sep 10, 15 3:52 **UrlLogger.java** Page 1/1

```java
package Monitor;

import java.io.UnsupportedEncodingException;
import java.util.concurrent.BlockingQueue;
import java.io.FileNotFoundException;
import java.io.PrintWriter;

/**
 * Created by adrian on 10/09/15.
 */
public class UrlLogger implements Runnable {

    private BlockingQueue<String> urlQueue;
    private PrintWriter writer;

    public UrlLogger(BlockingQueue<String> urlQueue) {
        this.urlQueue = urlQueue;
    }

    public void initializeLogger (String fileName) {
        try {
            writer = new PrintWriter(fileName, "UTF-8");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        while (¬Thread.interrupted()) {
            try {
                String url = urlQueue.take();
                writer.println(url);
                writer.flush();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }


        }
    }
}
```

sep 09, 15 21:15 **UrlIncreaseMessage.java** Page 1/1

```java
package Monitor;

/**
 * Created by adrian on 09/09/15.
 */
public class UrlIncreaseMessage implements MonitorMessage {
    public void updateMonitor(MonitorStats monitor) {
        monitor.increaseAnalyzedUrl();
    }
}
```

sep 10, 15 14:08 **ThreadUpdateMessage.java** Page 1/1

```java
package Monitor;

import Crawler.ThreadState;

/**
 * Created by adrian on 09/09/15.
 */
public class ThreadUpdateMessage implements MonitorMessage {

    private ThreadState state;

    public ThreadUpdateMessage (ThreadState state) {
        this.state = state;
    }

    public void updateMonitor(Monitor.MonitorStats monitor) {
        monitor.setStatus(this.state);
    }
}
```

sep 10, 15 14:08 **MonitorWriter.java** Page 1/1

```java
package Monitor;

import java.io.*;
import java.util.Date;

/**
 * Created by adrian on 09/09/15.
 */
public class MonitorWriter implements Runnable {

    private MonitorStats stats;

    private PrintWriter writer;
    private Integer logInterval;

    private Date flushDate;
    private Integer flushInterval;

    public MonitorWriter(MonitorStats stats) {
        this.stats = stats;
    }

    public void initializeMonitor (String fileName, Integer logInterval, Integer flushInterval) {
        flushDate = new Date();
        this.logInterval = logInterval;
        this.flushInterval = flushInterval;
        try {
            writer = new PrintWriter(fileName, "UTF-8");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        while (¬Thread.interrupted()) {
            String statsString = stats.getFormattedStats();
            writer.println(statsString);

            if (((new Date().getTime() - this.flushDate.getTime()) / 1000) > flushInterval) {
                writer.flush();
                this.flushDate = new Date();
            }

            try {
                Thread.sleep((long) logInterval * 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

sep 10, 15 14:08 **MonitorStats.java** Page 1/3

```java
package Monitor;

import Crawler.ThreadState;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.*;

/**
 * Created by adrian on 09/09/15.
 */
public class MonitorStats {

    private Integer urlsCount;

    private HashMap<String, Integer> fileTypes;

    private HashMap<Integer, ThreadState.Status> urlAnalyzerState;
    private HashMap<Integer, ThreadState.Status> fileAnalyzerState;
    private HashMap<Integer, ThreadState.Status> fileDownloaderState;

    public MonitorStats() {
        urlsCount = 0;
        fileTypes = new HashMap<String, Integer>();
        urlAnalyzerState = new HashMap<Integer, ThreadState.Status>();
        fileAnalyzerState = new HashMap<Integer, ThreadState.Status>();
        fileDownloaderState = new HashMap<Integer, ThreadState.Status>();
    }

    public synchronized void initMonitorStats (Integer urlAnalyzers, Integer fil
eAnalyzers, Integer fileDownloaders) {
        // I synchronize this block to be consistent, in that every public metho
d should be atomic, but it's not necessary
        this.initializeStats(fileAnalyzerState, fileAnalyzers);
        this.initializeStats(fileDownloaderState, fileDownloaders);
        this.initializeStats(urlAnalyzerState, urlAnalyzers);

        urlsCount = 0;
    }

    private void initializeStats(HashMap<Integer, ThreadState.Status> map, Integ
er amount) {
        for (Integer i = 0; i < amount; ++i) {
            map.put(i, ThreadState.Status.UNKNOWN);
        }
    }

    public synchronized void setStatus(ThreadState state) {
            HashMap<Integer, ThreadState.Status> map = null;
            switch (state.getThreadName()) {
                case URL_ANALYZER:
                    map = urlAnalyzerState;
                    break;
                case HTML_ANALYZER:
                    map = fileAnalyzerState;
                    break;
                case FILE_DOWNLOADER:
                    map = fileDownloaderState;
                    break;
            }
            map.put(state.getThreadId(), state.getThreadStatus());

    }

    public synchronized void increaseAnalyzedUrl() {
        urlsCount++;
```

sep 10, 15 14:08 **MonitorStats.java** Page 2/3

```java
    }

    public synchronized void increaseFileType(String fileType) {
        if (fileTypes.containsKey(fileType)) {
            fileTypes.put(fileType, fileTypes.get(fileType) + 1);
        } else {
            fileTypes.put(fileType, 1);
        }
    }

    public synchronized String getFormattedStats () {
        String result;
            DateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
            String stringDate = formatter.format(new Date());

            result = stringDate + " - Analyzed URLS: " + urlsCount + "\n";
            result = result + "Downloaded files by type: \n" + this.getFileTypesStats()
+ "\n";
            result = result + this.getThreadStats("Analyzing URL",urlAnalyzerState
);
            result = result + this.getThreadStats("Downloading File", fileDownloade
rState);
            result = result + this.getThreadStats("Analyzing File",fileAnalyzerState
);

        return result;
    }

    private String getFileTypesStats () {
        String filesStats = "";
        for (Map.Entry pair : fileTypes.entrySet()) {
            filesStats = filesStats + "\t" + pair.getKey() + ":" + pair.getValue(
);
        }
        return filesStats;
    }

    private String getThreadStats (String threadName, HashMap<Integer, ThreadSta
te.Status> map) {
        String threadStats;
        Integer working = 0;
        Integer blocked = 0;
        Integer unknown = 0;
        Integer starting = 0;

        for (Map.Entry pair : map.entrySet()) {
            ThreadState.Status status = (ThreadState.Status) pair.getValue();
            switch (status) {
                case UNKNOWN:
                    unknown++;
                    break;
                case STARTING:
                    starting++;
                    break;
                case BLOCKED:
                    blocked++;
                    break;
                case WORKING:
                    working++;
                    break;
            }
        }

        threadStats = "\t" + threadName + "\n\t\t";
        threadStats = threadStats + "Unknown: " + unknown;
        threadStats = threadStats + " Starting: " + starting;
```

sep 10, 15 14:08 **MonitorStats.java** Page 3/3

```
124             threadStats = threadStats + " Blocked: " + blocked;
125             threadStats = threadStats + " Working: " + working + "\n";
126             return threadStats;
127         }
128     }
```

sep 09, 15 21:28 **MonitorMessage.java** Page 1/1

```
1   package Monitor;
2
3   /**
4    * Created by adrian on 09/09/15.
5    */
6   public interface MonitorMessage {
7
8       public void updateMonitor(MonitorStats monitor);
9   }
```

sep 10, 15 14:08 **MonitorFetcher.java** Page 1/1

```java
package Monitor;

import java.util.concurrent.BlockingQueue;

/**
 * Created by adrian on 09/09/15.
 */
public class MonitorFetcher implements Runnable {

    private MonitorStats stats;
    private BlockingQueue<MonitorMessage> monitorQueue;

    public MonitorFetcher(MonitorStats stats, BlockingQueue<MonitorMessage> monitorQueue) {
        this.stats = stats;
        this.monitorQueue = monitorQueue;
    }

    public void run() {
        while (¬Thread.interrupted()) {
            try {
                MonitorMessage message = monitorQueue.take();
                message.updateMonitor(stats);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }
    }
}
```

sep 10, 15 14:08 **FileTypeUpdateMessage.java** Page 1/1

```java
package Monitor;

/**
 * Created by adrian on 09/09/15.
 */
public class FileTypeUpdateMessage implements MonitorMessage {

    private String fileType;

    public FileTypeUpdateMessage(String type) {
        this.fileType = type;
    }

    public void updateMonitor(MonitorStats monitor) {
        monitor.increaseFileType(this.fileType);
    }
}
```

sep 10, 15 14:08 **Launcher.java** Page 1/3

```
1  import Crawler.*;
2  import Monitor.*;
3
4  import java.io.*;
5  import java.net.URI;
6  import java.net.URISyntaxException;
7  import java.util.Properties;
8  import java.util.concurrent.*;
9
10 /**
11  * Created by adrian on 05/09/15.
12  */
13 public class Launcher {
14
15     public static void main(String[] args) {
16
17         Properties prop = new Properties();
18         InputStream input = null;
19
20         try {
21             String configFile = "Config/Config.properties";
22             File f = new File(configFile);
23             if(¬f.exists()) {
24                 System.err.println("Config file " + configFile + " missing");
25                 System.err.println("Format:\n" +
26                         "####Config file start####\n" +
27                         "iterations = #\n"+
28                         "fileAnalyzer = #\n"+
29                         "fileDownloader = #\n"+
30                         "urlAnalyzer = #\n"+
31                         "queueSize = #\n"+
32                         "urlLogFile = nameUrlLogFile.log\n"+
33                         "logFile = nameMonitorLogFile.log\n"+
34                         "logInterval = #\n"+
35                         "flushInterval = #\n"+
36                         "####Config file end####");
37                 System.exit(-1);
38             }
39             input = new FileInputStream(configFile);
40
41             prop.load(input);
42
43             int queueSize =  Integer.parseInt(prop.getProperty("queueSize"));
44
45             int numberOfFileAnalyzer = Integer.parseInt(prop.getProperty("fileAnal
   yzer"));
46             int numberOfUrlAnalyzer = Integer.parseInt(prop.getProperty("fileDownl
   oader"));
47             int numberOfFileDownloader = Integer.parseInt(prop.getProperty("urlAn
   alyzer"));
48
49             // Monitor
50             MonitorStats stats = new MonitorStats();
51             stats.initMonitorStats(numberOfUrlAnalyzer, numberOfFileAnalyzer, nu
   mberOfFileDownloader);
52
53             BlockingQueue<MonitorMessage> monitorQueue = new ArrayBlockingQueue<
   MonitorMessage>(queueSize);
54             MonitorWriter writer = new MonitorWriter(stats);
55             writer.initializeMonitor(prop.getProperty("logFile"), Integer.parseInt
   (prop.getProperty("logInterval")),Integer.parseInt(prop.getProperty("flushInterval")));
56             (new Thread(writer)).start();
57             MonitorFetcher fetcher = new MonitorFetcher(stats, monitorQueue);
58             (new Thread(fetcher)).start();
59
60             // Url Logger
```

sep 10, 15 14:08 **Launcher.java** Page 2/3

```
61             BlockingQueue<String> urlLoggerQueue = new ArrayBlockingQueue<String
   >(queueSize);
62             UrlLogger urlLogger = new UrlLogger(urlLoggerQueue);
63             urlLogger.initializeLogger(prop.getProperty("urlLogFile"));
64             new Thread(urlLogger).start();
65
66             // Crawler Queues
67             BlockingQueue<UrlMessage> urlToAnalyzeQueue = new ArrayBlockingQueue
   <UrlMessage>(queueSize);
68             BlockingQueue<Pair<String, UrlMessage>> fileToAnalyzedQueue = new Ar
   rayBlockingQueue<Pair<String, UrlMessage>>(queueSize);
69             BlockingQueue<UrlMessage> urlToDownloadQueue = new ArrayBlockingQueu
   e<UrlMessage>(queueSize);
70
71             ConcurrentHashMap<String,Boolean> hashMap = new ConcurrentHashMap<St
   ring, Boolean>();
72
73             // Crawler workers
74             for (int i = 0; i < numberOfFileAnalyzer; ++i) {
75                 (new Thread(new FileAnalyzer(i, fileToAnalyzedQueue, urlToAnalyz
   eQueue, monitorQueue))).start();
76             }
77
78             int maxIterations = Integer.parseInt(prop.getProperty("iterations"));
79             for (int i = 0; i < numberOfUrlAnalyzer; ++i) {
80                 (new Thread(new FileDownloader(i, maxIterations, urlToDownloadQu
   eue,fileToAnalyzedQueue, monitorQueue))).start();
81             }
82
83             for (int i = 0; i < numberOfFileDownloader; ++i) {
84                 (new Thread(new UrlAnalyzer(i, urlToAnalyzeQueue,urlToDownloadQu
   eue,hashMap, monitorQueue, urlLoggerQueue))).start();
85             }
86
87
88             while(¬ Thread.interrupted()) {
89                 BufferedReader br = new BufferedReader(new InputStreamReader(Sys
   tem.in));
90                 System.out.print("Enter Url:");
91                 String src = br.readLine();
92
93                 try {
94                     URI u = new URI(src);
95                     if (u.isAbsolute()) {
96                         urlToAnalyzeQueue.put(new UrlMessage(0,src));
97                     }
98                 } catch (URISyntaxException e) {
99                     e.printStackTrace();
100                }
101
102            }
103
104            System.out.println("Finishing web crawler");
105
106        } catch (IOException ex) {
107            ex.printStackTrace();
108        } catch (InterruptedException e) {
109            e.printStackTrace();
110        } finally {
111            if (input ≠ null) {
112                try {
113                    input.close();
114                } catch (IOException e) {
115                    e.printStackTrace();
116                }
117            }
```

sep 10, 15 14:08 **Launcher.java** Page 3/3

```
118             }
119         }
120     }
```

sep 10, 15 2:19 **UrlMessage.java** Page 1/1

```
1   package Crawler;
2
3   /**
4    * Created by adrian on 10/09/15.
5    */
6   public class UrlMessage {
7
8       private Integer iteration;
9       private String url;
10
11      public UrlMessage (Integer iteration, String url) {
12          this.iteration = iteration;
13          this.url = url;
14      }
15
16      public Integer getIteration() {
17          return iteration;
18      }
19
20      public String getUrl() {
21          return url;
22      }
23  }
```

sep 10, 15 14:08 **UrlAnalyzer.java** Page 1/2

```
1  package Crawler;
2
3  import Monitor.MonitorMessage;
4  import Monitor.ThreadUpdateMessage;
5  import Monitor.UrlIncreaseMessage;
6
7  import java.util.concurrent.BlockingQueue;
8  import java.util.concurrent.ConcurrentHashMap;
9
10 /**
11  * Created by adrian on 05/09/15.
12  */
13
14 public class UrlAnalyzer implements Runnable {
15
16     private Integer threadId;
17     private BlockingQueue<MonitorMessage> monitorQueue;
18     private BlockingQueue<String> urlLoggerQueue;
19
20     private BlockingQueue<UrlMessage> urlToAnalyzeQueue;
21     private BlockingQueue<UrlMessage> urlToDownloadQueue;
22
23     private ConcurrentHashMap<String,Boolean> analyzedUrls;
24
25     public UrlAnalyzer(Integer threadId, BlockingQueue<UrlMessage> analyzeQueue,
    BlockingQueue<UrlMessage> downloadQueue, ConcurrentHashMap<String,Boolean> map,
    BlockingQueue<MonitorMessage> monitorQueue, BlockingQueue<String> urlLoggerQueu
   e) {
26         this.threadId = threadId;
27         this.monitorQueue = monitorQueue;
28         this.urlLoggerQueue = urlLoggerQueue;
29         this.urlToAnalyzeQueue = analyzeQueue;
30         this.urlToDownloadQueue = downloadQueue;
31         this.analyzedUrls = map;
32     }
33
34     public void run() {
35
36         try {
37             this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Thread
   State.Type.URL_ANALYZER, this.threadId, ThreadState.Status.STARTING)));
38         } catch (InterruptedException e) {
39             e.printStackTrace();
40         }
41
42         while (¬Thread.interrupted()) {
43             try {
44                 this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Th
   readState.Type.URL_ANALYZER, this.threadId, ThreadState.Status.BLOCKED)));
45                 UrlMessage url = urlToAnalyzeQueue.take();
46                 this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Th
   readState.Type.URL_ANALYZER, this.threadId, ThreadState.Status.WORKING)));
47
48                 System.out.println("Analyzing url: "+url.getUrl());
49                     if (analyzedUrls.putIfAbsent(url.getUrl(), true) ≡ null) {
50                         System.out.println("Url NOT downloaded " + url.getUrl());
51                         this.urlLoggerQueue.put(url.getUrl());
52                         urlToDownloadQueue.put(url);
53                     } else {
54                         // Url already downloaded
55                         System.out.println("Url already downloaded " + url.getUrl());
56                     }
57
58
59                 this.monitorQueue.put(new UrlIncreaseMessage());
60
```

sep 10, 15 14:08 **UrlAnalyzer.java** Page 2/2

```
61             } catch (InterruptedException e) {
62                 e.printStackTrace();
63             }
64
65         }
66     }
67 }
```

sep 10, 15 1:03 **ThreadState.java** Page 1/1

```java
package Crawler;

/**
 * Created by adrian on 08/09/15.
 */

public class ThreadState {

    private Type threadName;
    private Integer threadId;
    private Status threadStatus;

    public enum Status {
        UNKNOWN,
        STARTING,
        BLOCKED,
        WORKING
    }

    public enum Type {
        URL_ANALYZER,
        HTML_ANALYZER,
        FILE_DOWNLOADER
    }

    public ThreadState(Type name, int id, Status status) {
        threadName = name;
        threadId = id;
        threadStatus = status;
    }

    public Type getThreadName() {
        return threadName;
    }

    public int getThreadId() {
        return threadId;
    }

    public Status getThreadStatus() {
        return threadStatus;
    }
}
```

sep 10, 15 14:08 **Pair.java** Page 1/1

```java
package Crawler;

/**
 * Created by adrian on 06/09/15.
 */
public class Pair<A, B> {
    private A first;
    private B second;

    public Pair(A first, B second) {
        super();
        this.first = first;
        this.second = second;
    }

    public A getFirst() {
        return first;
    }

    public B getSecond() {
        return second;
    }
}
```

sep 10, 15 14:08 **FileDownloader.java** Page 1/3

```java
package Crawler;

import Monitor.FileTypeUpdateMessage;
import Monitor.MonitorMessage;
import Monitor.ThreadUpdateMessage;

import java.io.*;
import java.net.*;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.util.concurrent.BlockingQueue;

public class FileDownloader implements Runnable {

    private Integer threadId;
    private Integer iterations;
    private BlockingQueue<MonitorMessage> monitorQueue;
    private BlockingQueue<UrlMessage> urlToDownloadQueue;
    private BlockingQueue<Pair<String, UrlMessage>> fileToAnalyzeQueue;

    public FileDownloader(Integer threadId, Integer iterations, BlockingQueue<UrlMessage> downloadQueue, BlockingQueue<Pair<String, UrlMessage>> fileAnalyzeQueue, BlockingQueue<MonitorMessage> monitorQueue) {
        this.threadId = threadId;
        this.iterations = iterations;
        this.monitorQueue = monitorQueue;
        this.urlToDownloadQueue = downloadQueue;
        this.fileToAnalyzeQueue = fileAnalyzeQueue;
    }

    public void run() {
        try {
            this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(ThreadState.Type.FILE_DOWNLOADER, this.threadId, ThreadState.Status.STARTING)));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        while (¬Thread.interrupted()) {
            try {
                this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(ThreadState.Type.FILE_DOWNLOADER, this.threadId, ThreadState.Status.BLOCKED)));
                UrlMessage url = urlToDownloadQueue.take();
                this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(ThreadState.Type.FILE_DOWNLOADER, this.threadId, ThreadState.Status.WORKING)));
                downloadFile(url, "Downloads");
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    private void downloadFile(UrlMessage message, String destinationDir) throws InterruptedException {
        String localFileName;
        int slashIndex = message.getUrl().lastIndexOf('/');
        // We should remode tha last / from the url because we use the string next to the slash to use as file name
        if (slashIndex ≡ message.getUrl().length() − 1) {
            String noLastSlashUrl = message.getUrl().substring(0,message.getUrl().length() − 1);
            slashIndex = noLastSlashUrl.lastIndexOf('/');
            localFileName = noLastSlashUrl.substring(slashIndex + 1);
        } else {
            localFileName = message.getUrl().substring(slashIndex + 1);
        }
```

sep 10, 15 14:08 **FileDownloader.java** Page 2/3

```java


        if (localFileName.length() > 0) {
            File dir = new File(destinationDir);

            // We create the download directory if it doesn't exists
            if (¬dir.exists()) {
                dir.mkdirs();
            }

            OutputStream outStream = null;
            URLConnection connection;
            InputStream inStream = null;
            try {
                URL url;
                byte[] buf;
                int byteRead;
                url = new URL(message.getUrl());
                connection = url.openConnection();

                outStream = new BufferedOutputStream(new FileOutputStream(destinationDir + "/" + localFileName));
                inStream = connection.getInputStream();

                buf = new byte[1024];
                while ((byteRead = inStream.read(buf)) ≠ −1) {
                    outStream.write(buf, 0, byteRead);
                }

                String fileType = Files.probeContentType(FileSystems.getDefault().getPath(destinationDir, localFileName));
                if ((message.getIteration() < this.iterations−1) ∧ fileType.equals("text/html")) {
                    fileToAnalyzeQueue.put(new Pair<String, UrlMessage>(destinationDir + "/" + localFileName, new UrlMessage(message.getIteration()+1,message.getUrl()) ));
                }

                this.monitorQueue.put(new FileTypeUpdateMessage(fileType));

            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {

                if (inStream ≠ null) {
                    try {
                        inStream.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                if (outStream ≠ null) {
                    try {
                        outStream.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
```

sep 10, 15 14:08 **FileDownloader.java** Page 3/3

```
120 }
```

sep 10, 15 14:04 **FileAnalyzer.java** Page 1/2

```
1  package Crawler;
2
3  import Monitor.MonitorMessage;
4  import Monitor.ThreadUpdateMessage;
5
6  import javax.swing.text.AttributeSet;
7  import javax.swing.text.html.HTML;
8  import javax.swing.text.html.HTMLDocument;
9  import javax.swing.text.html.HTMLEditorKit;
10 import javax.swing.text.html.parser.ParserDelegator;
11
12 import java.io.*;
13 import java.net.URI;
14 import java.net.URISyntaxException;
15
16 import java.util.concurrent.BlockingQueue;
17
18
19 /**
20  * Created by adrian on 04/09/15.
21  */
22 public class FileAnalyzer implements Runnable {
23
24     private Integer threadId;
25     private BlockingQueue<MonitorMessage> monitorQueue;
26     private BlockingQueue<UrlMessage> urlToAnalyzeQueue;
27     private BlockingQueue<Pair<String, UrlMessage>> fileToAnalyzedQueue;
28
29     public FileAnalyzer(Integer threadId, BlockingQueue<Pair<String, UrlMessage>
   > fileQueue, BlockingQueue<UrlMessage> urlQueue, BlockingQueue<MonitorMessage> m
   onitorQueue) {
30         this.threadId = threadId;
31         this.monitorQueue = monitorQueue;
32         this.urlToAnalyzeQueue = urlQueue;
33         this.fileToAnalyzedQueue = fileQueue;
34     }
35
36     public void run() {
37         try {
38             this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Thread
   State.Type.HTML_ANALYZER, this.threadId, ThreadState.Status.STARTING)));
39         } catch (InterruptedException e) {
40             e.printStackTrace();
41         }
42
43         while (¬Thread.interrupted()) {
44             try {
45                 this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Th
   readState.Type.HTML_ANALYZER, this.threadId, ThreadState.Status.BLOCKED)));
46                 Pair<String, UrlMessage> file = fileToAnalyzedQueue.take();
47                 this.monitorQueue.put(new ThreadUpdateMessage(new ThreadState(Th
   readState.Type.HTML_ANALYZER, this.threadId, ThreadState.Status.WORKING)));
48                 System.out.println("Analyzing file: "+file.getFirst());
49                 analyzeFile(file.getFirst(), file.getSecond());
50             } catch (InterruptedException e) {
51                 e.printStackTrace();
52             }
53         }
54     }
55
56     public void analyzeFile (String fileName, UrlMessage url) throws Interrupted
   Exception {
57         InputStream is = null;
58         try {
59             is = new FileInputStream(fileName);
60         } catch (FileNotFoundException e) {
```

sep 10, 15 14:04 **FileAnalyzer.java** Page 2/2

```
61              e.printStackTrace();
62          }
63
64          InputStreamReader isr = new InputStreamReader(is);
65          BufferedReader br = new BufferedReader(isr);
66
67          HTMLEditorKit htmlKit = new HTMLEditorKit();
68          HTMLDocument htmlDoc = (HTMLDocument) htmlKit.createDefaultDocument();
69          HTMLEditorKit.Parser parser = new ParserDelegator();
70          HTMLEditorKit.ParserCallback callback = htmlDoc.getReader(0);
71
72          try {
73              parser.parse(br, callback, true);
74          } catch (IOException e) {
75              e.printStackTrace();
76          }
77
78          getResource(url, htmlDoc, HTML.Tag.IMG, HTML.Attribute.SRC);
79          getResource(url, htmlDoc, HTML.Tag.A, HTML.Attribute.HREF);
80          getResource(url, htmlDoc, HTML.Tag.LINK, HTML.Attribute.HREF);
81          getResource(url, htmlDoc, HTML.Tag.SCRIPT, HTML.Attribute.SRC);
82      }
83
84      private void getResource(UrlMessage url, HTMLDocument htmlDoc, HTML.Tag tag,
     HTML.Attribute attribute) throws InterruptedException {
85          for (HTMLDocument.Iterator iterator = htmlDoc.getIterator(tag); iterator
    .isValid(); iterator.next()) {
86              AttributeSet attributes = iterator.getAttributes();
87              String src = (String) attributes.getAttribute(attribute);
88
89              if (src ≠ null) {
90
91                  URI u = null;
92                  try {
93                      u = new URI(src);
94                      String finalUrl;
95                      if (u.isAbsolute()) {
96                          finalUrl = src;
97                      } else {
98                          if (src.startsWith("//")) {
99                              int index = url.getUrl().indexOf(":");
100                             String protocol = url.getUrl().substring(0,index);
101                             finalUrl = protocol + ":" + src;
102                         } else {
103                             finalUrl = u.resolve(url.getUrl()).normalize().toStr
    ing();
104                         }
105                     }
106
107                     urlToAnalyzeQueue.put(new UrlMessage(url.getIteration(), fin
    alUrl));
108                 } catch (URISyntaxException e) {
109                       e.printStackTrace();
110                 }
111             }
112         }
113     }
114 }
```

sep 10, 15 14:24 **Table of Content** Page 1/1