364     * @param JAVA ile ilgili bir şeyler
365     */
366     private void jStringDPActionPerformed(java.awt.event.ActionEvent evt) {
367         // Seçili hostların listesini alalım.
368         int[] selectedIx = jHostList.getSelectedIndices();
369
370         // Bu hostlara teker teker gönderilecek veriyi yollayalım.
371         /* Gönderme işlemi threaded olduğu için bir diğeri için
372         * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
373         */
374         for (int i=0; i<selectedIx.length; i++) {
375             IHost sel = (IHost) jHostList.getModel().getElementAt(selectedIx[i]);
376             sel.sendMessage(new StringDataPackage(jText.getText(),this.nickname));
377         }
378     }
379
380     /**
381     * Bu buton FileDataPackage tipinde bir veriyi listede seçili tüm hostlara
382     * yollar.
383     * Burada gösterilmek istenen dosyaların dahi kolaylıkla gönderilebileceğidir.
384     * Program göndermeden önce dosyayı okumakta ve bunu byte dizisi olarak belleğe alıp
385     * nesneye yazmaktadır. (haliyle)
386     *
387     * @param JAVA ile ilgili bir şeyler
388     */
389     private void jDosyaYollaActionPerformed(java.awt.event.ActionEvent evt) {
390
391         JFileChooser fc = new JFileChooser();
392
393         // Dosya açma dialogunu göster
394         fc.showOpenDialog(UMESENApp.getApplication().getMainFrame());
395         File selFile = fc.getSelectedFile();
396         if (selFile == null) return; // Cancel
397
398         // Seçili hostların listesini alalım.
399         int[] selectedIx = jHostList.getSelectedIndices();
400
401         // Bu hostlara teker teker gönderilecek veriyi yollayalım.
402         /* Gönderme işlemi threaded olduğu için bir diğeri için
403         * ilkinin bekleme söz konusu değildir. (adamlar yapmış)
404         */
405         for (int i=0; i<selectedIx.length; i++) {
406             IHost sel = (IHost) jHostList.getModel().getElementAt(selectedIx[i]);
407
408             byte [] fileByte = new byte [(int)selFile.length()];
409             /* Dosyayı okuyoruz */
410             FileInputStream fis;
411             try {
412                 fis = new FileInputStream(selFile);
413                 fis.read(fileByte,0,fileByte.length);

```

```

414         FileDataPackage fp = new FileDataPackage(selFile.getName(), fi-
leByte, this.nickname);
415         sel.sendMessage(fp);
416     } catch (FileNotFoundException ex) {
417         System.out.println("FileNotFoundException yedik: " +
ex.getMessage());
418     } catch (IOException ex) {
419         System.out.println("IOException yedik: " + ex.getMessage());
420     }
421 }
422 }
423
424 // Variables declaration - do not modify
425 private javax.swing.JDialog jDialog1;
426 private javax.swing.JDialog jDialog2;
427 private javax.swing.JDialog jDialog3;
428 private javax.swing.JButton jDosyaYolla;
429 private javax.swing.JButton jHostEkle;
430 private javax.swing.JTextField jHostIP;
431 private javax.swing.JList jHostList;
432 private javax.swing.JTextField jHostPort;
433 private javax.swing.JLabel jLabel1;
434 private javax.swing.JLabel jLabel2;
435 private javax.swing.JLabel jLabel3;
436 private javax.swing.JLabel jLabel4;
437 private javax.swing.JList jList2;
438 private javax.swing.JButton jSafString;
439 private javax.swing.JScrollPane jScrollPane1;
440 private javax.swing.JScrollPane jScrollPane2;
441 private javax.swing.JButton jStringDP;
442 private javax.swing.JTextField jText;
443 private javax.swing.JPanel mainPanel;
444 private javax.swing.JMenuBar menuBar;
445 // End of variables declaration
446
447 private JDialog aboutBox;
448 private JDialog portSelectionBox;
449 }
450
451

```

PortAndNickSelectionBox.java

```

1 package umesen;
2
3 /**
4  * Program başlamadan önce bağlantı kurulacak portu ve
5  * kullanıcı bilgilerini alan formdur.
6  *
7  * @author UB
8  */
9 public class PortAndNickSelectionBox extends javax.swing.JDialog {
10     private MainWindow u;
11
12     public PortAndNickSelectionBox(java.awt.Frame parent, MainWindow u) {
13         super(parent, true);
14         this.u = u;

```

```

15     u.setPort(-1);
16     initComponents();
17 }
18
19 @SuppressWarnings("unchecked")
20 // <editor-fold defaultstate="collapsed" desc="Generated Code">
21 private void initComponents() {
22
23     jOK = new javax.swing.JButton();
24     jLabel1 = new javax.swing.JLabel();
25     jPortNumber = new javax.swing.JTextField();
26     jLabel2 = new javax.swing.JLabel();
27     jComp = new javax.swing.JTextField();
28
29     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
30     org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(umesen.UMESENApp.class).getContext().getResourceMap(PortAndNickSelectionBox.class);
31     setTitle(resourceMap.getString("Form.title")); // NOI18N
32     setName("Form"); // NOI18N
33     setResizable(false);
34
35     jOK.setText(resourceMap.getString("jOK.text")); // NOI18N
36     jOK.setName("jOK"); // NOI18N
37     jOK.addActionListener(new java.awt.event.ActionListener() {
38         public void actionPerformed(java.awt.event.ActionEvent evt) {
39             jOKActionPerformed(evt);
40         }
41     });
42
43     jLabel1.setText(resourceMap.getString("jLabel1.text")); // NOI18N
44     jLabel1.setName("jLabel1"); // NOI18N
45
46     jPortNumber.setHorizontalAlignment(javax.swing.JTextField.LEFT);
47     jPortNumber.setText(resourceMap.getString("jPortNumber.text")); //
NOI18N
48     jPortNumber.setName("jPortNumber"); // NOI18N
49
50     jLabel2.setText(resourceMap.getString("jLabel2.text")); // NOI18N
51     jLabel2.setName("jLabel2"); // NOI18N
52
53     jComp.setHorizontalAlignment(javax.swing.JTextField.LEFT);
54     jComp.setText(resourceMap.getString("jComp.text")); // NOI18N
55     jComp.setName("jComp"); // NOI18N
56
57     javax.swing.GroupLayout layout = new ja-
vax.swing.GroupLayout(getContentPane());
58     getContentPane().setLayout(layout);
59     layout.setHorizontalGroup(
60         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
61         .addGroup(layout.createSequentialGroup()
62             .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
63                 .add(layout.createSequentialGroup()
64                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
65                         .add(layout.createSequentialGroup()
66

```

```

67 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
68     .addComponent(jComp, javax.swing.GroupLayout.DEFAULT_SIZE,
131, Short.MAX_VALUE))
69     .addGroup(layout.createSequentialGroup())
70     .addComponent(jLabel1)
71
72 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
73     .addComponent(jPortNumber, javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE))
74     .addComponent(jOK, javax.swing.GroupLayout.Alignment.TRAILING))
75     .addContainerGap())
76 );
77 layout.setVerticalGroup(
78     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
79     .addGroup(layout.createSequentialGroup())
80     .addContainerGap())
81 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
82     .addComponent(jLabel1)
83     .addComponent(jPortNumber, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
84 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
85 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
86     .addComponent(jLabel2)
87     .addComponent(jComp, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
88 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
89     .addComponent(jOK)
90     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
91 );
92 pack();
93 }// </editor-fold>
94
95 private void jOKActionPerformed(java.awt.event.ActionEvent evt) {
96     try {
97         String pS = jPortNumber.getText();
98         int p = Integer.parseInt(pS);
99         u.setPort(p);
100        u.nickname = jComp.getText();
101        dispose();
102    } catch (NumberFormatException numberFormatException) {
103    }
104 }
105
106 // Variables declaration - do not modify
107 private javax.swing.JTextField jComp;
108 private javax.swing.JLabel jLabel1;
109 private javax.swing.JLabel jLabel2;
110 private javax.swing.JButton jOK;
111 private javax.swing.JTextField jPortNumber;

```

```
112 // End of variables declaration
113
114 }
115
116
```

UMESENApp.java

```
1 package umesen;
2
3 import java.awt.event.WindowAdapter;
4 import java.awt.event.WindowEvent;
5 import org.jdesktop.application.Application;
6 import org.jdesktop.application.SingleFrameApplication;
7
8 /**
9  * KULLANICI tarafının çalışan sınıfıdır.
10 * Main burada bulunmaktadır.
11 *
12 * @author UB
13 */
14 public class UMESENApp extends SingleFrameApplication {
15
16     @Override protected void startup() {
17         show(new MainWindow(this));
18     }
19     @Override
20     protected void configureWindow(java.awt.Window root) {
21
22         root.addWindowListener(new WindowAdapter() {
23
24             @Override
25             public void windowClosing(WindowEvent e) {
26                 System.exit(0);
27             }
28
29         });
30     }
31
32     public static UMESENApp getApplication() {
33         return Application.getInstance(UMESENApp.class);
34     }
35
36     public static void main(String[] args) {
37         launch(UMESENApp.class, args);
38     }
39 }
40
```

FileDataPackage.java

```
1 package umesen.datapackages;
2
3 import java.io.Serializable;
4
5 /**
6  * İçinde bir bayt dizisi ve gönderen kişinin "nickname" bilgisini tutan
```

```

7  * KULLANICI programı içerisinde yer alan örnek bir veri paketidir.
8  *
9  * Bu byte dizisinin bir dosya olması öngörülmektedir.
10 *
11 * @author UB
12 */
13 public class FileDataPackage implements Serializable {
14     private String filename;
15     private String sender;
16     private byte[] file;
17
18     public FileDataPackage (String filename, byte[] file, String sender) {
19         this.file = file;
20         this.filename = filename;
21         this.sender = sender;
22     }
23     public String getSender() {
24         return sender;
25     }
26     public String getFilename() {
27         return filename;
28     }
29     public byte[] getFile() {
30         return file;
31     }
32
33 }
34
35

```

StringDataPackage.java

```

1  package umesen.datapackages;
2
3  import java.io.Serializable;
4
5  /**
6   * İçinde metin ve metnin gönderen kişinin "nickname" bilgisini tutan
7   * KULLANICI programı içerisinde yer alan örnek bir veri paketidir.
8   *
9   * @author UB
10 */
11 public class StringDataPackage implements Serializable {
12     private String metin;
13     private String nickname;
14
15     public StringDataPackage (String s, String nickname) {
16         this.metin = s;
17         this.nickname = nickname;
18     }
19
20     public String getString() {
21         return metin;
22     }
23
24     public String getSender() {
25         return nickname;

```

```
26 }
27 }
28
```

DefaultADP.java

```
1 package umesen.dataprocessors;
2
3 import java.io.FileNotFoundException;
4 import java.io.IOException;
5 import umesen.datapackages.StringDataPackage;
6 import com.ubenzer.usock.interfaces.ArrivedDataProcessor;
7 import java.io.File;
8 import java.io.FileOutputStream;
9 import javax.swing.DefaultListModel;
10 import javax.swing.JFileChooser;
11 import umesen.UMESENApp;
12 import umesen.datapackages.FileDataPackage;
13
14 /**
15  * Uzak makineden gelen tüm verileri ArrivedDataProcessorlara
16  * gönderilir.
17  *
18  * Bu KULLANICI uygulaması yapılandırmasına göre gelen verileri bu
19  * sınır almaktadır. Bir sınıf ArrivedDataProcessor arayüzünü
20  * implement ettiği sürece gelen verileri isteyen tüm sınıflar
21  * process edebilir.
22  *
23  * @author UB
24  */
25 public class DefaultADP implements ArrivedDataProcessor {
26     DefaultListModel writeList;
27
28     /**
29      * Gelen verilerin ekranda yazdırılacağı nesneyi parametre olarak alır.
30      * USock ile bir alakası olmayıp tamamen KULLANICI uygulaması ile alakalı-
31      * dır.
32      *
33      * @param Ekranda verilerin yazılacağı nesne
34      */
35     public DefaultADP(DefaultListModel writeList) {
36         this.writeList = writeList;
37     }
38     /**
39      * Bu kısım ArrivedDataProcessor implementasyonudur.
40      *
41      * Gelen nesnenin türü incelenir ve gerekenler yapılır.
42      *
43      * @param Uzak makineden gelen veri
44      * @param Uzak makinenin adresi
45      */
46     public void processArrivedData(Object dataReceived, String sender) {
47         System.out.println("THREAD " + Thread.currentThread().getName() + " // "
48 + "Gelen nesnenin türü: " + dataReceived.getClass().toString());
49         if(dataReceived instanceof String) {
50             writeList.addElement(sender + " // " + (String)dataReceived);
51         } else if (dataReceived instanceof StringDataPackage) {
52             writeList.addElement(sender + " // " + ((StringDataPacka-
```

```

ge)dataReceived).getSender() + " // " + ((StringDataPacka-
ge)dataReceived).getString());
51     } else if (dataReceived instanceof FileDataPackage) {
52         FileDataPackage f = (FileDataPackage)dataReceived;
53
54         writelist.addElement(sender + " // " + f.getSender() + " isimli kulla-
nıcı dosya gönderdi: " + f.getFilename());
55
56         /*
57          * Dosyanın kaydedileceği yeri soruyoruz ve kullanıcının istediği yere
58          * dosyayı yazıyoruz.
59          */
60         JFileChooser fc = new JFileChooser();
61
62         fc.showSaveDialog(UMESENApp.getApplication().getMainFrame());
63         File selFile = fc.getSelectedFile();
64         if (selFile == null) {
65             writelist.addElement(sender + " // " + f.getSender() + " isimli
kullanıcı dosya kaydetme iptal edildi.");
66             return;
67         }
68
69         FileOutputStream fos;
70         try {
71             fos = new FileOutputStream(selFile);
72             fos.write(f.getFile());
73             fos.close();
74         } catch (FileNotFoundException ex) {
75             System.out.println("THREAD " + Thread.currentThread().getName() + "
// " + "FileNotFoundException yedik: " +
76                 ex.getMessage());
77         } catch (IOException ex) {
78             System.out.println("THREAD " + Thread.currentThread().getName() + "
// " + "IOException yedik: " +
79                 ex.getMessage());
80         }
81     } else {
82         writelist.addElement(sender + " // DefaultADP'nin anlamadığı bir paket
geldi. Ignore ettim.");
83     }
84 }
85
86 }
87
88

```

USOCK

Host.java

```

1 package com.ubenzer.usock.classes;
2
3
4 import com.ubenzer.usock.interfaces.IHost;
5 import java.io.Serializable;
6

```

```

7 /**
8  * Kendisine veri yollanabilecek bir HOST.
9  *
10 * @author UB
11 */
12 public class Host implements IHost {
13     private String hostName;
14     private String hostAddr;
15     private int hostPort;
16
17     /**
18     * Kendisine veri yollanabilecek bir Host yaratır.
19     *
20     * @param Host adı, tamamen görsel amaçlı
21     * @param Host adresi (IP)
22     * @param Hostun dinlemekte olduğu port (varsayılan 40'tır.)
23     */
24     public Host(String hostName, String hostAddr, int port) {
25         this.hostAddr = hostAddr;
26         this.hostName = hostName;
27         this.hostPort = port;
28     }
29     /**
30     * Kendisine veri yollanabilecek bir Host yaratır.
31     *
32     * @param Host adresi (IP)
33     * @param Hostun dinlemekte olduğu port (varsayılan 40'tır.)
34     */
35     public Host(String hostAddr, int port) {
36         this("",hostAddr,port);
37     }
38     /**
39     * Host adını döndürür.
40     * @return Hostun adı
41     */
42     public String getHostName() {
43         return this.hostName;
44     }
45     /**
46     * Hostun dinlediği port numarasını döndürür.
47     *
48     * @return Hostun portu
49     */
50     public int getHostPort() {
51         return this.hostPort;
52     }
53     /**
54     * Hostun ağdaki adresini döndürür.
55     *
56     * @return Hostun adresi
57     */
58     public String getHostAddr() {
59         return this.hostAddr;
60     }
61
62     /**
63     * Hosta bir mesaj yollar. Bu mesaj Serializable olduktan sonra

```

```

64     * her şey olabilir. Doğru bir şekilde Serialize edildikten sonra
65     * ister String, ister 300MB'lık divX yollanabilir.
66     *
67     * @param Yollanan veri
68     */
69     public void sendMessage(Serializable msg) {
70         System.out.println(hostAddr + " adresine mesaj yollamak için thread
yaratılıyor...");
71         OutgoingMessageProcessor mp = new OutgoingMessageProcessor(this,msg);
72         new Thread(mp,"OUTGOING MESSAGE PROCESSOR " + mp.hashCode()).start();
73     }
74
75     @Override
76     public String toString() {
77         if (!this.hostName.equals("")) {
78             return this.hostName;
79         } else {
80             return "Host " + this.hostAddr + ":" + this.hostPort;
81         }
82     }
83
84     @Override
85     public boolean equals(Object o) {
86         if(o.getClass() != this.getClass()) return false;
87         IHost h;
88         try {
89             h = (IHost) o;
90         } catch (Exception e) {
91             return false;
92         }
93         if(h.getHostAddr().equals(this.hostAddr) && h.getHostPort() ==
this.hostPort) {
94             return true;
95         }
96         return false;
97     }
98     @Override
99     public int hashCode() {
100        int hash = 3;
101        hash = 17 * hash + (this.hostAddr != null ? this.hostAddr.hashCode() :
0);
102        hash = 17 * hash + this.hostPort;
103        return hash;
104    }
105

```

IncomingMessagePasser.java

```

1 package com.ubenzer.usock.classes;
2
3 import com.ubenzer.usock.debug.Debug;
4 import com.ubenzer.usock.interfaces.ArrivedDataProcessor;
5
6 /**
7  * Başarı ile bizim bilgisayarımıza gelmiş olan
8  * bir verinin ne olduğunun anlaşılması ve bu veriyi
9  * işleyecek olan KULLANICI sınıfına geçirecek olan threaddır.
10  */

```

```

11 * Bu sınıf bir threaddir çünkü:
12 *
13 * Veriyi alan kullanıcı sınıfı bu veriyle çok vakit geçirebilir, deyim yerin-
deyse
14 * turşusunu kurabilir. Bu yüzden gelen verilerin paralel işlenmesini sağlamak
15 * adına bunlar threadlere işletilir.
16 *
17 * @author UB
18 */
19 public class IncomingMessagePasser implements Runnable {
20     private ArrivedDataProcessor ADP;
21     private Object msg;
22     private String sender;
23
24     /**
25     * İşleyecek olan sınıf, gelen veri gibi bilgiler thread çalıştırılmadan
26     * önce kendisine yüklenmelidir.
27     *
28     * @param ArrivedDataProcessor arayüzünü implement eden, gelen verilerin
29     * yollanacağı KULLANICI sınıfı.
30     * @param Nesne halinde gelen mesaj
31     * @param Göndericinin adresi
32     */
33     protected IncomingMessagePasser(ArrivedDataProcessor ADP, Object msg, String
sender) {
34         this.ADP = ADP;
35         this.msg = msg;
36         this.sender = sender;
37     }
38     /**
39     * Veriyi alıp işleyecek KULLANICI sınıfı çağrılır.
40     */
41     public void run() {
42         Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Başla-
dı.");
43         ADP.processArrivedData(msg, sender);
44         Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Bit-
ti.");
45     }
46 }

```

IncomingMessageProcessor.java

```

1 package com.ubenzer.usock.classes;
2
3 import com.ubenzer.usock.debug.Debug;
4 import java.io.IOException;
5 import java.io.ObjectInputStream;
6 import java.net.Socket;
7
8 /**
9 * Gelen isteklerin kabul edilip, verinin aktarılmasından sorumlu sınıftır.
Veriler
10 * bir defa aktarıldıktan sonr, ne olduklarının algılanıp gerekenlerin yapılma-
sı
11 * bu sürecin sorumluluğu altında yer almamaktadır.
12 *
13 * @author UB

```

```

14 */
15 public class IncomingMessageProcessor implements Runnable {
16     private Socket cs;
17     private USock us;
18
19     /**
20      * Verinin aktarılması için gerekli olan bu süreç çalışmadan önce
21      * constructor ile yapılandırılmalıdır.
22      *
23      * @param Verinin geleceği socket
24      * @param İşlemin bağlı olduğu USock nesnesi
25      */
26     protected IncomingMessageProcessor(Socket clientSocket, USock us) {
27         this.cs = clientSocket;
28         this.us = us;
29     }
30
31     /**
32      * Verinin uzak sunucudan aktarılması bu kısımda yapılır. Gelen verinin ne
33      * olduğunu
34      * anlamak ve gerekeni yapmak bu sürecin işi değildir.
35      *
36      * Gelen veri tamamen alındıktan sonra algılanması ve gerekenin yapılması
37      * için
38      * bununla ilgili bir süreç yaratılır.
39      */
40     public void run() {
41         Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Başla-
42         di.");
43         try {
44             /* Gelen veriyi al */
45             ObjectInputStream in = new ObjectInputStream(cs.getInputStream());
46             /* Gelen veri Serializable bir obje olduğu için alttaki satır çalışır.
47             */
48             Object o = (Object) in.readObject();
49             in.close();
50
51             /* Gelen veriyi işleyecek sınıf bu bilginin nereden geldiğine ihtiyaç
52             duyabilir. */
53             String incomingAddr = cs.getInetAddress().getHostAddress();
54
55             /* Gelen veriyi işleyecek süreç yaratılır. Aslında bu süreçte de bu
56             işlem yapılabilirdi,
57             ancak hem parça parça yazmanın kolay olması, hem de gelecekte birden
58             fazla sürecin gelen
59             veriyi aynı anda işleyebilme gereksinimi göz önüne alınarak bunlar
60             ayrı threadlere ayrıldı.
61             */
62             IncomingMessagePasser IMP = new IncomingMessagePas-
63             ser(us.getDefaultADP(),o,incomingAddr);
64             new Thread(IMP,"INCOMING MESSAGE PASSER " + IMP.hashCode()).start();
65
66         } catch (IOException ex) {
67             Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
68             "IOException yedik: " + ex.getMessage());
69         } catch (ClassNotFoundException ex) {
70             Debug.log("THREAD " + Thread.currentThread().getName() + " // " +

```

```

"ClassNotFoundException yedik: "
61         + ex.getMessage());
62     }
63     Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Bit-
ti.");
64 }
65 }
66
67

```

OutgoingMessageProcessor.java

```

1 package com.ubenzer.usock.classes;
2
3 import com.ubenzer.usock.debug.Debug;
4 import com.ubenzer.usock.interfaces.IHost;
5 import java.io.IOException;
6 import java.io.ObjectOutputStream;
7 import java.io.Serializable;
8 import java.net.Socket;
9 import java.net.UnknownHostException;
10
11 /**
12  * Uzaktaki bilgisayara dosya gönderilmesini sağlayan threaddir.
13  * Her gönderilecek dosya kendi sürecinde gönderilir. Böylece dosya
14  * gönderilemesinin diğer hiçbir işi aksatmaması hedeflenmektedir.
15  *
16  * @author UB
17  */
18 public class OutgoingMessageProcessor implements Runnable {
19     private IHost to;
20     private Serializable msg;
21
22     /**
23      * Thread çalıştırılmadan önce bu constructor vasıtasıyla yapılandırılır.
24      *
25      * @param Verinin gönderileceği adres
26      * @param Veri. Veri, JAVA Serializable interfaceini implement eden tüm
27      * sınıflar olabilir. Hiçbir kısıtımız yok. ;)
28      */
29     protected OutgoingMessageProcessor(IHost to, Serializable msg) {
30         this.msg = msg;
31         this.to = to;
32     }
33
34     /**
35      * Gönderilecek veriyi yollamayı sağlayan süreçtir.
36      * Her yollanan veri kendi süreci ile gider.
37      */
38     public void run() {
39         Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Başla-
di.");
40         Socket socket = null;
41         try {
42             socket = new Socket(to.getHostAddr(), to.getHostPort());
43             ObjectOutputStream oos = new ObjectOutputStream(
44                 socket.getOutputStream());
45             Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Ve-

```

```

ri yollanıyor...");
45     oos.writeObject(msg);
46     socket.close();
47     Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Ve-
ri yollandı. :)");
48     } catch (UnknownHostException e) {
49         Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
50             "UnknownHostException yedik. Host: " + to.getHostAddr() + " --
" + to.getHostName() + " -- "
51             + e.getMessage());
52     } catch (IOException e) {
53         Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
"IOException yedik: " + e.getMessage());
54     } catch (Exception e) {
55         Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
"Exception yedik: " + e.getMessage());
56     }
57     Debug.log("THREAD " + Thread.currentThread().getName() + " // " + "Bit-
ti.");
58 }
59
60 }
61
62

```

Server.java

```

1 package com.ubenzer.usock.classes;
2
3 import com.ubenzer.usock.debug.Debug;
4 import java.io.IOException;
5 import java.net.ServerSocket;
6 import java.net.Socket;
7
8 /**
9  * Gelen istekleri dinlemek ve gelen istekleri anlayıp
10 * bunların hepsini ayrı birer threadda alarak işlemekle
11 * mükellef bir sınıftır.
12 *
13 * @author UB
14 */
15 public class Server implements Runnable {
16     private USock us;
17     private int port = 40;
18     private ServerSocket serverSocket = null;
19
20     /**
21     * Sunucu sınıf. Gelen istekleri dinler.
22     *
23     * @param Dinlenecek port numarası
24     * @param Bağlı olduğu altyapı nesnesi
25     * @throws Port dinlenemiyor, çünkü başka uygulama dinliyor
26     */
27     protected Server(int portToBeListened, USock us) throws IOException {
28         this.port = portToBeListened;
29         this.us = us;
30         serverSocket = new ServerSocket(port);
31     }

```

```

32
33  /**
34   * Gelen istekleri dinleme threadi. Uygulama gelen istekleri dinlerken
35   * aynı zamanda arayüz gösterme, başka dosyalar alma ve dosya gönderme
36   * gibi şeyler de yapacağından dinleme işlemi süreç olarak çalışır.
37   *
38   * Burada dinleyen socket bir istek geldiği zaman verinin aktarılmasını
39   * ve alındıktan sonra anlaşılıp işlenmesini yürütmek üzere yeni bir süreç
40   * yaratıp anında dinlemeye devam eder.
41   *
42   * Böylece hiçbir zaman gelen veriler kaçırılmaz.
43   */
44  public void run() {
45      /* Gelen istekleri dinleyelim. */
46      Debug.log("THREAD " + Thread.currentThread().getName() + " // Başladı.");
47
48      /* Gelen istekleri ihmal etmeksizin dinleyelim. */
49      while (true) {
50          Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
51  this.port + " numaralı port gelen istekler"
52          + " için dinleniyor...");
53          /* Gelen isteği kabul et. */
54          Socket clientSocket;
55          try {
56              clientSocket = serverSocket.accept();
57              System.out.println("THREAD " + Thread.currentThread().getName() + "
58  // " +
59              clientSocket.getInetAddress() + " adresinden gelecek veri
60  var. Almak için thread yaratılıyor...");
61              IncomingMessageProcessor mp = new IncomingMessageProces-
62  sor(clientSocket,us);
63              new Thread(mp,"INCOMING MESSAGE PROCESSOR " +
64  mp.hashCode()).start();
65          } catch (IOException ex) {
66              Debug.log("THREAD " + Thread.currentThread().getName() + " // " +
67              "IOException yedik: " + ex.getMessage());
68          }
69      }
70  }
71  }
72  }
73  }
74  }
75  }
76  }
77  }
78  }
79  }
80  }
81  }
82  }
83  }
84  }
85  }
86  }
87  }
88  }
89  }
90  }
91  }
92  }
93  }
94  }
95  }
96  }
97  }
98  }
99  }
100 }

```

USock.java

```

1  package com.ubenzer.usock.classes;
2
3  import com.ubenzer.usock.debug.Debug;
4  import com.ubenzer.usock.interfaces.ArrivedDataProcessor;
5  import com.ubenzer.usock.interfaces.IHost;
6  import java.io.IOException;
7  import java.util.ArrayList;
8
9  /**
10   * Kullanıcının kendi uygulamasında USock alt yapısını
11   * kullanmak için ilk başvuracağı ana classtır.
12   *
13   * Aynı proje birden çok USock nesnesini aynı anda kullanabilir (ama

```

```

14 * buna gerek yoktur.)
15 *
16 * USock, çok rahat bir şekilde yeni veri tipleri yollamayı sağlar,
17 * bu konuda tek sınır JAVA'nın kendisidir. :)
18 *
19 * @author UB
20 */
21 public class USock {
22     private Server server = null;
23     private Thread serverThread = null;
24     private ArrayList<IHost> hostList = new ArrayList<IHost>();
25     private ArrivedDataProcessor ADP;
26
27     /**
28      * Yeni bir USock iletişim altyapısı yaratılır. Bu yapı iki yönlüdür.
29      * Her USock nesnesi, gelen istekleri alabilmek adına bir port dinleme ihtiyacı
30      * duyar. ArrivedDataProcessor ise, bu USock nesnesine uzaktan gelen isteklerin
31      * hangi kullanıcı sınıfına aktarılacağını belirtir.
32      *
33      * KULLANICI: USock altyapısını kullanan program.
34      * USOCK: Bizim altyapımız.
35      *
36      * @param Gelen istekler için dinlenecek olan port numarası
37      * @param Gelen verilerin hangi kullanıcı classına devredileceği
38      * @throws Hatalı verileri sevmeyiz.
39      */
40     public USock(int portToBeListened, ArrivedDataProcessor ADP) throws Exception {
41         if(portToBeListened < 0) throw new Exception("Port 0'dan küçük olamaz.");
42         if(portToBeListened > 65535) throw new Exception("Port 65535'ten büyük olamaz.");
43
44         this.ADP = ADP;
45
46         this.startServer(portToBeListened);
47     }
48
49     /**
50      * USock altyapısının kendisine ulaşan mesajları
51      * işlemesi adına geçireceği kullanıcı nesnesinin referansını
52      * döndürür.
53      *
54      * @return Kullanıcı Nesnesi
55      */
56     public ArrivedDataProcessor getDefaultADP() {
57         return this.ADP;
58     }
59     /**
60     * USock altyapısı, gelen istekleri dinleme hazırlıkları yapar. Buna port dinlemek
61     * dahildir.
62     *
63     * @param Dinlenecek Port
64     * @throws Port başka uygulamada, bu yüzden dinlenemiyor hatası

```

```

65     */
66     private void startServer(int portToBeListened) throws IOException {
67         Debug.log("Gelen isteklerin dinlenmesi için port " + portToBeListened +
" açılıyor...");
68         try {
69             server = new Server(portToBeListened, this);
70         } catch (IOException iOException) {
71             Debug.log("Port dinlemeye çalışırken IO Exception oluştu.");
72             throw iOException;
73         }
74         Debug.log("Port açılma işlemi tamamlandı, şimdi gelen istekleri dinleye-
cek bir thread başlatılıyor.");
75         serverThread = new Thread(server, "Server Thread " + server.hashCode());
76         serverThread.start();
77     }
78
79     /**
80     * USock altyapısı, daha sonra hızlıca erişmek için Host'ları kendi bünye-
sine
81     * kaydetmeye izin verir. "Host" uzaktaki bir bilgisayarın adı, dinlediği
port numarası
82     * ve adresidir.
83     *
84     * Böylece kullanıcı programı bir defa Host nesnesi yaratıp bunu bizim sis-
temimize register
85     * edince, bu Hosta ait adres ve port gibi bilgileri ayrıca tutmasına gerek
kalmayacak,
86     * bunlar tamamıyla USOCK tarafından yönetilecektir.
87     *
88     * Bu fonksiyon USock bünyesine kayıtlı hostlar arasında adres ve port bil-
gilerine göre
89     * arama yapar, bu bilgilere sahip host daha önce kaydedilmişse bu nesneyi
geri döndürür.
90     *
91     * Eğer bilgileri verilen host kayıtlı değilse geriye null döner.
92     *
93     * @param Aranan hostun adresi
94     * @param Aranan hostun dinlediği port
95     * @return IHost bulunan host veya null
96     */
97     public IHost searchHost(String hostAddr, int port) {
98         for(IHost h: hostList) {
99             if(h.getHostAddr().equalsIgnoreCase(hostAddr) && h.getHostPort() ==
port) {
100                 return h;
101             }
102         }
103         return null;
104     }
105     /**
106     * USock altyapısına daha sonra hızlıca erişmek için yeni bir host kaydet.
107     *
108     * @param Hostun adı (tamamen görsel amaçlı)
109     * @param Hostun adresi (IP)
110     * @param Hostun portu (Uzak makine hangi portu dinliyor?)
111     * @return Oluşturulan ve register edilen IHost nesnesi.
112     */

```

```

113 public IHost registerHost(String hostName, String hostAddress, int port) {
114     IHost host = new Host(hostName,hostAddress,port);
115     for(IHost h:hostList) {
116         if(h.equals(host)) return h;
117     }
118     hostList.add(host);
119     return host;
120 }
121 /**
122  * USock altyapısına daha sonra hızlıca erişmek için yeni bir host kaydet.
123  *
124  * @param Hostun adresi (IP)
125  * @param Hostun portu (Uzak makine hangi portu dinliyor?)
126  * @return Oluşturulan ve register edilen IHost nesnesi.
127  */
128 public IHost registerHost(String hostAddress, int port) {
129     IHost host = new Host(hostAddress,port);
130     for(IHost h:hostList) {
131         if(h.equals(host)) return h;
132     }
133     hostList.add(host);
134     return host;
135 }
136 /**
137  * USock altyapısından, daha önce register edilmiş bir hostu siler.
138  *
139  * @param Silinecek olan host
140  * @return Silidiyse true, zaten sisteme kayıtlı değilse false
141  */
142 public boolean unregisterHost(IHost hostToBeDeleted) {
143     return hostList.remove(hostToBeDeleted);
144 }
145 }
146

```

Debug.java

```

1 package com.ubenzer.usock.debug;
2
3 /**
4  * Hata ayıklamakta ve bilgi almakta kullanılmak amacıyla
5  * kritik bilgilerin kendisine gönderildiği sınıftır.
6  *
7  * @author UB
8  */
9 public class Debug {
10
11     /**
12     * Gelen bilgiyi ekrana yazar ve bir satır aşağı iner.
13     *
14     * @param Önemli bilgi
15     */
16     public static void log(String string) {
17         log(string,true);
18     }
19     /**
20     * Gelen bilgiyi ekrana yazar.
21     *

```

```

22     * @param Önemli bilgi
23     * @param true ise satır atlar, false ise atlamaz.
24     */
25     public static void log(String string, boolean satirAtla) {
26         if(satirAtla) {
27             System.out.println(string);
28         } else {
29             System.out.print(string);
30         }
31     }
32 }
33
34

```

ArrivedDataProcessor.java

```

1 package com.ubenzer.usock.interfaces;
2
3 /**
4  * Gelen verileri işleyecek KULLANICI sınıfı mutlaka bu
5  * arayüzü implement etmelidir.
6  *
7  * Zaten implement etmesi çok karışık bir şey de değildir. :)
8  *
9  * @author UB
10 */
11 public interface ArrivedDataProcessor {
12
13     /**
14     * Gelen veriyi işler.
15     *
16     * @param Nesne halinde gelen veri
17     * @param Yollayanın adresi
18     */
19     public void processArrivedData(Object dataReceived,String sender);
20
21 }
22
23

```

IHost.java

```

1 package com.ubenzer.usock.interfaces;
2
3 import java.io.Serializable;
4
5 /**
6  * USock'a bağımlı kalmadan Host yaratabilmek için bir arayüz.
7  * USock aracılığı ile veri gönderilecek tüm makinelerin nesneleri
8  * mutlaka IHost'u implement etmelidir.
9  *
10 * @author UB
11 */
12 public interface IHost {
13
14     /**
15     * Hostun adını döndürür.
16     * @return hostAdı

```

```
17     */
18     public String getHostAddr();
19
20     /**
21     * Hostun adresini döndürür.
22     * @return hostAdresi
23     */
24     public String getHostName();
25
26     /**
27     * Hostla bağlantı kurulan port numarasını döndürür.
28     *
29     * @return portNo
30     */
31     public int getHostPort();
32
33     /**
34     * Hosta bir mesaj yollar.
35     *
36     * @param Herhangi bir Serializable sınıf.
37     */
38     public void sendMessage(Serializable msg);
39
40 }
```