```
                System.out.println("адрес недоступен");
                e.printStackTrace();
            } catch (IOException e) {
                System.out.println("ошибка I/O потока");
                e.printStackTrace();
            } catch (InterruptedException e) {
                System.out.println(
                            "ошибка потока выполнения");
                e.printStackTrace();
            }
        }
}
```

Сервер должен быть инициализирован до того, как клиент попытается осуществить сокетное соединение. При этом может быть использован IP-адрес локального компьютера.

Датаграммы и протокол UDP

UDP (User Datagram Protocol) не устанавливает виртуального соединения и не гарантирует доставку данных. Отправитель просто посылает пакеты по указанному адресу; если отосланная информация была повреждена или вообще не дошла, отправитель об этом даже не узнает. Однако достоинством UDP является высокая скорость передачи данных. Данный протокол часто используется при трансляции аудио- и видеосигналов, где потеря небольшого количества данных не может привести к серьезным искажениям всей информации.

По протоколу UDP данные передаются пакетами. Пакетом в этом случае UDP является объект класса **DatagramPacket**. Этот класс содержит в себе передаваемые данные, представленные в виде массива байт. Конструкторы класса:

```
DatagramPacket(byte[] buf, int length)
DatagramPacket(byte[] buf, int length,
                  InetAddress address, int port)
DatagramPacket(byte[] buf, int offset, int length)
DatagramPacket(byte[] buf, int offset, int length,
                   InetAddress address, int port)
DatagramPacket(byte[] buf, int offset, int length,
                   SocketAddress address)
DatagramPacket(byte[] buf, int length,
                   SocketAddress address)
```

Первый конструктор используется в тех случаях когда датаграмма только принимает в себя пришедшие данные, так как созданный с его помощью объект не имеет информации об адресе и порте получателя. Остальные конструкторы используются для отправки датаграм.

Класс **DatagramSocket** может выступать в роли клиента и сервера, то есть он способен получать и отправлять пакеты. Отправить пакет можно с помощью метода **send(DatagramPacket pac)**, для получения пакета используется метод **receive(DatagramPacket pac)**.

```
/* пример # 9 : отправка файла по UDP протоколу : Sender.java */
package chapt15;
```

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

public class Sender {
      public static void main(String[] args) {
            try {
                  byte[] data = new byte[1000];
                  DatagramSocket s = new DatagramSocket();
                  InetAddress addr =
                        InetAddress.getLocalHost();
            /*файл с именем toxic.mp3 должен лежать в корне проекта*/
                  FileInputStream fr =
                        new FileInputStream(
                              new File("toxic.mp3"));
                  DatagramPacket pac;

                  while (fr.read(data) != -1) {
                  //создание пакета данных
pac = new DatagramPacket(data, data.length, addr, 8033);
                        s.send(pac);//отправление пакета
                  }
                  fr.close();
                  System.out.println("Файл отправлен");
            } catch (UnknownHostException e) {
            // неверный адрес получателя
                  e.printStackTrace();
            } catch (SocketException e) {
            // возникли ошибки при передаче данных
                  e.printStackTrace();
            } catch (FileNotFoundException e) {
            // не найден отправляемый файл
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
}
/* пример # 10 : прием данных по протоколу UDP : Recipient.java */
package chapt15;
import java.io.*;
import java.net.*;

public class Recipient {
      public static void main(String[] args) {
            File file = new File("toxic2.mp3");
            System.out.println("Прием данных...");
            try { // прием файла
                  acceptFile(file, 8033, 1000);
```

```
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
      private static void acceptFile(File file, int port,
                  int pacSize) throws IOException {
            byte data[] = new byte[pacSize];
            DatagramPacket pac =
                  new DatagramPacket(data, data.length);
            DatagramSocket s = new DatagramSocket(port);
            FileOutputStream os =
                  new FileOutputStream(file);
            try {
            /* установка времени ожидания: если в течение 10 секунд
            не принято ни одного пакета, прием данных заканчивается */
                  s.setSoTimeout(60000);
                  while (true) {
                        s.receive(pac);
                        os.write(data);
                        os.flush();
                  }
            } catch (SocketTimeoutException e) {
                  // если время ожидания вышло
                  os.close();
                  System.out.println(
      "Истекло время ожидания, прием данных закончен");
            }
      }
}
```

Задания к главе 15

Вариант А

Создать на основе сокетов клиент/серверное визуальное приложение:

1. Клиент посылает через сервер сообщение другому клиенту.
2. Клиент посылает через сервер сообщение другому клиенту, выбранному из списка.
3. Чат. Клиент посылает через сервер сообщение, которое получают все клиенты. Список клиентов хранится на сервере в файле.
4. Клиент при обращении к серверу получает случайно выбранный сонет Шекспира из файла.
5. Сервер рассылает сообщения выбранным из списка клиентам. Список хранится в файле.
6. Сервер рассылает сообщения в определенное время определенным клиентам.
7. Сервер рассылает сообщения только тем клиентам, которые в настоящий момент находятся в on-line.
8. Чат. Сервер рассылает всем клиентам информацию о клиентах, вошедших в чат и покинувших его.