# CS 580 Client-Server Programming

Spring Semester, 2009

Assignment 3 Part One Comments

17 March, 2009

## Subversion Issues

Assignment 2

Three students still have problems

Assignment 3

Five Students have problems

## Parsing Command line Part 1

```
public static void main(String args[]){
        SDWitterServer server;
        try{
                if(args.length==0){
                        server=new SDWitterServer("defaultConfiguration.conf");
                }
                else if(args.length==2){
                        if(args[0].equals("-port"))
                                server=new SDWitterServer(Integer.parseInt(args[1]),"defaultConfiguration.conf");
                        else if(args[0].equals("-log"))
                                server=new SDWitterServer(args[1],"defaultConfiguration.conf");
                        else if(args[0].equals("-conf"))
                                server=new SDWitterServer(args[1]);
                        else
                                throw new InvalidAttributesException();
                }
```

## Parsing Command line Part 2

```
else if(args.length==4){
        String flag1=args[0];
        String flag2=args[2];
        if(flag1.equals("-port")){
                if(flag2.equals("-log"))
                        server=new SDWitterServer(Integer.parseInt(args[1]),args[3],"defaultConfiguration.conf");
                else if(flag2.equals("-conf"))
                        server=new SDWitterServer(Integer.parseInt(args[1]),args[3]);
                else
                        throw new InvalidAttributesException();
        }
        else if(flag1.equals("-log")){
                if(flag2.equals("-port"))
                        server=new SDWitterServer(Integer.parseInt(args[3]),args[1],"defaultConfiguration.conf");
                else if(flag2.equals("-conf"))
                        server=new SDWitterServer(args[1],args[3]);
                else
                        throw new InvalidAttributesException();
        }
        else if(flag1.equals("-conf")){
                if(flag2.equals("-port"))
                        server=new SDWitterServer(Integer.parseInt(args[3]),args[1]);
                else if(flag2.equals("-log"))
                        server=new SDWitterServer(args[3],args[1]);
                else
                        throw new InvalidAttributesException();
        }
        else
                throw new InvalidAttributesException();
}
```

## Parsing Command line Part 3

```
            else if(args.length==6){
                int port=8010;
                String conf="defaultConfiguration.conf";
                String log="";
                for(int i=0;i<6;i=i+2){
                    if(args[i].equals("-port"))
                        port=Integer.parseInt(args[i+1]);
                    else if(args[i].equals("-log"))
                        log=args[i+1];
                    else if(args[i].equals("-conf"))
                        conf=args[i+1];
                    else
                        throw new InvalidAttributesException();
                }
                server=new SDWitterServer(port,log,conf);
            }
            else
                throw new InvalidAttributesException();
            try{
                server.run();
            }catch(IOException ioe){
                server.logger.log(Level.ALL," Some error in Starting server at port" + server.port+ ioe.getMessage());
            }
        }catch(InvalidAttributesException iae){
            System.out.println("Invalid arguments");
        }
    }
```

## A Small Improvement

```
public static void main(String args[]){
        SDWitterServer server;
        try{
            server = configureServer(args);
            server.run();
        } catch(IOException ioe) {
            server.logger.log(Level.ALL," Some error in Starting server at port" +
          server.port + ioe.getMessage());
        } catch(InvalidAttributesException iae) {
            System.out.println("Invalid arguments");
        }
```

## A Small Improvement

```
private Server static configureServer(String[] args) {
    if(args.length==0)
        return defaultServer();
    if(args.length==2)
        return twoArgumentServer(args);
    if(args.length==4)
        return fourArgumentServer(args);
    if(args.length==6)
        return sixArgumentServer(args);
    throw new InvalidAttributesException();
}
```

## Options Order

Why should the user need to know your order?

```
java SDwitterServer -port 8888 -configFile foo -logfile bar
java SDwitterServer -port 8888  -logfile bar
java SDwitterServer -logfile bar -port 8888 -configFile foo
java SDwitterServer -configFile foo -p 8888
```

How are you going to support all variations

## flag

```
public SDWitterServer(String logOrConfi,int flag){
    //if flag is 0 then logOrConfi is a Log File Path
```

## My

```
private Logger myLog;
private Properties myProperties;
private String loggerName;
private int myPort;
private int latestLogFull;
private ServerSocket myServer;
```

## Which is Better

```
int EOC = 59;// ';' character
```

Verses

```
int EOC = (int) ';'
```

## How to set the port number?

```
public static void main(String [ ] args) throws IOException{
    SDWitterServer newServer;
    switch(args.length){
    case 0:newServer=new SDWitterServer();break;
    case 1:newServer=new SDWitterServer(Integer.parseInt(args[0]));break;
    case 2:if(args[0].equals("-l")){
        newServer=new SDWitterServer(args[1],0);
        break;
    }else if(args[1].equals("-c")){
        newServer=new SDWitterServer(args[1],1);
        break;
    }
    case 3:if(args[1].equals("-l")){
        newServer=new SDWitterServer(Integer.parseInt(args[0]),args[2],0);
        break;
    }else if(args[1].equals("-c")){
        newServer=new SDWitterServer(Integer.parseInt(args[0]),args[2],1);
        break;
    }
```

## How does the User know how start server?

```
public static void main(String[] args) throws IOException {
    //int port = 0;
    Properties properties = null;
    if(args.length == 0 || args.length > 2){
        System.out.println(usage());
    etc.



private static String usage() {
    return "Usage: SDWitterServer -port <port number> | -config <config file>";
}
```

## -h or -help

```
java -h
java -help
```

```
Usage: java [-options] class [args...]
          (to execute a class)
   or  java [-options] -jar jarfile [args...]
          (to execute a jar file)

where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available (implies -server, only for x86_64)
    -client  to select the "client" VM
    -server       to select the "server" VM
etc
```

## Part 1 of 3

```
public static void main(String args[]) throws IOException
{
    if(args.length>0)
        parseArguments(args);
    else
    {
        SDwitterServer test=new SDwitterServer();
        test.run();
    }
}
```

## Part 2 of 3

```
public static void parseArguments(String args[]) throws IOException
{


    if(args.length==2)
    {
        check_flag(args);
    }

    else if(args.length==4)
    {
        checkBothFlags(args);
    }


}
```

## Part 3

```
private static void check_flag(String args[]) throws IOException
{
    SDwitterServer test=null;
    defaultConfigFile=new DefaultConfiguration(defaultConfigFileName);
    if(args[0].equalsIgnoreCase("-p"))
        test=new SDwitterServer(Integer.parseInt(args[1]),"");

    else if(args[0].equalsIgnoreCase("-l"))
        test=new SDwitterServer(defaultConfigFile.getPortNumber(),args[1]);

    else if(args[0].equalsIgnoreCase("-conf"))
        test=new SDwitterServer(args[1]);
    test.run();
}
```

Edited to fit one slide

## Find The Bug part 1 of 2

```
public void run() throws IOException{
    ServerSocket server=new ServerSocket(port);
    while(true){
        Socket client=server.accept();
        logger.info("Request from"+client.getInetAddress());
        processRequest(client.getInputStream(),client.getOutputStream());
    }
    }
```

# Find The Bug part 2

```
public void processRequest(InputStream in,OutputStream out) throws IOException{
        fromClient=in;
        toClient=out;
        String messageString= receive();
        if(messageString.startsWith("login;")){
                logger.log(Level.ALL,messageString);
                send("ok:success;;");
        }
        else if(messageString.startsWith("screenName:")){
                logger.log(Level.ALL,messageString);
                send("ok:available;;");
        }
        //three else if statements removed to save space on slide
        else if(messageString.startsWith("quit")){
                logger.log(Level.ALL,messageString);
                send("ok:quit;;");
        }
}
```

## How is This?

```
public void run(int port) throws IOException {
    ServerSocket input = new ServerSocket(port);
    while (true) {
        Socket connection = input.accept();
        processRequest(connection.getInputStream(), connection
                .getOutputStream());
        connection.close();
    }
}

void processRequest(InputStream input, OutputStream output)
        throws IOException {
    UpToStream uts = new UpToStream(input, "UTF-8");
    String uptoLastTwoChars = uts.upto(";;");
    MessageReader msgObject = new MessageReader(uptoLastTwoChars);
    MessageType messageObject = new MessageType(logFile, uptoLastTwoChars);
    output.write(msgObject.response().getBytes());
    output.flush();
}
```

## How is this for Server Tests?

```
    public void testConstructor() throws IOException, SecurityException,
NoSuchFieldException, IllegalArgumentException, IllegalAccessException {
        File f = new File(".");
        System.out.println(f.getCanonicalPath());

        Properties properties = new Properties();
        String configFile = new String("src/main/resources/config.properties");
        FileInputStream in = new FileInputStream(configFile);
        properties.load(in);
        SDWitterServer server = new SDWitterServer(properties);

        Class c = server.getClass();
        Field field = c.getDeclaredField("port");
        field.setAccessible(true);
        assertEquals(8010, field.getInt(server));
        field.setAccessible(false);
    }
```

## Response

```
public void processRequest(UpToStream in,OutputStreamWriter out) throws IOException
{

    clientMessages=new MessageReader(in);
    while(true)
    {
        String response=clientMessages.readUpTo(";;");
        System.out.println("Response:"+response);
        witterLog.log(Level.ALL,"[client "+clientSocket.getInetAddress()+" ] "+response);

        if(response.startsWith("login"))
            login(response);
        etc.
```

## Protocol?

```
            Writer out = new OutputStreamWriter(connection.getOutputStream( ),
"UTF8");
            out.write("ok:successfully connected;;");
            out.flush( );
            connection.close( );
```

## Path Problems

```
    private static String defaultConfigFile = "c:\\documents and settings\\foo bar\\my documents\
\school\\cs580\\foobar\\assignment3\\SDWitterServer.properties";
    private static String defaultPort = "8010";
```

## A ReadMe File :)

The main class is SDwitterServerGUI.  SDwitterServer contains a main class but is set to run the JUnit tests.  The SDwitterServerTest uses the previously implemented client to test the server so SDwitterClient is included.  The GUI basically just wraps the console in a JTextArea to be displayed in Swing.

Requests are logged to both file and console at the INFO level.  Each request log contains an identifier for the client, which includes the IP address and a unique id. New connections are also logged.

Program usage:

```
javac sdsu\cs580\SDwitter\*.java
java sdsu.cs580.SDwitter.SDwitterServerGUI [-p portNum] [-c configFile] [-l logFile]
```

## Will this Work?

```
  public static void main (String args[]) throws Exception {
          String source = "c\u1561t";
          InputStream byteSource = new ByteArrayInputStream(source.getBytes("UTF-8"));

          BufferedReader inReader = new BufferedReader(new
InputStreamReader(byteSource));
          int next;
          byte[] buffer = new byte[10];
          int index = 0;
          while((next = inReader.read()) != -1) {
               buffer[index++] = (byte) next;
          }
          String result = new String(buffer, "UTF-8");
          System.out.println(result.equals(source));
          System.out.println(result);
}
```

## Output of Last Line

```
System.out.println(result);
```

Using Xcode

```
c??t
```

Using TextMate

```
cat
```

Eclipse

```
c??t
```

Command line compile

```
c??t
```

## Does not Compile

```
public static void main (String args[]) throws Exception {
        String source = "c\u1561t";
        InputStream byteSource = new ByteArrayInputStream(source.getBytes("UTF-8"));

        BufferedReader inReader = new BufferedReader(new
InputStreamReader(byteSource));
        int next;
        char[] buffer = new char[3];
        int index = 0;
        while((next = inReader.read()) != -1) {
            buffer[index++] = (char) next;
        }
        String result = new String(buffer, "UTF-8");
        System.out.println(result.equals(source));
        System.out.println(result);

 }
```

## Compiles But Results Vary

```
   public static void main (String args[]) throws Exception {
           System.out.println(Charset.defaultCharset());
           String source = "c\u1561t";
           InputStream byteSource = new ByteArrayInputStream(source.getBytes("UTF-8"));

           BufferedReader inReader = new BufferedReader(new
InputStreamReader(byteSource));
           int next;
           char[] buffer = new char[3];
           int index = 0;
           while((next = inReader.read()) != -1) {
                  buffer[index++] = (char) next;
           }
           String result = new String(buffer);
           System.out.println(result.equals(source));
           System.out.println(result);                    //cᕡt
```

```
public static void main (String args[]) throws Exception {
        String source = "c\u1561t";
        System.out.println(source);
        InputStream byteSource = new ByteArrayInputStream(source.getBytes("UTF-8"));

        BufferedReader inReader = new BufferedReader(
                                    new InputStreamReader(byteSource,"UTF-8"));
        int next;
        char[] buffer = new char[3];
        int index = 0;
        while((next = inReader.read()) != -1) {
              buffer[index++] = (char) next;
        }
        String result = new String(buffer);
        System.out.println(result.equals(source));
        System.out.println(result);
}
```

# Can't Save Source File

```
import java.io.ByteArrayInputStream;
import java.io.IOException;

import junit.framework.TestCase;

public class TestUpToStream extends TestCase {
	UpToStream uts;

	public void testUpto() throws IOException {
		ByteArrayInputStream in = new ByteArrayInputStream("pqrsaᓂa;;bb"
				.getBytes("UTF-8"));
		uts = new UpToStream(in, "UTF-8");
		assertEquals(uts.upto(";;"), "pqrsaᓂa;;");
	}
}
```

```
Socket client = myServer.accept();  // wait for a connection
log.info("Request from " + myServer.getInetAddress());

DataInputStream parsedInput = new DataInputStream(client.getInputStream());
PrintWriter parsedOutput = new PrintWriter(client.getOutputStream(), true);
```