

CS 580 Client-Server Programming
Spring Semester, 2010
Doc 11 NIO, Logging, Configuration
4 March, 2010

Copyright ©, All rights reserved. 2010 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

Reference

Java On-line API, <http://java.sun.com/javase/6/docs/api/>
SDSU Java Library, <http://www.eli.sdsu.edu/java-SDSU/>

Assignment 3 part 2 Comments

Names

```
serverPort = new javax.swing.JTextField();  
serverName = new javax.swing.JTextField();  
jLabel1 = new javax.swing.JLabel();  
jLabel2 = new javax.swing.JLabel();  
serverStatus = new javax.swing.JTextField();  
jLabel3 = new javax.swing.JLabel();  
jLabel4 = new javax.swing.JLabel();
```

About that comment

```
public String[] chatWaitingList() throws Exception {
    String[] newList;
    if ( currentState > START ) {
        //String currentList = sendAndReceive("waitingList;;");
        String currentList = "ok:1;nickname:biteme;;";
        if ( currentList.indexOf(";nickname:") >= 0 ) {
            currentList = currentList.substring(currentList.indexOf(";nickname:")+10, currentList.length() - 2);
            newList = currentList.split(";nickname:");
            for ( int ctr = 0; ctr < newList.length; ctr++ )
                newList[ctr] = unescapeString(newList[ctr]);
        }
        else
            newList = new String[0];
        return newList;
    }
    return null;
}
```

Include your external Libraries

```
.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
```

Duh

```
// Static variables  
private static final int START = 0;  
private static final int AUTHENTICATED = 1;  
private static final int HANDSHAKE = 2;
```

Hard Coded

```
private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_loginButtonActionPerformed
    if (chatClient == null) {
        chatClient = new SDChatClient("bismarck.sdsu.edu", 8009);
    }
    String password = "";
    for (int i = 0; i < passwordField.getPassword().length; i++) {
        password += passwordField.getPassword()[i];
    }
    boolean success = chatClient.login(nicknameTextField.getText(), password);
    if (success) {
        dispose();
        new SDChatWaitingListFrame(chatClient).setVisible(true);
    } else {
        displayError("Login Failed!");
    }
}
} //GEN-LAST:event_loginButtonActionPerformed
```


Packages

- ▼ >src
 - AvailableCommand.java
 - LoginCommand.java
 - NicknameCommand.java
 - QuitCommand.java
 - RegisterCommand.java
 - SDChatClient.java
 - SDChatClientTest.java
 - SDChatCommand.java
 - SDChatCommandsTest.java
 - SDChatCreateAccountFrame.java
 - >SDChatLoginFrame.java
 - SDChatMessage.java
 - SDChatMessageTest.java
 - SDChatReader.java
 - SDChatReaderTest.java
 - SDChatWaitingListFrame.java
 - WaitingListCommand.java

- ▼ src
 - ▶ com
 - ▼ edu
 - ▼ sdsu
 - ▼ cs
 - ▼ sd
 - ▼ chat
 - ▶ message
 - ▶ test
 - ▶ threads
 - ▶ ui
 - Images.java
 - ISdChatConstants.java
 - ISdChatControls.java
 - MessageDialogFactory.java
 - SdChat.java
 - SdChatReader.java
 - UnknownResponseExceptior
 - ▶ images

What happens with Exceptions

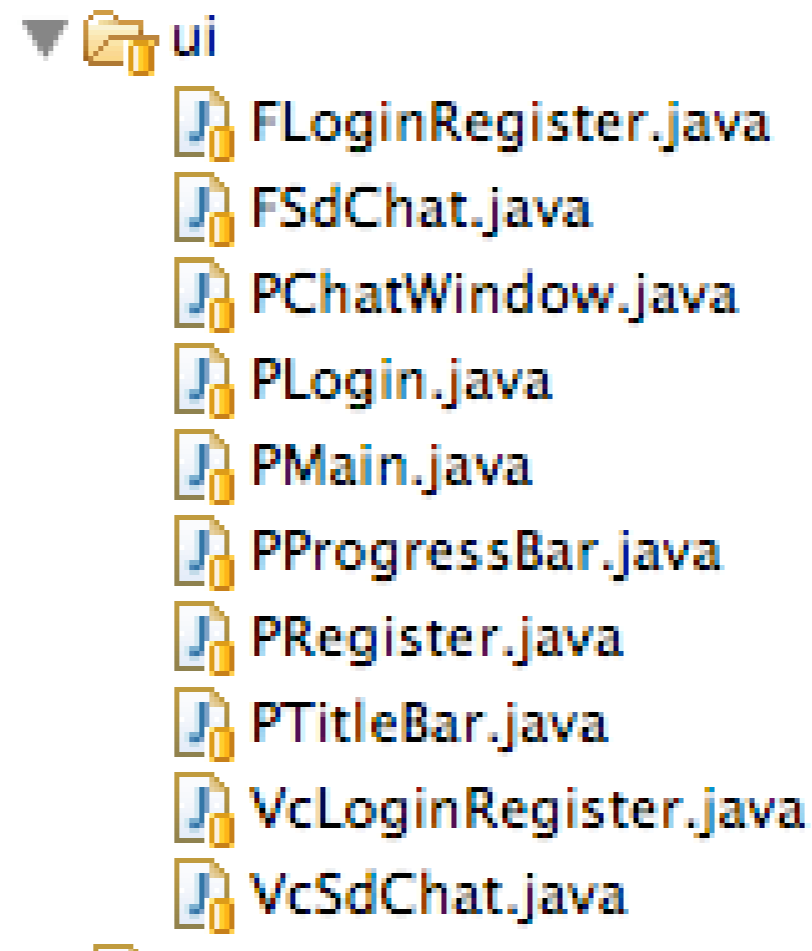
Turn off your network connections

Run your Client

Try connecting to the server

What happens?

Which class to Start the Application?



NIO

Motivation

```
public class SomeClientThread extends Thread {
    private Socket connection;

    public SomeCleintThread(Socket toServer) {
        connection = toServer;
    }

    public run() {
        InputStream rawIn = connection.getInputStream();
        BufferedReader in = new BufferedReader(new InputStreamReader(rawIn));
        while ( !isInterrupted() ) {
            String answer = in.readLine();
            process input here
        }
        in.close();
    }
}
```

Scenario

User makes a request

Client code connects to server

User changes mind - clicks on "Stop" button

"Stop" button sends interrupt to thread

What happens?

Answer

Thread remains block in I/O

Java interrupt ()

Sent to a thread to interrupt it

If thread is blocked on a call to wait, join or sleep

InterruptedException is thrown &

The interrupted status flag is cleared

if the thread is blocked on I/O operation on an interruptible channel (NIO)

ClosedByInterruptException is thrown

The interrupted status flag is set

If the thread is blocked by a selector (NIO)

Interrupt status is set

The thread returns from the selector call as normal

If none of the other conditions hold then the thread's interrupt status is set

NIO

New I/O

Non-blocking I/O

Buffers for data of primitive types

Character set encoders and decoders

A pattern-matching facility based on Perl-style regular expressions

Channels

Interruptible I/O

Blocking & non-blocking I/O

A file interface that supports locks and memory mapping of files

A multiplexed, non-blocking I/O facility for writing scalable servers

Channels (`java.nio.channels`)

Open connection to an entity such as

- hardware device

- file

- network socket

- program component

that is capable of performing I/O operations

Buffer (java.nio)

Buffers for different types

ByteBuffer

CharBuffer

DoubleBuffer

FloatBuffer

IntBuffer

LongBuffer

MappedByteBuffer

ShortBuffer

```
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SocketChannel;
```

Example

```
public class NIOExample {
    public static void main(String[] args) throws IOException {
        SocketChannel sdChatServer = SocketChannel.open();
        sdChatServer.configureBlocking(true);
        sdChatServer.connect(new InetSocketAddress("bismarck.sdsu.edu", 8009));
        ByteBuffer ioBuffer = ByteBuffer.allocate(1024);
        try {
            String message = "nickname:foo;;";
            ioBuffer.put(message.getBytes("UTF8"));
            ioBuffer.flip();
            int bytesWritten = sdChatServer.write(ioBuffer);
        } catch (IOException e) {
            System.out.println("Socket write error");
        }
    }
}
```

Example Continued

```
try {  
    ioBuffer.clear();  
    int numberBytesRead = sdChatServer.read(ioBuffer);  
  
    if (numberBytesRead == -1) {  
        sdChatServer.close();  
    } else {  
        ioBuffer.flip();  
        byte[] responseBytes = new byte[numberBytesRead];  
  
        ioBuffer.get(responseBytes, 0, numberBytesRead - 1);  
        String response = new String(responseBytes, "UTF8");  
        System.out.println(response);  
    }  
} catch (IOException e) {  
    System.out.println("Socket read error");  
}  
sdChatServer.close();  
}
```

Blocking IO Example

```
public class NIOThreadExample extends Thread {  
  
    public void run() {  
        SocketChannel sdChatServer = null;  
        try {  
            System.out.println("Start Run");  
            sdChatServer = SocketChannel.open(new InetSocketAddress(  
                "bismarck.sdsu.edu", 8009));  
            ByteBuffer ioBuffer = ByteBuffer.allocate(1024);  
            System.out.println("Start Read");  
            int numberBytesRead = sdChatServer.read(ioBuffer);  
            System.out.println("Bytes read " + numberBytesRead);  
        } catch (ClosedByInterruptException error) {  
            System.out.println("Closed " + error.toString());  
            System.out.println(sdChatServer.isConnected());  
        } catch (Exception error) {  
            System.out.println("Error " + error.toString());  
        }  
    }  
}
```

Running the Example

```
public static void main(String[] args) {
    NIOThreadExample blocked =
new NIOThreadExample();
    blocked.start();
    try {
        Thread.sleep(200);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    blocked.interrupt();
    System.out.println("End");
}
```

Output

```
Start Run
Start Read
End
Closed java.nio.channels.ClosedByInterrupt
```

Non-Blocking IO Example

```
public void run() {
    SocketChannel sdChatServer = null;
    try {
        System.out.println("Start Run");
        sdChatServer = SocketChannel.open(new InetSocketAddress(
            "bismarck.sdsu.edu", 8009));
        sdChatServer.configureBlocking(false);
        ByteBuffer ioBuffer = ByteBuffer.allocate(1024);
        System.out.println("Start Read");
        System.out.println(sdChatServer.isConnected());
        int numberBytesRead = sdChatServer.read(ioBuffer);
        System.out.println("Bytes read " + numberBytesRead);
    } catch (ClosedByInterruptException error) {
        System.out.println("Closed " + error.toString());
        System.out.println(sdChatServer.isConnected());
    } catch (Exception error) {
        System.out.println("Error " + error.toString());
    }
}
```


Running the Example

```
public static void main(String[] args) {  
    NIOThreadExample blocked =  
new NIOThreadExample();  
    blocked.start();  
    try {  
        Thread.sleep(200);  
    } catch (InterruptedException e) {  
        e.printStackTrace();  
    }  
    blocked.interrupt();  
    System.out.println("End");  
}
```

Output

```
Start Run  
Start Read  
true  
Bytes read 0  
End
```

Application Parameters & Configuration Files

Application Parameters & Configuration Files

Applications normally have configuration files to store

- User preferences

- Cached values

- Window settings

- Port numbers

- Database connection information

- Log file information

- Recent documents/web pages

- Cookies

- Values that need changing without recompiling

cv\$ co assignment2

ls -la

ps -aux

Servers normally use configuration files & command line flags

Environment variables are not used much in servers (why?)

Some systems have libraries to handle config files & command line arguments

JDK does not seem to have such classes

There should be a number of Java libraries that provide such support

sdsu Java library is one such library

Parses

Configuration files

Command line arguments

Command Line argument

-flag=value

-flag value

-flag

--xyz

-- (ignore rest of the command line)

File Formats

properties format

#A comment to the end of the line

key1=value1

key2=value2 with spaces

key3 with spaces=value3 #part of the value

sdsu.util.LabeledData format

#A comment to the end of the line,

key1 = value1;

key2='value2 with spaces';

'key3 with spaces'=value3; # a comment

```
import sdsu.util.ProgramProperties;

public class ConfigurationExample {
    public static void main(String args[]) {
        try {
            ProgramProperties flags =
                new ProgramProperties( args, "configurationFile");
            String nameValue =
                flags.getString( "name" , "No name given");
            int size = flags.getInt( "size", 0);
            boolean switchOn = flags.containsKey( "s");
            System.out.println( " nameValue: " + nameValue);
            System.out.println( " size: " + size);
            System.out.println( " switchOn: " + switchOn);
        }
        catch (java.io.IOException readParseProblem) {
            System.err.println( "Program aborted on error " +
                readParseProblem);
        }
    }
}
```



```
java ConfigurationExample
```

Output

```
nameValue: Roger  
size: 12  
switchOn: false
```

File "configurationFile.labeledData"

```
name=Roger;  
size=12;
```

```
java ConfigurationExample -s -name Pete
```

Output

```
nameValue: Pete  
size: 12  
switchOn: true
```

```
java ConfigurationExample -conf=otherFile
```

Output

```
nameValue: Sam  
size: 8  
switchOn: true
```

```
require 'optparse'

class SampleOptionParser
  def initialize
    parseOptions(ARGV)
  end

  def parseOptions(args)
    options = OptionParser.new
    options.on("-x") {|value| @x = true}
    options.on("-s SIZE", "--size SIZE", Integer, "Size of new file in bytes") {|size| @fileSize = size}
    options.on("-p=[PORT]", "--port=[PORT]", Integer,
      "Port for server") {|port| @fileSize = size}
    options.on_tail("-h", "--help", "Show this message") do
      puts options.to_s
      exit
    end
    options.on_tail("--version", "Show version") do
      puts OptionParser::Version.join(".")
      exit
    end

    options.parse(args)
  end
end
```

```
A1 77->ruby SampleOptionParser.rb --h
```

```
Usage: SampleOptionParser [options]
```

```
-x
```

```
-s, --size SIZE
```

```
Size of new file in bytes
```

```
-p, --port=[PORT]
```

```
Port for server
```

```
-h, --help
```

```
Show this message
```

```
--version
```

```
Show version
```

Performance tuning
Upgrade justification
Problem tracking
Access counting

Logging

Date and time

Service that caused the entry

Client address that caused the entry

Host on which the server runs

Event

Apache Access Log

```
211.90.88.43 - - [21/Oct/2002:08:33:29 -0700] "GET /scripts/..%25%35%63../winnt/  
system32/cmd.exe?/c+dir HTTP/1.0" 404 303
```

```
211.90.88.43 - - [21/Oct/2002:08:33:30 -0700] "GET /scripts/..%252f../winnt/system32/  
cmd.exe?/c+dir HTTP/1.0" 404 303
```

Apache Error Log

```
[Mon Oct 21 08:33:29 2002] [error] [client 211.90.88.43] File does not exist: /opt/etc/  
apache-1.3.26/htdocs/scripts/..%5c../winnt/system32/cmd.exe
```

```
[Mon Oct 21 08:33:30 2002] [error] [client 211.90.88.43] File does not exist: /opt/etc/  
apache-1.3.26/htdocs/scripts/..%2f../winnt/system32/cmd.exe
```

Multiple log levels
Multiple output formats
Output to different IO devices

Filters for additional filtering of message to accept
ResourceBundles for localization of log messages
Initialization of loggers by configuration file
Hierarchical loggers in one program

Log Levels

ALL
SEVERE (highest value)
WARNING
INFO (usual default)
CONFIG
FINE
FINER
FINEST (lowest value)
OFF

Output formats

XML (default for files output)
Normal Text (default for screen output)

Output devices

Stream
System.err
File or rotating set of files
Socket for network logging
Memory

```
import java.util.logging.*;
public class SimpleLoggingExample {
    private static Logger logger = Logger.getLogger("edu.sdsu.cs580");

    public static void main (String args[]) {
        new SimpleLoggingExample().someLogMessages();
    }

    public void someLogMessages() {
        logger.severe( "A severe log message");
        Logger.getLogger("edu.sdsu.cs580").fine( "A fine message");
        logger.warning( "Be careful" );
    }
}
```

Output To System.err

Feb 16, 2004 10:51:37 PM Logging someLogMessages SEVERE: A severe log message

Feb 16, 2004 10:51:37 PM Logging someLogMessages WARNING: Be careful

Default Settings

Use a ConsoleHandler

Level set to INFO

System administrator can change default settings

Convenience Methods

```
severe( String message);  
warning( String message);  
info( String message);  
config( String message);  
fine( String message);  
finer( String message);  
finest( String message);
```

Convenience Methods for Tracing Methods

```
entering(String sourceClass, String sourceMethod);  
entering(String sourceClass, String sourceMethod, Object parameter);  
entering(String sourceClass, String sourceMethod, Object[] parameters);  
exiting(String sourceClass, String sourceMethod);  
exiting(String sourceClass, String sourceMethod, Object result);
```

Log Methods

```
log(Level logLevel, String message);  
log(Level logLevel, String message, Object parameter);  
log(Level logLevel, String message, Object[] parameters);  
log(Level logLevel, String message, Throwable exception);
```

```
import java.io.*;
import java.util.Vector;
import java.util.logging.*;

public class MessageTypes {
    private static Logger logger = Logger.getLogger("edu.sdsu.cs580");

    static {
        try {
            Handler textLog = new FileHandler("textLog.txt");
            textLog.setFormatter( new SimpleFormatter());
            textLog.setLevel(Level.ALL);
            Handler xmlLog = new FileHandler("xmlLog.txt");
            xmlLog.setFormatter( new XMLFormatter());
            xmlLog.setLevel(Level.ALL);

            logger.addHandler(textLog);
            logger.addHandler(xmlLog);
            logger.setLevel(Level.ALL);
        }
        catch (IOException fileError) {
            System.err.println( "Could not open log files");
        }
    }
}
```

```
public static void main (String args[]) {
    new MessageTypes().someLogMessages();
}

public void someLogMessages() {
    logger.entering("MessageTypes", "someLogMessages");
    Vector data = new Vector();
    data.add( "Cat");
    logger.log(Level.SEVERE, "Show Vector", data);
    logger.severe( "A severe log message");
    logger.logp(Level.SEVERE, "MessageTypes", "someLogMessages", "Logp example");
    try {
        int zeroDivide = 1/ (1 - 1);
    }
    catch (Exception zeroDivide) {
        logger.log(Level.SEVERE, "Exception example", zeroDivide);
    }
    logger.exiting("MessageTypes", "someLogMessages");
}
}
```

SimpleFormatter Output

```
Feb 16, 2004 11:01:53 PM MessageTypes someLogMessages FINER: ENTRY
Feb 16, 2004 11:01:53 PM MessageTypes someLogMessages SEVERE: Show Vector
Feb 16, 2004 11:01:53 PM MessageTypes someLogMessages SEVERE: A severe log message
Feb 16, 2004 11:01:54 PM MessageTypes someLogMessages SEVERE: Logp example
Feb 16, 2004 11:01:54 PM MessageTypes someLogMessages SEVERE: Exception example
java.lang.ArithmeticException: / by zero
    at MessageTypes.someLogMessages(MessageTypes.java:45)
    at MessageTypes.main(MessageTypes.java:32)
Feb 16, 2004 11:01:54 PM MessageTypes someLogMessages FINER: RETURN
```

XMLFormatter Sample Output

```
<?xml version="1.0" encoding="US-ASCII" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2004-02-16T23:01:53</date>
  <millis>1077001313695</millis>
  <sequence>0</sequence>
  <logger>edu.sdsu.cs580</logger>
  <level>FINER</level>
  <class>MessageTypes</class>
  <method>someLogMessages</method>
```

Can be set to rotate files

Can be located in temp directory

Can be set to

- Append existing files

- Overwrite existing files (default)

To change append setting either

Use constructor

```
FileHandler(String pattern, boolean append)
```

Or use configuration file

Can have

- Multiple handlers

- Multiple handlers of same type

Loggers and handlers have differ log levels

Logger

- Drops all messages below it log level

- Passes remaining messages to all handlers

- Handler can further drop more messages

Logger names are arbitrary

```
Logger.getLogger("edu.sdsu.cs580")
```

```
Logger.getLogger("foo")
```

```
Logger.getLogger("")
```

Sun recommends using hierarchical names with format

```
"domain.package"
```

```
"domain.package.class"
```

Loggers inherit settings from “parent” logger

Logger "edu.sdsu.cs580" would inherit settings of "edu.sdsu"

Logger settings can be defined in
Program
Configuration File

Logger settings defined in a program exist only in that program

Logger settings defined in a configuration file can be used by multiple programs


```
# Use two loggers
handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler

# Default global logging level.
.level= WARNING

# File logger default settings
# Default file output is in user's home directory (%h/).
# %g – use generation numbers to distinguish rotated logs
# limit = max size of each log file
# count = number of output files to cycle through
java.util.logging.FileHandler.pattern = %h/cs580Server%g.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 3
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter

# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter

# Set levels of specific loggers
edu.sdsu.level = SEVERE
edu.sdsu.cs580.level = INFO
```

Assume that configuration file is in
Local directory
In a file called cs580Log.properties

The following command will use the configuration file

```
java -Djava.util.logging.config.file=cs580Log.properties yourClassGoesHere
```

```
require 'logger'

class LogExample
  @@logger = nil
  def self.log
    return @@logger if !@@logger.nil?
    @@logger = Logger.new('foo.log', 5, 1024000)
    @@logger.level = Logger::WARN
    @@logger
  end

  def example
    LogExample.log.debug('me')
    LogExample.log.info('some info')
    LogExample.log.warn('a warning')
    LogExample.log.error('an error')
    LogExample.log.fatal('death')

    LogExample.log.warn {"Argument 'foo' not given"}
    LogExample.log.warn "Argument #{@foo} not given"
    LogExample.log.warn( caller(0).first) {"dog"}
  end
end
```

```
# Logfile created on Tue Feb 07 11:06:39 PST 2006 by logger.rb/1.5.2.4
W, [2006-02-07T11:06:39.035087 #26488] WARN -- : a warning
E, [2006-02-07T11:06:39.035371 #26488] ERROR -- : an error
F, [2006-02-07T11:06:39.036559 #26488] FATAL -- : death
W, [2006-02-07T11:06:39.036807 #26488] WARN -- : Argument 'foo' not given
W, [2006-02-07T11:06:39.036938 #26488] WARN -- : Argument not given
W, [2006-02-07T11:06:39.037082 #26488] WARN -- /Users/whitney/Courses/580/Spring06/examples/
logging/LoggingExample.rb:21:in `example': dog
```