CS 580 Client-Server Programming
Fall Semester, 2012
Doc 8 Assignment 2
Sept 20, 2012

Assignment 2 Comments

Names

```
public class methods {
```

```
char[] b = new char[50];
in.read(b, 0, 50);
String s = new String(b);
out.close();
in.close();
```

```
public String construct_Message(int totalPeople,int peopleWeight,int
milesPerGallon,int milesPerYr)
```

What is s, s1, s2, ...

```
Scanner s = new Scanner( System.in );
System.out.println("Enter Destination");
String dest=s.next();
String s1="destination:"+dest+";";

System.out.println("Enter Number of people");
String people=s.next();
String s2="people:"+people+";";
```

Java Naming Conventions

n abvtins bcs they r t hd t rd in cd & thrs may thnk temp mns temperature

ClassName
methodName
argumentName
localVariable

Use short meaningful names

Variable names

Indicate the role the variable plays

How readable?

```
int i=(int)(';');
int nextVal;
boolean cont = true;

while(cont){
        response = br.read();
        strBuf.append((char)response);
        if (response ==i) //to check if character that is read is ;
        {
                nextVal= br.read();
                strBuf.append((char)nextVal);
              if (nextVal==i)
              cont = false;
            } //if
```

Use names to make code readable

```
int semicolon =(int)(';');
int nextChar;
boolean atMessageEnd = false;

while(! atMessageEnd){
        nextChar = serverStream.read();
        serverResponse.append((char) nextChar);
        if (nextChar == semicolon)  {
                nextChar = serverStream.read();
            serverResponse.append((char) nextChar);
              if (nextChar == semicolon)
                    atMessageEnd = true;
             }
```

What is wrong with static here?

```
public class TCPClient {

      private static Socket clientSocket ;
       static String destination = "mars";
    static int people =3;
    //trying to test by sending null value for weight
    static float weight=500;
    static float mpg=32;
    static float milesperyear=12000;
```

And here

```
public class Client

    {

        static Socket socket;

        static String DESTINATION="mars";


    private static void send(String trip) {
```

ReadLine - Why is it bad?

```
while ((lineback = bufferedreader.readLine()) != null){
```

Don't use readLine()

End of line is platform dependent

Mars protocol does not send end of line characters

Q. But it worked so can I use it in assignment 3?
I promise not to do it again

A. See previous slide

Q. But it will save me lots of time.
I promise not to do it again

A. See previous slide

What is 13?

```
while ( (c = isr.read()) != 13)
```

Oh, So how could this work?

```
while ( (c = isr.read()) != 13)
```

```
int carriageReturn = (int) '\r'
while ( (c = isr.read()) != carriageReturn)
```

```
while ( (c = isr.read()) != '\r')
```

how different from readLine()?

Why do these work?

```
while ((character = in.read()) != -1)
```

```
do {
        System.out.print((char) next);
} while ((next = inFromServer.read()) != -1);
```

Notice "UTF-8"

```
private static void send(String trip) {              // passing in message "trip"
      try {
            Socket connection = new Socket("bismarck.sdsu.edu", 8009);
            OutputStream rawOut = connection.getOutputStream();
            OutputStream buffered = new BufferedOutputStream(rawOut);
            OutputStreamWriter out = new OutputStreamWriter(buffered, "UTF-8");
            out.write(trip);
            out.flush();    // Flushing stream buffer (ie force sending unfilled buffer)

            InputStream rawIn = connection.getInputStream();
            InputStream bufferedIn = new BufferedInputStream(rawIn);
            InputStreamReader in = new InputStreamReader(bufferedIn, "UTF-8");
```

Common mistake

```
Socket connection = new Socket("bismarck.sdsu.edu", 8009);
OutputStream rawOut = connection.getOutputStream();
OutputStream buffered = new BufferedOutputStream(rawOut);
OutputStreamWriter out = new OutputStreamWriter(buffered);
out.write(trip);
out.flush();

InputStream rawIn = connection.getInputStream();
InputStream bufferedIn = new BufferedInputStream(rawIn);
InputStreamReader in = new InputStreamReader(bufferedIn);
```

What is the default encoding?

How useful?

```
int next;
char[] tempArray = new char[350];
int i = 0;
while ((next = in.read() ) != ';') {  // convert numbers to numbers
        tempArray[i]=(char)next;
        i++;
    }
i++;
tempArray[i]=(char)100;;
    while ((next = in.read() ) != ';') {
        tempArray[i]=(char)next;
        i++;
        }
System.out.println(tempArray);
    out.close();    // Closing connections
}
catch (IOException e) {
    System.out.println("ErrorIO");
}
```

Why Repeat Logic

```
public HashMap<String, Float> sendTrip(int weight, int mpg,
        int milesPerYear, int people) {

    HashMap<String, Float> response = new HashMap<String, Float>();
    String trip = "trip;weight:" + weight + ";mpg:" + mpg
            + ";milesPerYear:" + milesPerYear + ";people:" + people + ";;";
    response = serverResponseToHashMap(send(trip));
    return response;
}

// Overloaded because people is optional
public HashMap<String, Float> sendTrip(int weight, int mpg, int milesPerYear) {
    HashMap<String, Float> response = new HashMap<String, Float>();
    String trip = "trip;weight:" + weight + ";mpg:" + mpg
            + ";milesPerYear:" + milesPerYear + ";;";
    response = serverResponseToHashMap(send(trip));
    return response;
}
```

Do it Once

```
public HashMap<String, Float> sendTrip(int weight, int mpg,
        int milesPerYear, int people) {

    HashMap<String, Float> response = new HashMap<String, Float>();
    String trip = "trip;weight:" + weight + ";mpg:" + mpg
            + ";milesPerYear:" + milesPerYear + ";people:" + people + ";;";
    response = serverResponseToHashMap(send(trip));
    return response;
}

// Overloaded because people is optional
public HashMap<String, Float> sendTrip(int weight, int mpg, int milesPerYear) {
    return sendTrip(weight, mpg, milesPerYear, 1);
}
```

Not Usable

```
void sendMessage() throws IOException{
        oStream = new PrintWriter(new
          OutputStreamWriter(toMars.getOutputStream(),"UTF-8"),true);
        oStream.println("trip;destination:mars;people:3;weight:500;mpg:
32;milesperyear:12000;;");
        oStream.println("quit;;");

        System.out.println("Information sent to server for validation.");
        oStream.flush();
```

System.exit()

```
} catch (IOException e) {
                System.err.println("Couldn't get I/O for "
                        + "the connection to: bismarck.sdsu.edu with port 8009.");
                System.exit(1);
            }
```

Outline of a Solution

Outline of Solution

Pubic interface

public class MarsServer {

public MarsServer(String host, int port)

public String quit()

public ArrayList trip(int people, float weight, float mpg, float milesPerYear)

public ArrayList trip(float weight, float mpg, float milesPerYear)

public void connect()

UpToReader

```
public class UpToReader extends FilterReader {
      public UpToReader(Reader in) {super(in); }

      public String upto(char end) throws IOException {
     // TODO make buffer growable, read in blocks
            int EOF = -1;
            CharBuffer buffer = CharBuffer.allocate(128);
            int nextChar;
            while (( nextChar = super.read()) != EOF )  {
                  if (nextChar == end )
                        break;
                  buffer.append( (char)nextChar);
            }
            buffer.flip();
            return buffer.toString();
            }

      public String upto(String end) throws IOException (source not shown)
```

MarsServer

```
private Socket server;
private UpToReader in;
private Writer out;

public void connect() throws UnknownHostException, IOException {
        server = new Socket(host, port);
        in = reader();
        out = writer();
}

        public UpToReader reader() throws UnsupportedEncodingException, IOException
{
                return new UpToReader( new InputStreamReader( server.getInputStream(),
"UTF8"));
        }

        public Writer writer() throws UnsupportedEncodingException, IOException {
                return new OutputStreamWriter( server.getOutputStream(), "UTF8");
        }
```

Helpful methods

```
public static String quitMessage() {
        return "quit;;";
}

public static String tripMessage(int people, float weight, float mpg, float malesPerYear) {
        return "trip;destination:mars;people:" + people + ";weight:" + weight
                        + ";mpg:" + mpg + ";milesperyear:" + malesPerYear + ";;";
}
```

```
private void send(String message ) throws IOException {
        if (out == null) {
                connect();
        }
        out.append(message);
        out.flush();
}
```

Trip

```
public ArrayList trip(int people, float weight, float mpg, float milesPerYear) {
      try {
            send(tripMessage(people,weight,mpg,milesPerYear));
            String serverResponse = readResponse();
   return parseResponse(serverResponse);
     } catch (Exception error) {
            Log.e("rew",error.toString(),error);
   ArrayList error = new Arraylist();
      error.add("Error");
      error.add(error.toString());
            return error;
        }
   }

private String readResponse() {
    return in.upTo(";;"):
}
```

Using the Server

```
MarsServer server = new MarsServer();
ArrayList response = server.trip(1,1,1,1);

if (response.get(0) == "ok") {
    System.out.println("food: " + response.get(1));
    System.out.println("weight: " + response.get(2));
```

Testing the Server

```
public final void testTrip() throws UnknownHostException, IOException {
        MarsServer server = new MarsServer();
    ArrayList response = server.trip(1,1,1,1);

        assertEquals("type",response.get(0), 'ok');
        assertEquals("food",response.get(1).floatValue(),53764.3f, 0.01f);
        assertEquals("weight",response.get(2).floatValue(),61.1f, 0.01f);
     }
```

Testing

Testing

Johnson's Law

If it is not tested it does not work

The more time between coding and testing

- More effort is needed to write tests
- More effort is needed to find bugs
- Fewer bugs are found
- Time is wasted working with buggy code
- Development time increases
- Quality decreases

Unit Testing

Tests individual code segments

Automated tests

When to Write Tests

First write the tests

Then write the code to be tested

Writing tests first saves time

- Makes you clear of the interface & functionality of the code
- Removes temptation to skip tests

XUnit

Free frameworks for Unit testing

SUnit originally written by Kent Beck 1994

JUnit written by Kent Beck & Erich Gamma

Available at: http://www.junit.org/

Ports to many languages at:

http://www.xprogramming.com/software.htm

Android and JUnit

Support of JUnit part of Android

Only supports JUnit 3 type tests

JUnit Example - JUnit 3.x

Goal: Implement a Stack containing integers.

Tests:

Subclass junit.framework.TestCase

Methods starting with 'test" are run by TestRunner

Sample Testcase

```
import junit.framework.*;
public class Sampletest extends TestCase {
        public void testPushPop() {
                Stack<String> test = new Stack<String>();
                assertTrue( test.isEmpty() );
                test.push("A");
                assertFalse( test.isEmpty() );
                test.push("B");
                test.push("C");
                assertEquals("C", test.pop());
                assertEquals("B", test.pop());
                assertEquals("A", test.pop());
                assertTrue( test.isEmpty() );
                try {
                        test.pop();
                        fail();
                } catch (EmptyStackException e) {
                }
        }
}
```

Assert Methods

assertTrue()
assertFalse()
assertEquals()
assertNotEquals()
assertSame()
assertNotSame()
assertNull()
assertNotNull()
fail()

For a complete list see

http://junit.sourceforge.net/javadoc/org/junit/Assert.html