Java Socket Workshop

July 2012

Purpose of this workshop:

The objective of this workshop is to gain experience with writing and compiling programs using the Java programming language.

The exercises provide an introduction to Java datagram sockets and to opening and closing a file.

Table of Content

Background

There is an informative and short tutorial on Java datagrams is available at:
http://java.sun.com/docs/books/tutorial/networking/datagrams/index.html

Java datagram sockets use the UDP transport protocol to transmit traffic. The relationship between Java datagrams and the UDP protocol is described in:
http://www.roseindia.net/java/example/java/net/udp/

1. Programming with Java

The following are a set of exercises that provide an introduction to the Java programming language. The exercises are from the online textbook:

> R. Sedgewick, K. Wayne: Introduciton to Programming in Java
> http://introcs.cs.princeton.edu/java/home/

The online textbook has numerous interesting examples.

Exercise 1.1 Writing a "Hello World" program

The following program prints "Hello World!" in a terminal window.

```
public class HelloWorld {
      public static void main(String[] args) {
      System.out.println("Hello, World");
      }
}
```

Follow the steps from http://introcs.cs.princeton.edu/java/11hello/ (in Dropbox: javafiles/Intro-Java-Hello World.html) to write, compile and run the example.

Exercise 1.2. Program with user argument

The following program prints the command line argument.

```
public class UseArgument {
        public static void main(String[] args) {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.println(". How are you?");
        }
}
```

The program is described in http://introcs.cs.princeton.edu/java/11hello/ (in Dropbox: javafiles/Intro-Java-Hello World.html).

Exercise 1.3. Program with user argument

Write a program that uses a for-loop which generates the following output:

Line 1
Line 2
Line 3
Line 4
Line 5

Exercise 1.4. Adding integers

The following program takes a command line argument N, reads in N integers, and prints out their sum. (E.g., when you type “java AddInts 5”, the program asks for 5 numbers and prints their sum.

```
public class AddInts {
        public static void main(String[] args)     {
        int N = Integer.parseInt(args[0]);
        int sum = 0;
        for (int i = 0; i < N; i++)
                sum = sum + StdIn.readInt();
        System.out.println("Sum is " + sum);
        }
}
```

Compile and run the program. Then, modify the program so that it does not use a command line argument (i.e., you run it as “java AddInts”). Instead, when the program starts it firsts asks to enter the number of integers N.

2. Programming with Datagram Sockets and with Files

The purpose of this part of the lab is to become familiar with programming Datagram sockets and with writing Java programs that read and write data to/from a file. The programs provided in this part intend to offer guidance for the programming tasks needed later on.

Exercise 2.1 Programming with datagram sockets

Compile and run the following two programs. The program *Sender.java* transmits a string to the receiver over a datagram socket. The program *Receiver.java* displays the string when it is received.

Sender.java

```
import java.io.*;
import java.net.*;
public class Sender {
    public static void main(String[] args) throws IOException {
    InetAddress  addr = InetAddress.getByName(args[0]);
    byte[] buf  = args[1].getBytes();
    DatagramPacket packet =
                 new DatagramPacket(buf, buf.length, addr, 4444);
    DatagramSocket socket = new DatagramSocket();
    socket.send(packet);
  }
```

Receiver.java

```
import java.io.*;
import java.net.*;
public class Receiver {
  public static void main(String[] args) throws IOException {
    DatagramSocket socket = new DatagramSocket(4444);
    byte[] buf = new byte[256];
    DatagramPacket packet = new DatagramPacket(buf, buf.length);
    System.out.println("Waiting ...");
    socket.receive(packet);
    String s = new String(p.getData(), 0, p.getLength());
    System.out.println(p.getAddress().getHostName() + ": " + s);
  }
}
```

- Compile the programs.
- Start the receiver by running “java Receiver”.
- Assuming that the receiver is running on a host with IP address 128.100.13.131, start the sender by running:
  java Sender 128.100.13.131 “My String”
- The receiver program should now display the string “My String”.
- Repeat this exercise, with the difference, that you run the sender and receiver on two different hosts.

Exercise 1.2 Reading and Writing data from a file

Download the Java program *ReadFileWriteFile.java*. The program reads an input file "data.txt" which has entries of the form

```
0    0.000000     I     536    98.190 92.170 92.170
4    133.333330   P     152    98.190 92.170 92.170
1    33.333330    B     136    98.190 92.170 92.170
     …            …            …
```

Each line has seven columns:

- Column 1: a sequence number (note that the numbers are slightly out of order)
- Column 2: a time stamp (in milliseconds)
- Column 3: a type field. There are 3 types: I, P, B.
- Column 4: a size field (the unit is byte)
- Columns 5 – 7: not used

The file is read line-by-line, the values in a line are parsed and assigned to variables. Then the values are displayed, and written to a file with name *output.txt*.

- Run the program with the using the following file as input file (found in the Dropbox): **movietrace.data**

- Modify the program so that it computes and displays the average size of the following frame types:

  - Lines with type “I”

  - Lines with type “P”

  - Lines with type “B”.