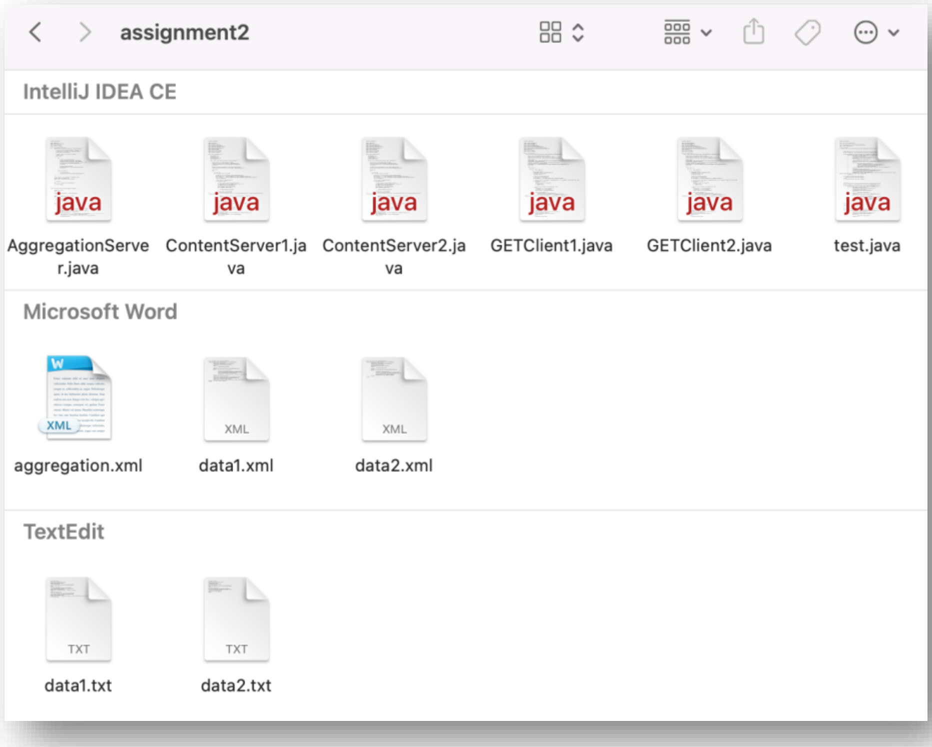
README

Introduction

My assignment2 has 6 java files, 2 txt files and 3 xml files.

The path is src/assignment2/



I created two **ContentServer** and two **GETClient** for running multi-clients.

ContentServer1 input the source file "data1.txt" to parse to "data1.xml", and then send the content to **AggregationServer**

ContentServer2 input the source file "data2.txt" to parse to "data2.xml", and then send the content to **AggregationServer**

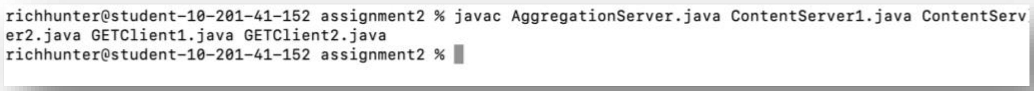
AggregationServer will put all received message from **ContentServer** to "aggregation.xml", and then send to **GETClient** if requested.

GETClient1 send "GET" request to **AggregationServer** to get message.

GETClient2 send "GET" request to **AggregationServer** to get message

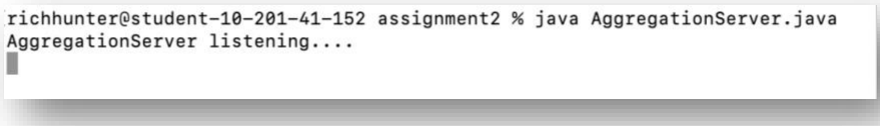
1. Open a terminal to compile all java files first

javac AggregationServer.java ContentServer1.java ContentServer2.java GETClient1.java GETClient2.java



```
richhunter@student-10-201-41-152 assignment2 % javac AggregationServer.java ContentServer1.java ContentServ
er2.java GETClient1.java GETClient2.java
richhunter@student-10-201-41-152 assignment2 % 
```

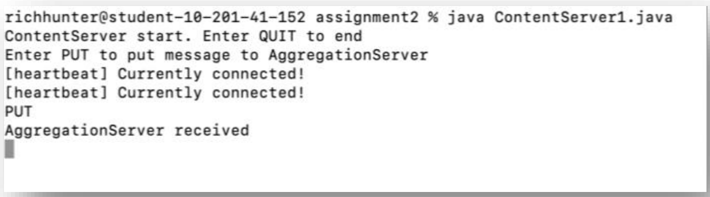
Start “AggregationServer.java” it will display AggregationServer listening means waiting for clients to connect



```
richhunter@student-10-201-41-152 assignment2 % java AggregationServer.java
AggregationServer listening....
```

2. Open another terminal to start ContentServer1.java

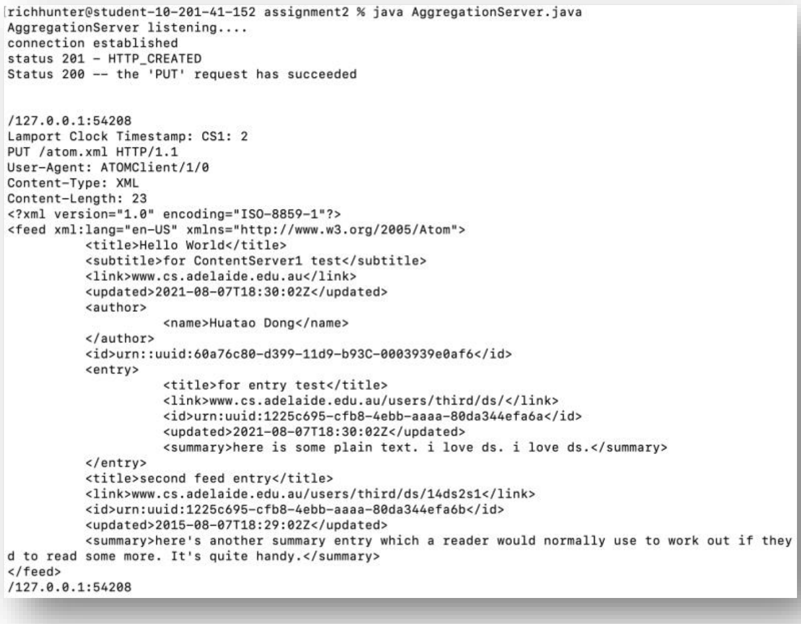
Enter “PUT” to put message to AggregationServer. The heartbeat will automatically send each 12 seconds. You can PUT infinitely and the AggregationServer will receive each PUT message.



```
richhunter@student-10-201-41-152 assignment2 % java ContentServer1.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
[heartbeat] Currently connected!
PUT
AggregationServer received
```

The terminal of AggregationServer will display status and xml message send by ContentServer1 with HTTP headers and Lamport clock timestamp.

The terminal of AggregationServer will receive the PUT message and aggregate all message to “aggregation.xml”. You can also open “aggregation.xml” to check.

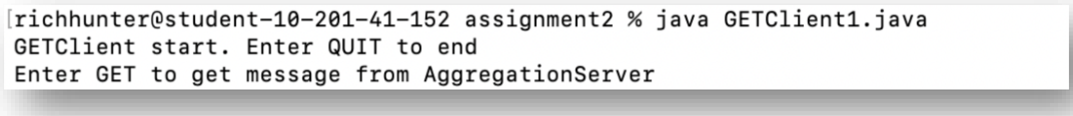


```
richhunter@student-10-201-41-152 assignment2 % java AggregationServer.java
AggregationServer listening....
connection established
status 201 - HTTP_CREATED
Status 200 -- the 'PUT' request has succeeded


/127.0.0.1:54208
Lamport Clock Timestamp: CS1: 2
PUT /atom.xml HTTP/1.1
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 23
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
          <title>Hello World</title>
          <subtitle>for ContentServer1 test</subtitle>
          <link>www.cs.adelaide.edu.au</link>
          <updated>2021-08-07T18:30:02Z</updated>
          <author>
                    <name>Huatao Dong</name>
          </author>
          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
          <entry>
                    <title>for entry test</title>
                    <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                    <updated>2021-08-07T18:30:02Z</updated>
                    <summary>here is some plain text. i love ds. i love ds.</summary>
          </entry>
          <title>second feed entry</title>
          <link>www.cs.adelaide.edu.au/users/third/ds/14ds2s1</link>
          <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b</id>
          <updated>2015-08-07T18:29:02Z</updated>
          <summary>here's another summary entry which a reader would normally use to work out if they
d to read some more. It's quite handy.</summary>
</feed>
/127.0.0.1:54208
```

The step of start ContentServer2.java is exactly same with ContentServer1.java

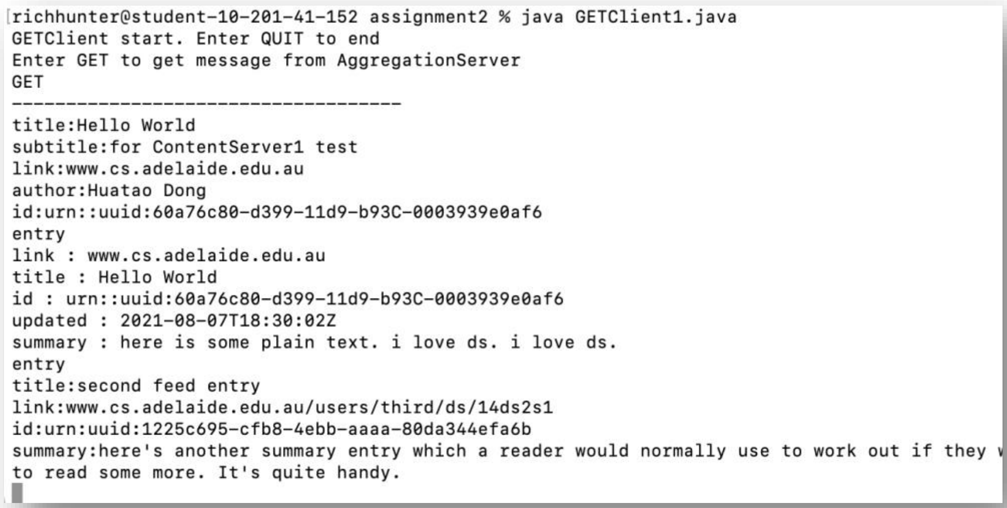
3. Open another terminal to start GETClient1.java

Enter “GET” to receive message from AggregationServer

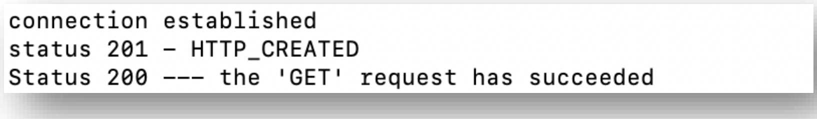

```
richhunter@student-10-201-41-152 assignment2 % java GETClient1.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
```

GETClient will receive the xml message from AggregationServer and automatically parse to txt type. It also can enter “GET” infinitely to receive message.

New ContentServer put or removed all message because of disconnection, the GETClient will also update by enter a new “GET”. It is synchronized. See more information at Test after.


```
richhunter@student-10-201-41-152 assignment2 % java GETClient1.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
GET
------------------------------------
title:Hello World
subtitle:for ContentServer1 test
link:www.cs.adelaide.edu.au
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.cs.adelaide.edu.au
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-08-07T18:30:02Z
summary : here is some plain text. i love ds. i love ds.
entry
title:second feed entry
link:www.cs.adelaide.edu.au/users/third/ds/14ds2s1
id:urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b
summary:here's another summary entry which a reader would normally use to work out if they
to read some more. It's quite handy.
```

After send “GET” request, the AggregationServer will display status feedback.


```
connection established
status 201 - HTTP_CREATED
Status 200 --- the 'GET' request has succeeded
```

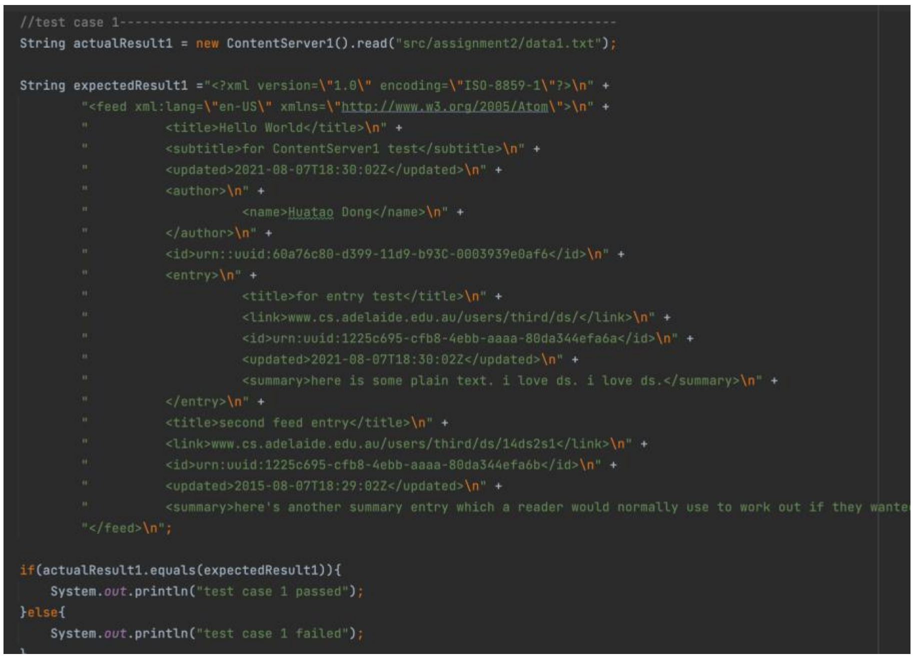
Here is the basic operation of PUT and GET of my program. See more information on test after.

Test

I created a test.java for testing xml parse to txt and txt parse to xml

Test case 1:

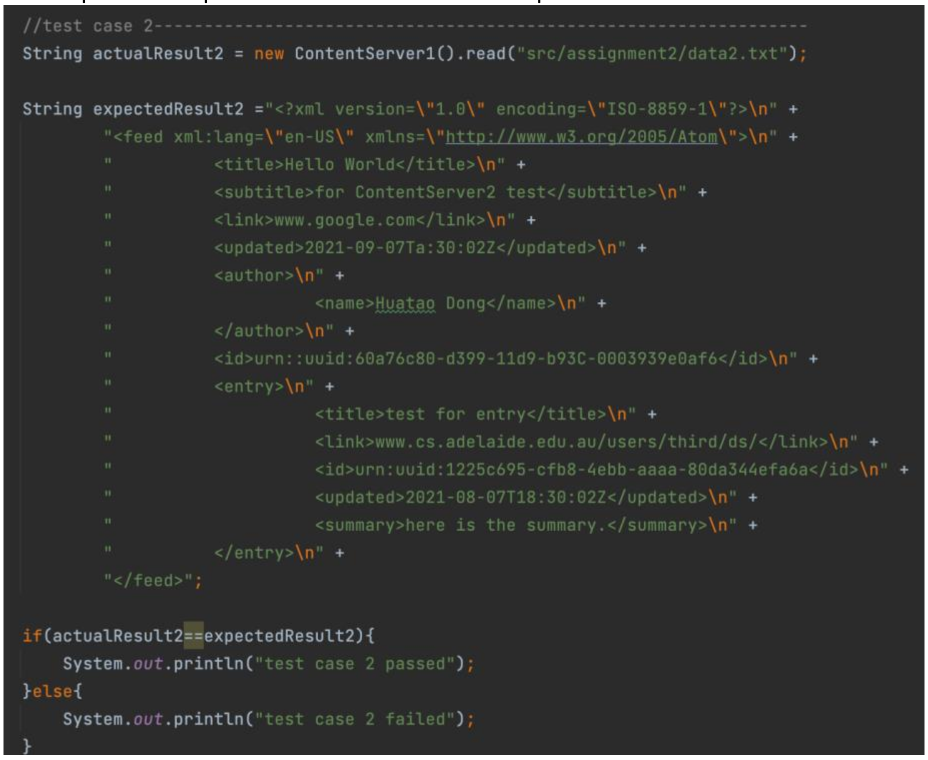
the method read() in ContentServer used for parse txt to xml
so compare the expected xml with actual xml output to test



```java
//test case 1-----------------------------------------------------------------
String actualResult1 = new ContentServer1().read("src/assignment2/data1.txt");

String expectedResult1 ="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n" +
        "<feed xml:lang=\"en-US\" xmlns=\"http://www.w3.org/2005/Atom\">\n" +
        "          <title>Hello World</title>\n" +
        "          <subtitle>for ContentServer1 test</subtitle>\n" +
        "          <updated>2021-08-07T18:30:02Z</updated>\n" +
        "          <author>\n" +
        "                    <name>Huatao Dong</name>\n" +
        "          </author>\n" +
        "          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>\n" +
        "          <entry>\n" +
        "                    <title>for entry test</title>\n" +
        "                    <link>www.cs.adelaide.edu.au/users/third/ds/</link>\n" +
        "                    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>\n" +
        "                    <updated>2021-08-07T18:30:02Z</updated>\n" +
        "                    <summary>here is some plain text. i love ds. i love ds.</summary>\n" +
        "          </entry>\n" +
        "          <title>second feed entry</title>\n" +
        "          <link>www.cs.adelaide.edu.au/users/third/ds/14ds2s1</link>\n" +
        "          <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b</id>\n" +
        "          <updated>2015-08-07T18:29:02Z</updated>\n" +
        "          <summary>here's another summary entry which a reader would normally use to work out if they wante
        "</feed>\n";

if(actualResult1.equals(expectedResult1)){
    System.out.println("test case 1 passed");
}else{
    System.out.println("test case 1 failed");
```

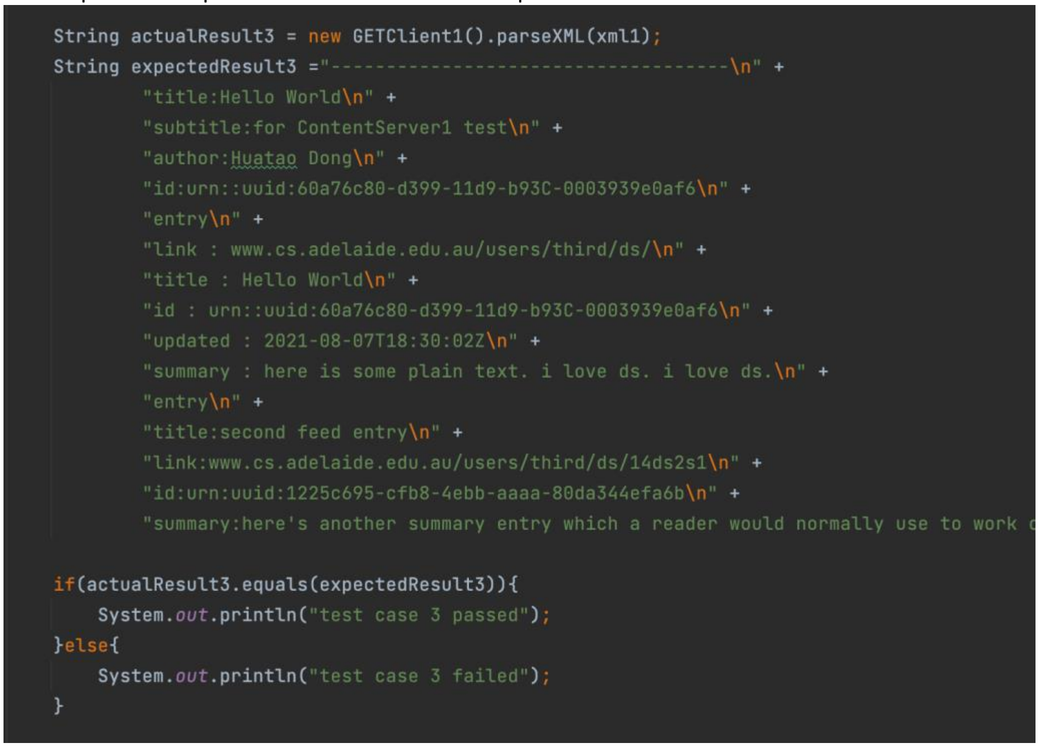
Test case 2:

the method read() in ContentServer used for parse txt to xml
so compare the expected xml with actual xml output to test



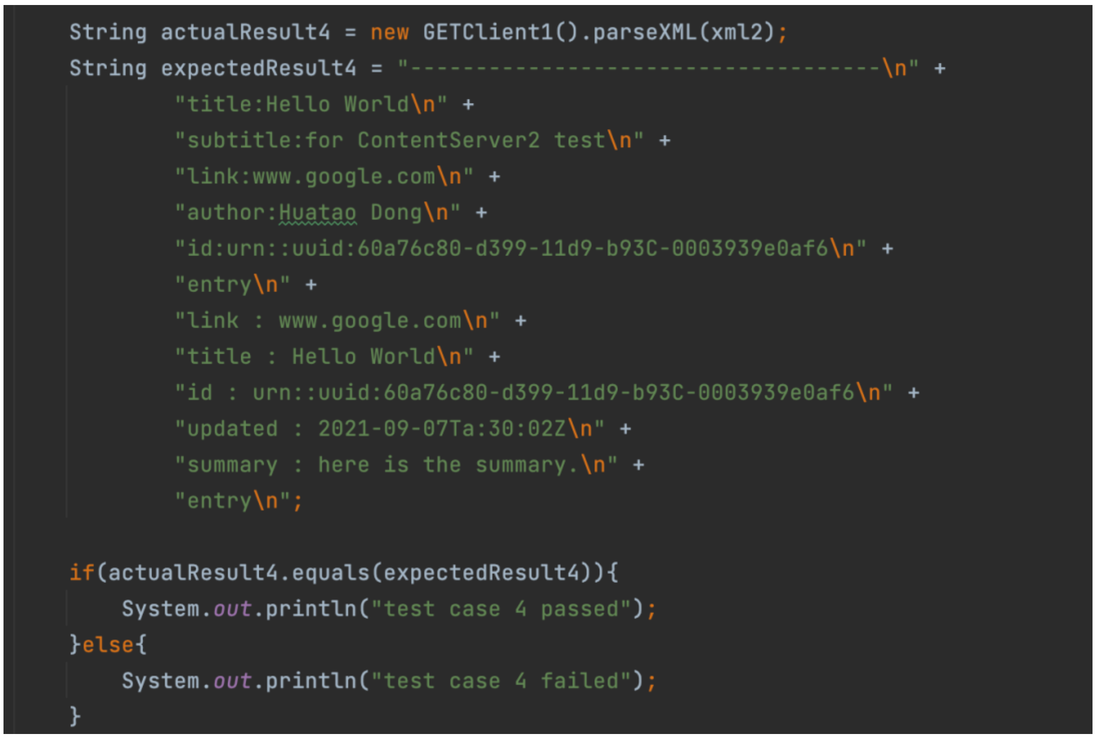
```java
//test case 2-----------------------------------------------------------------
String actualResult2 = new ContentServer1().read("src/assignment2/data2.txt");

String expectedResult2 ="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n" +
        "<feed xml:lang=\"en-US\" xmlns=\"http://www.w3.org/2005/Atom\">\n" +
        "          <title>Hello World</title>\n" +
        "          <subtitle>for ContentServer2 test</subtitle>\n" +
        "          <link>www.google.com</link>\n" +
        "          <updated>2021-09-07Ta:30:02Z</updated>\n" +
        "          <author>\n" +
        "                    <name>Huatao Dong</name>\n" +
        "          </author>\n" +
        "          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>\n" +
        "          <entry>\n" +
        "                    <title>test for entry</title>\n" +
        "                    <link>www.cs.adelaide.edu.au/users/third/ds/</link>\n" +
        "                    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>\n" +
        "                    <updated>2021-08-07T18:30:02Z</updated>\n" +
        "                    <summary>here is the summary.</summary>\n" +
        "          </entry>\n" +
        "</feed>";

if(actualResult2==expectedResult2){
    System.out.println("test case 2 passed");
}else{
    System.out.println("test case 2 failed");
}
```

Test case 3:
the method parseXML() in GETClient used for parse xml to txt
so compare the expected txt with actual txt output to test



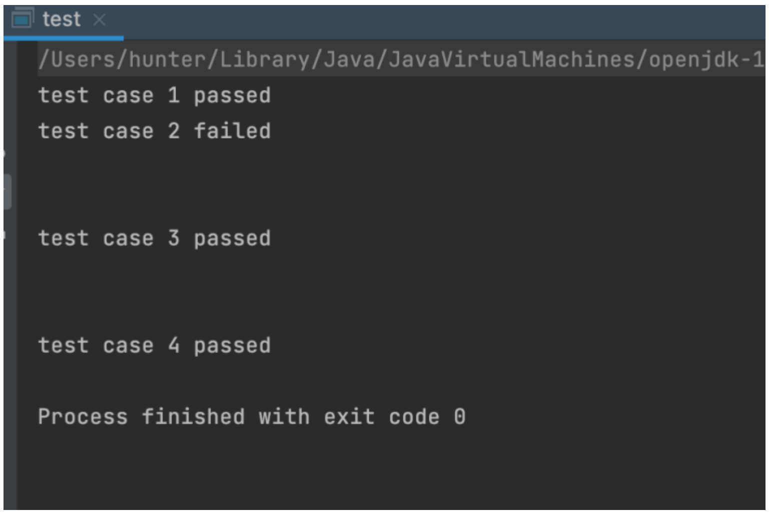
```
String actualResult3 = new GETClient1().parseXML(xml1);
String expectedResult3 ="----------------------------------\n" +
        "title:Hello World\n" +
        "subtitle:for ContentServer1 test\n" +
        "author:Huatao Dong\n" +
        "id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6\n" +
        "entry\n" +
        "link : www.cs.adelaide.edu.au/users/third/ds/\n" +
        "title : Hello World\n" +
        "id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6\n" +
        "updated : 2021-08-07T18:30:02Z\n" +
        "summary : here is some plain text. i love ds. i love ds.\n" +
        "entry\n" +
        "title:second feed entry\n" +
        "link:www.cs.adelaide.edu.au/users/third/ds/14ds2s1\n" +
        "id:urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b\n" +
        "summary:here's another summary entry which a reader would normally use to work

if(actualResult3.equals(expectedResult3)){
    System.out.println("test case 3 passed");
}else{
    System.out.println("test case 3 failed");
}
```

Test case 4:
the method parseXML() in GETClient used for parse xml to txt
so compare the expected txt with actual txt output to test



```
String actualResult4 = new GETClient1().parseXML(xml2);
String expectedResult4 = "------------------------------------\n" +
        "title:Hello World\n" +
        "subtitle:for ContentServer2 test\n" +
        "link:www.google.com\n" +
        "author:Huatao Dong\n" +
        "id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6\n" +
        "entry\n" +
        "link : www.google.com\n" +
        "title : Hello World\n" +
        "id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6\n" +
        "updated : 2021-09-07Ta:30:02Z\n" +
        "summary : here is the summary.\n" +
        "entry\n";

if(actualResult4.equals(expectedResult4)){
    System.out.println("test case 4 passed");
}else{
    System.out.println("test case 4 failed");
}
```

See the result of these four test cases. Run test.java to see.

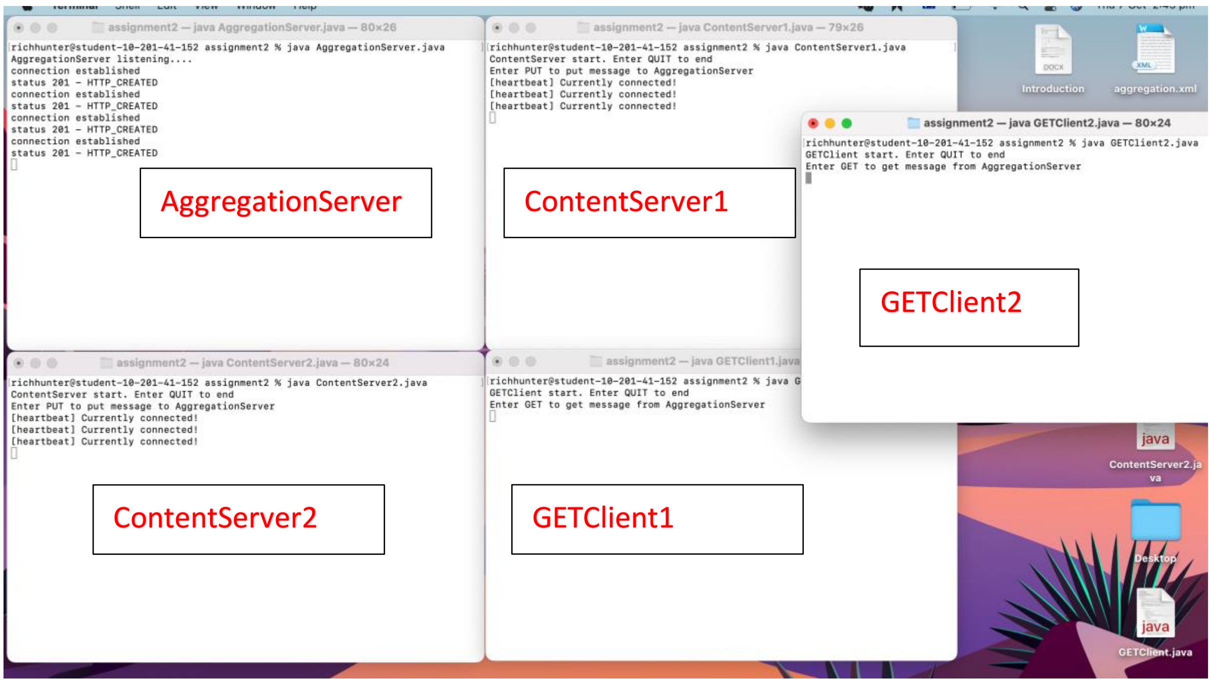


```
/Users/hunter/Library/Java/JavaVirtualMachines/openjdk-1
test case 1 passed
test case 2 failed


test case 3 passed


test case 4 passed

Process finished with exit code 0
```

Test case 5:

Test for if multiple GETClients and multiple ContentServers can connect and access to AggregationServer.

Open 5 Terminals and start these 5 java files. You can see all clients connected to AggregationServer by looking at the status in AggregationServer terminal.



```
richhunter@student-10-201-41-152 assignment2 % java AggregationServer.java
AggregationServer listening....
connection established
status 201 - HTTP_CREATED
connection established
status 201 - HTTP_CREATED
connection established
status 201 - HTTP_CREATED
connection established
status 201 - HTTP_CREATED
```

```
richhunter@student-10-201-41-152 assignment2 % java ContentServer1.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
```

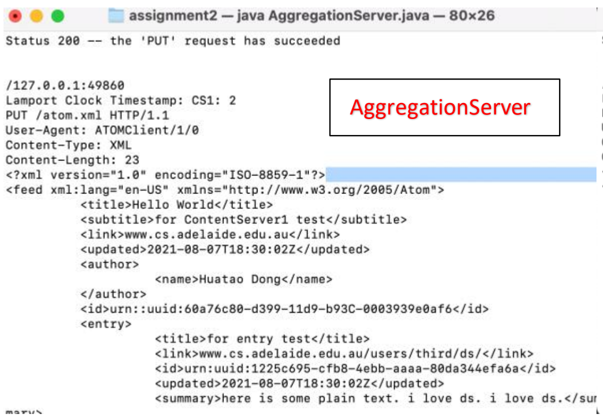
```
richhunter@student-10-201-41-152 assignment2 % java GETClient2.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
```

```
richhunter@student-10-201-41-152 assignment2 % java ContentServer2.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
```

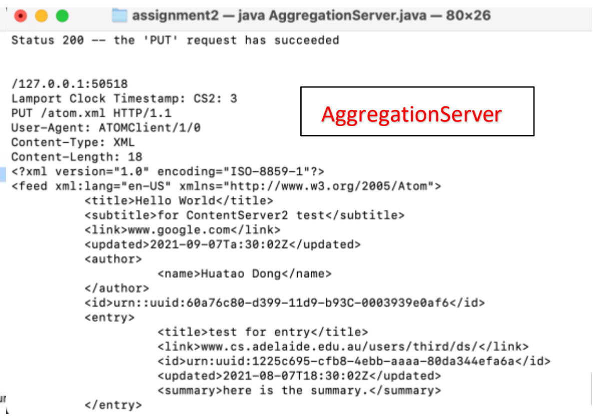
```
richhunter@student-10-201-41-152 assignment2 % java G
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
```

assignment2 — java AggregationServer.java — 80×26
assignment2 — java ContentServer1.java — 79×26
assignment2 — java GETClient2.java — 80×24
assignment2 — java ContentServer2.java — 80×24
assignment2 — java GETClient1.java
AggregationServer
ContentServer1
GETClient2
ContentServer2
GETClient1
Introduction
aggregation.xml
ContentServer2.java
Desktop
GETClient.java

Each PUT request by ContentServer will the display on terminal AggregationServer. The message will also be aggregated and saved in “aggregation.xml”, open it to have a look.
See the IP address and Lamport clock to tell different ContentServer.



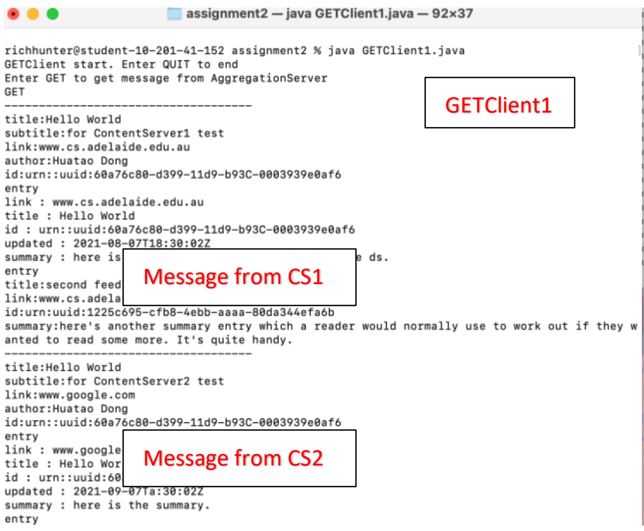
```
Status 200 -- the 'PUT' request has succeeded


/127.0.0.1:49860
Lamport Clock Timestamp: CS1: 2
PUT /atom.xml HTTP/1.1
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 23
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
        <title>Hello World</title>
        <subtitle>for ContentServer1 test</subtitle>
        <link>www.cs.adelaide.edu.au</link>
        <updated>2021-08-07T18:30:02Z</updated>
        <author>
                <name>Huatao Dong</name>
        </author>
        <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
        <entry>
                <title>for entry test</title>
                <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                <updated>2021-08-07T18:30:02Z</updated>
                <summary>here is some plain text. i love ds. i love ds.</sur
mary>
```



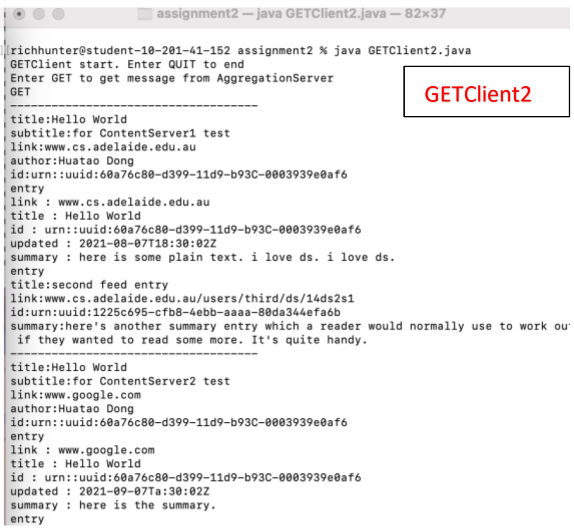
```
Status 200 -- the 'PUT' request has succeeded


/127.0.0.1:50518
Lamport Clock Timestamp: CS2: 3
PUT /atom.xml HTTP/1.1
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 18
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
        <title>Hello World</title>
        <subtitle>for ContentServer2 test</subtitle>
        <link>www.google.com</link>
        <updated>2021-09-07Ta:30:02Z</updated>
        <author>
                <name>Huatao Dong</name>
        </author>
        <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
        <entry>
                <title>test for entry</title>
                <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                <updated>2021-08-07T18:30:02Z</updated>
                <summary>here is the summary.</summary>
        </entry>
```

GETClient send "GET" request to AggregationServer, and it will send xml message back to GETClient, and then GETClient will parse xml to txt.

See GETClient1 and GETClient2 can get both message form ContentServer1 and ContentServer2 followed the order by Lamport clock.

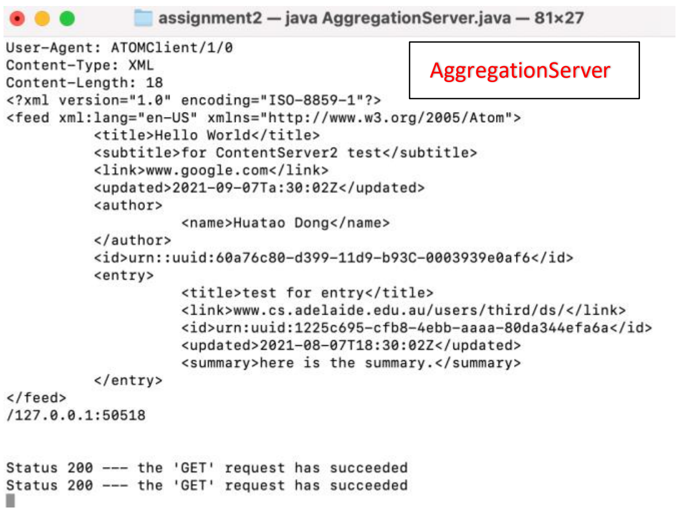


```
richhunter@student-10-201-41-152 assignment2 % java GETClient1.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
GET
------------------------------------
title:Hello World
subtitle:for ContentServer1 test
link:www.cs.adelaide.edu.au
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.cs.adelaide.edu.au
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-08-07T18:30:02Z
summary : here is                                  e ds.
entry
title:second feed
link:www.cs.adela
id:urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b
summary:here's another summary entry which a reader would normally use to work out if they w
anted to read some more. It's quite handy.
------------------------------------
title:Hello World
subtitle:for ContentServer2 test
link:www.google.com
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.google
title : Hello Wor
id : urn::uuid:60
updated : 2021-09-07Ta:30:02Z
summary : here is the summary.
entry
```



```
richhunter@student-10-201-41-152 assignment2 % java GETClient2.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
GET
------------------------------------
title:Hello World
subtitle:for ContentServer1 test
link:www.cs.adelaide.edu.au
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.cs.adelaide.edu.au
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-08-07T18:30:02Z
summary : here is some plain text. i love ds. i love ds.
entry
title:second feed entry
link:www.cs.adelaide.edu.au/users/third/ds/14ds2s1
id:urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b
summary:here's another summary entry which a reader would normally use to work ou
 if they wanted to read some more. It's quite handy.
------------------------------------
title:Hello World
subtitle:for ContentServer2 test
link:www.google.com
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.google.com
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-09-07Ta:30:02Z
summary : here is the summary.
entry
```

The AggregationServer will note Status 200 to show GET succeed



```
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 18
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
        <title>Hello World</title>
        <subtitle>for ContentServer2 test</subtitle>
        <link>www.google.com</link>
        <updated>2021-09-07Ta:30:02Z</updated>
        <author>
                <name>Huatao Dong</name>
        </author>
        <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
        <entry>
                <title>test for entry</title>
                <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                <updated>2021-08-07T18:30:02Z</updated>
                <summary>here is the summary.</summary>
        </entry>
</feed>
/127.0.0.1:50518


Status 200 --- the 'GET' request has succeeded
Status 200 --- the 'GET' request has succeeded
```

Test Case 6:

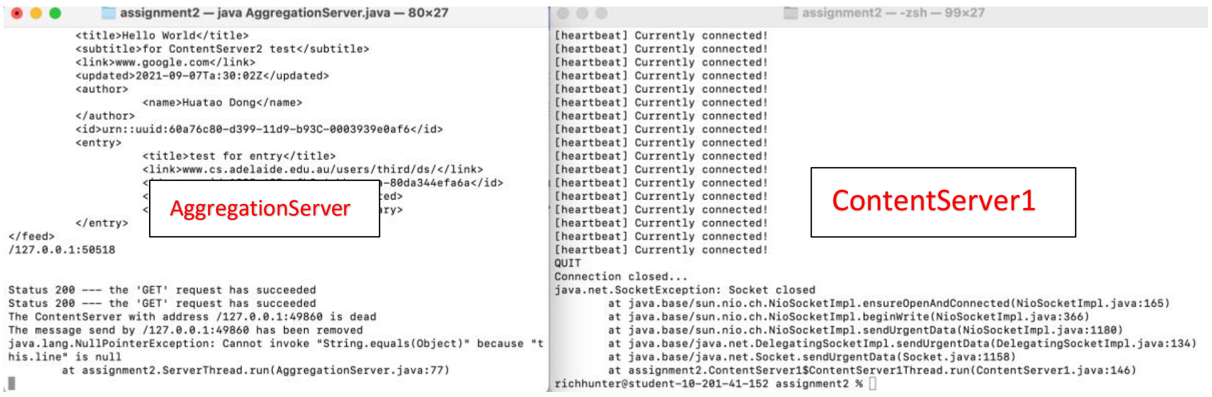
When ContentServer1 disconnected, all messages PUT by ContentServer1 will be removed by AggregationServer, and then if GETClient want to get message, it can only get message from ContentServer2.

Follow the step of test case 3 above, we can see GETClient can get message form both ContentServer1 and ContentServer2 send.

If we close ContentServer1 by entering “QUIT” or control +c

We can see that the AggregationServer display:

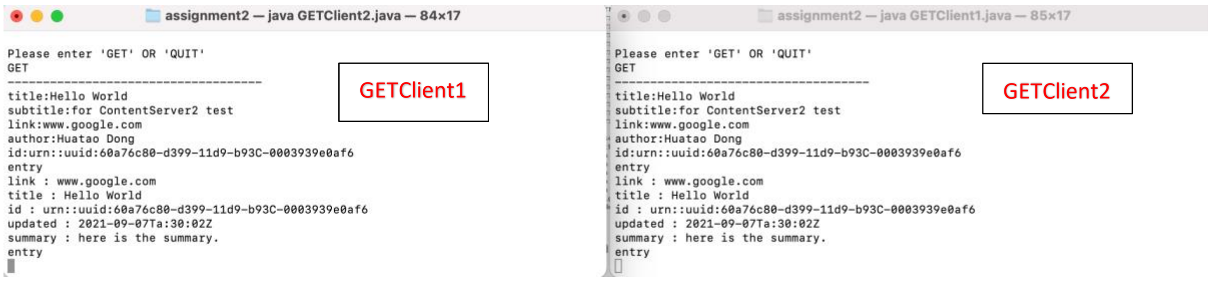
The ContentServer with address /127.0.0.1:49860 is dead
The message sends by /127.0.0.1:49860 has been removed


assignment2 — java AggregationServer.java — 80×27
AggregationServer
assignment2 — -zsh — 99×27
ContentServer1

```
        <title>Hello World</title>
        <subtitle>for ContentServer2 test</subtitle>
        <link>www.google.com</link>
        <updated>2021-09-07Ta:30:02Z</updated>
        <author>
                <name>Huatao Dong</name>
        </author>
        <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
        <entry>
                <title>test for entry</title>
                <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                <                        a-80da344efa6a</id>
                <                        ted>
                <                        ary>
        </entry>
</feed>
/127.0.0.1:50518


Status 200 --- the 'GET' request has succeeded
Status 200 --- the 'GET' request has succeeded
The ContentServer with address /127.0.0.1:49860 is dead
The message send by /127.0.0.1:49860 has been removed
java.lang.NullPointerException: Cannot invoke "String.equals(Object)" because "t
his.line" is null
        at assignment2.ServerThread.run(AggregationServer.java:77)
```

```
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
QUIT
Connection closed...
java.net.SocketException: Socket closed
        at java.base/sun.nio.ch.NioSocketImpl.ensureOpenAndConnected(NioSocketImpl.java:165)
        at java.base/sun.nio.ch.NioSocketImpl.beginWrite(NioSocketImpl.java:366)
        at java.base/sun.nio.ch.NioSocketImpl.sendUrgentData(NioSocketImpl.java:1180)
        at java.base/java.net.DelegatingSocketImpl.sendUrgentData(DelegatingSocketImpl.java:134)
        at java.base/java.net.Socket.sendUrgentData(Socket.java:1158)
        at assignment2.ContentServer1$ContentServer1Thread.run(ContentServer1.java:146)
richhunter@student-10-201-41-152 assignment2 %
```

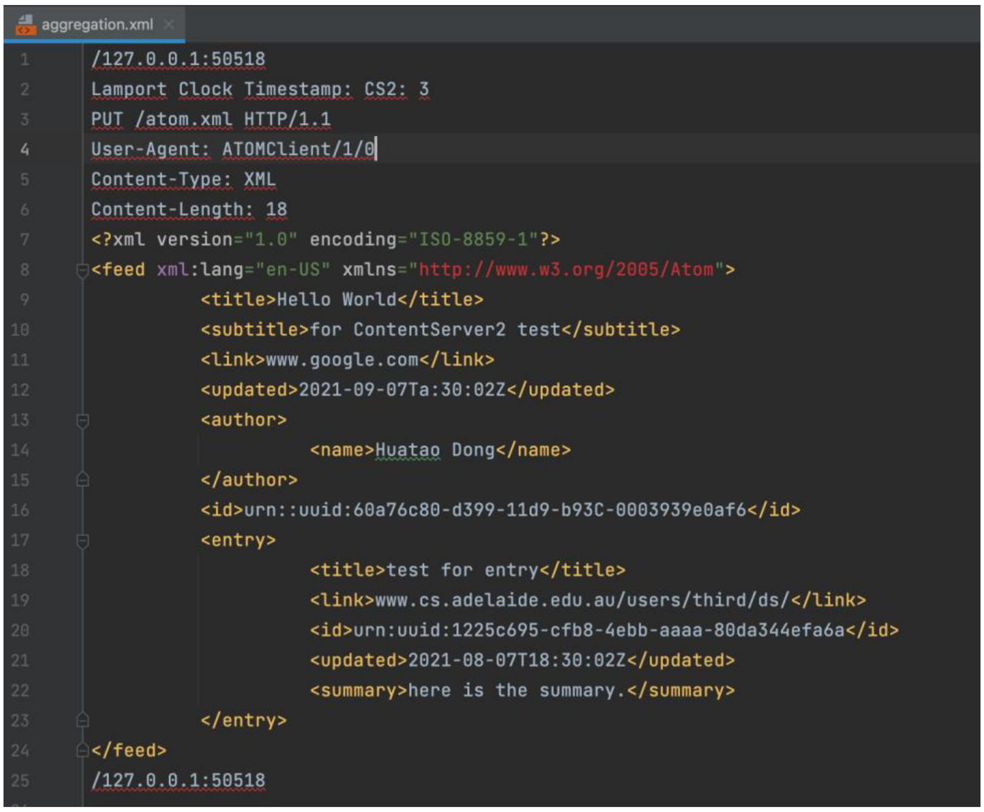
Now we enter “GET” request at GETClient again, we can only get message put by ContentServer2.


assignment2 — java GETClient2.java — 84×17
GETClient1
assignment2 — java GETClient1.java — 85×17
GETClient2

```
Please enter 'GET' OR 'QUIT'
GET
------------------------------------
title:Hello World
subtitle:for ContentServer2 test
link:www.google.com
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.google.com
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-09-07Ta:30:02Z
summary : here is the summary.
entry
```

```
Please enter 'GET' OR 'QUIT'
GET
------------------------------------
title:Hello World
subtitle:for ContentServer2 test
link:www.google.com
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.google.com
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-09-07Ta:30:02Z
summary : here is the summary.
entry
```

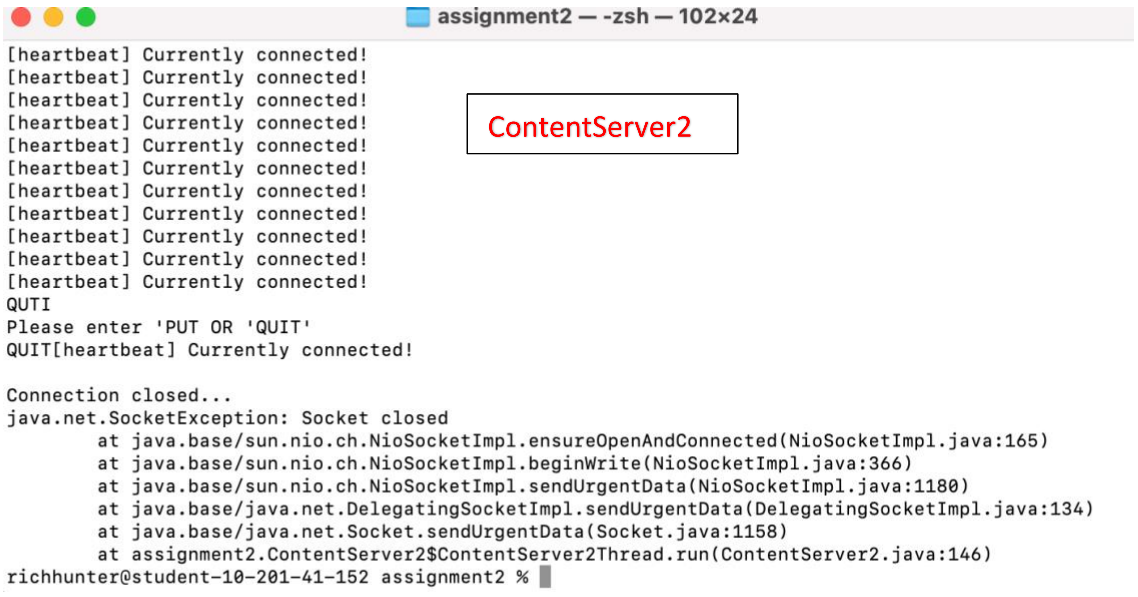
You don’t need to start GETClient again, just enter GET to update. It is Synchronized.
You also can see aggregation.xml, the message put by CS1 has been removed. Only left CS2.


```
aggregation.xml
/127.0.0.1:50518
Lamport Clock Timestamp: CS2: 3
PUT /atom.xml HTTP/1.1
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 18
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
          <title>Hello World</title>
          <subtitle>for ContentServer2 test</subtitle>
          <link>www.google.com</link>
          <updated>2021-09-07Ta:30:02Z</updated>
          <author>
                    <name>Huatao Dong</name>
          </author>
          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
          <entry>
                    <title>test for entry</title>
                    <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                    <updated>2021-08-07T18:30:02Z</updated>
                    <summary>here is the summary.</summary>
          </entry>
</feed>
/127.0.0.1:50518
```

Test Case 7:

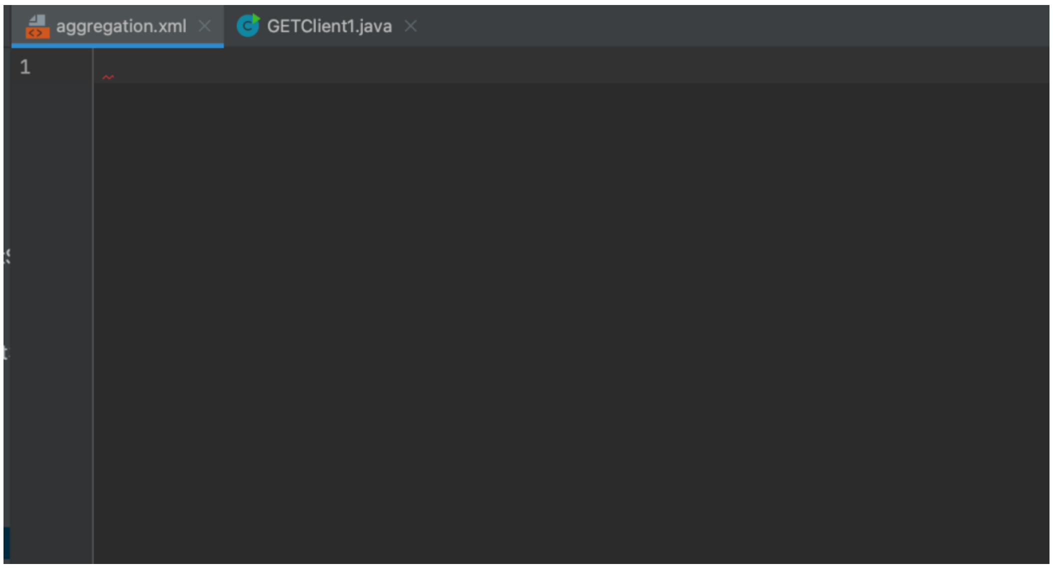
Test for failure tolerant of GETClient: send request to AggregationServer, if no response 3 times, then close GETClient.

Follow the step of test case 4, now we only have ContentServer2 alive and if we close ContentServer2 the message put by it will be remove. That means there is empty on AggregationServer. When GETClient send GET request, there will be no response.

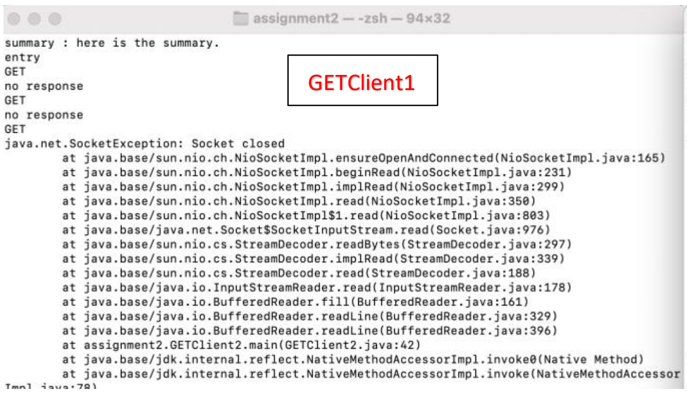


```
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
QUTI
Please enter 'PUT OR 'QUIT'
QUIT[heartbeat] Currently connected!

Connection closed...
java.net.SocketException: Socket closed
        at java.base/sun.nio.ch.NioSocketImpl.ensureOpenAndConnected(NioSocketImpl.java:165)
        at java.base/sun.nio.ch.NioSocketImpl.beginWrite(NioSocketImpl.java:366)
        at java.base/sun.nio.ch.NioSocketImpl.sendUrgentData(NioSocketImpl.java:1180)
        at java.base/java.net.DelegatingSocketImpl.sendUrgentData(DelegatingSocketImpl.java:134)
        at java.base/java.net.Socket.sendUrgentData(Socket.java:1158)
        at assignment2.ContentServer2$ContentServer2Thread.run(ContentServer2.java:146)
richhunter@student-10-201-41-152 assignment2 %
```

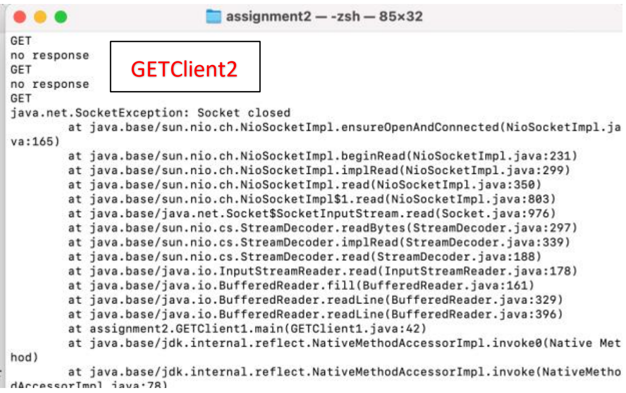
ContentServer2 closed and message has been removed. aggregation.xml is empty now.



Then if we send "GET" request on GETClient, there will be no xml message response. After send 3 times the GETClient will be closed. Test passed.



```
summary : here is the summary.
entry
GET
no response
GET
no response
GET
java.net.SocketException: Socket closed
        at java.base/sun.nio.ch.NioSocketImpl.ensureOpenAndConnected(NioSocketImpl.java:165)
        at java.base/sun.nio.ch.NioSocketImpl.beginRead(NioSocketImpl.java:231)
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:299)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
        at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:297)
        at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:339)
        at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
        at java.base/java.io.InputStreamReader.read(InputStreamReader.java:178)
        at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
        at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
        at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
        at assignment2.GETClient2.main(GETClient2.java:42)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessor
Impl.java:78)
```

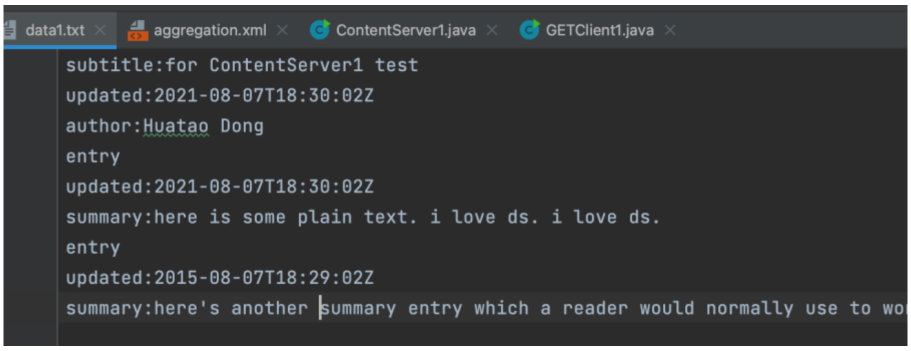


```
GET
no response
GET
no response
GET
java.net.SocketException: Socket closed
        at java.base/sun.nio.ch.NioSocketImpl.ensureOpenAndConnected(NioSocketImpl.ja
va:165)
        at java.base/sun.nio.ch.NioSocketImpl.beginRead(NioSocketImpl.java:231)
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:299)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:350)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:803)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:976)
        at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:297)
        at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:339)
        at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
        at java.base/java.io.InputStreamReader.read(InputStreamReader.java:178)
        at java.base/java.io.BufferedReader.fill(BufferedReader.java:161)
        at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
        at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
        at assignment2.GETClient1.main(GETClient1.java:42)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Met
hod)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMetho
dAccessorImpl.java:78)
```

Test case 8:

Test for input the source file with no title, link or id. The parser will reject to parse.

Open the input source file data1.txt, and then delete all title, link or id.

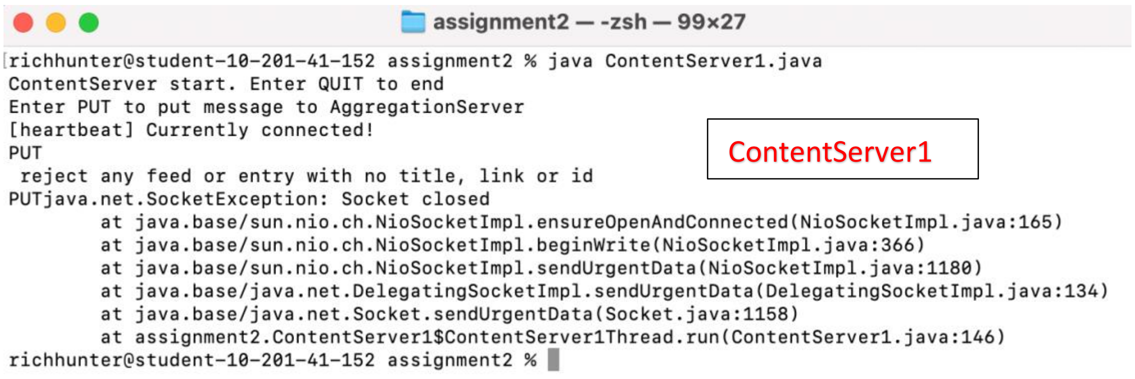


```
subtitle:for ContentServer1 test
updated:2021-08-07T18:30:02Z
author:Huatao Dong
entry
updated:2021-08-07T18:30:02Z
summary:here is some plain text. i love ds. i love ds.
entry
updated:2015-08-07T18:29:02Z
summary:here's another summary entry which a reader would normally use to wo
```

Open ContentServer1.java to send PUT request to AggregationServer. See the result.

The ContentServer1 will refuse to parse the source file and close. You must change the data1.txt to add title, link or id.

Test passed.



assignment2 — -zsh — 99×27

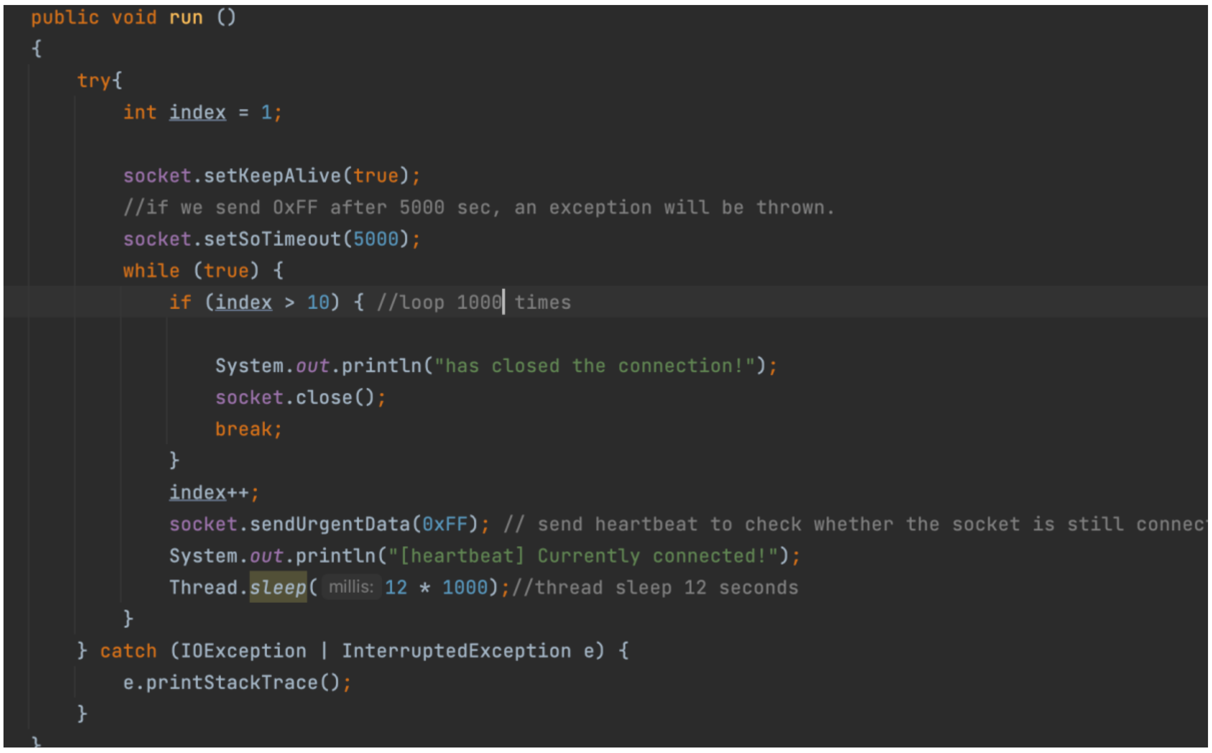
```
richhunter@student-10-201-41-152 assignment2 % java ContentServer1.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
PUT
 reject any feed or entry with no title, link or id
PUTjava.net.SocketException: Socket closed
        at java.base/sun.nio.ch.NioSocketImpl.ensureOpenAndConnected(NioSocketImpl.java:165)
        at java.base/sun.nio.ch.NioSocketImpl.beginWrite(NioSocketImpl.java:366)
        at java.base/sun.nio.ch.NioSocketImpl.sendUrgentData(NioSocketImpl.java:1180)
        at java.base/java.net.DelegatingSocketImpl.sendUrgentData(DelegatingSocketImpl.java:134)
        at java.base/java.net.Socket.sendUrgentData(Socket.java:1158)
        at assignment2.ContentServer1$ContentServer1Thread.run(ContentServer1.java:146)
richhunter@student-10-201-41-152 assignment2 %
```

ContentServer1

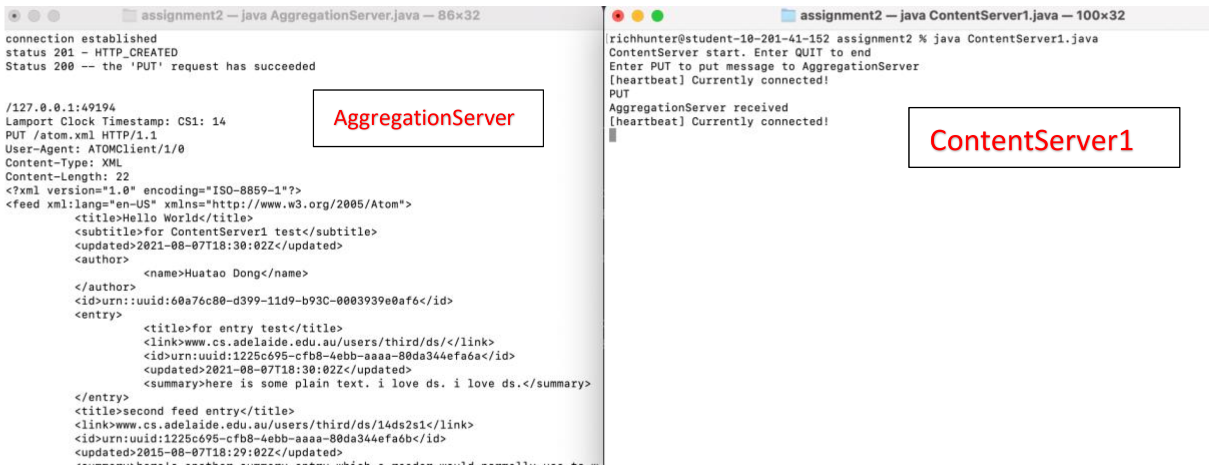
Test case 9:

test for heartbeat. If ContentServer lose heartbeat, it will be closed, and message send will also be removed.

Here is ContentServer1 heartbeat. The default loop is 1000 times. I changed it to loop 10 time, and after send 10 times heartbeat I will lose it.

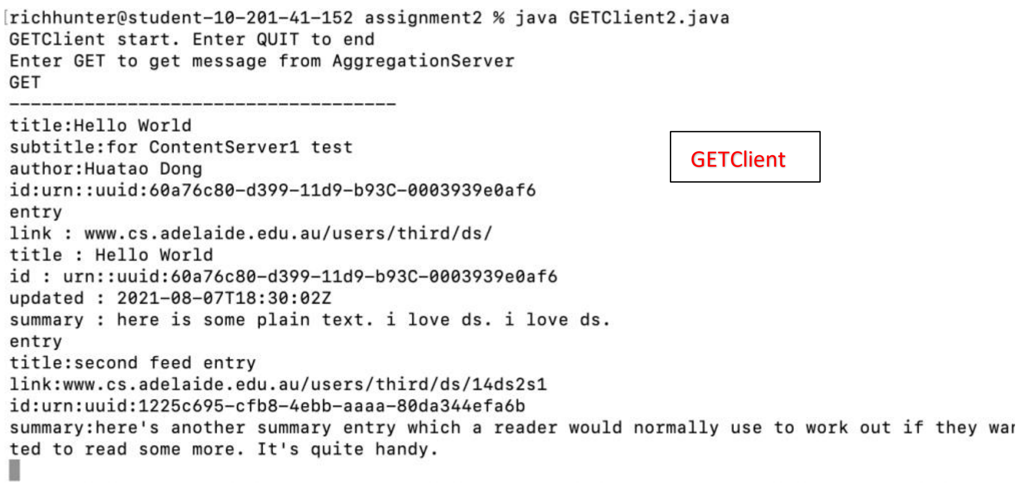


```
public void run ()
{
    try{
        int index = 1;

        socket.setKeepAlive(true);
        //if we send 0xFF after 5000 sec, an exception will be thrown.
        socket.setSoTimeout(5000);
        while (true) {
            if (index > 10) { //loop 1000 times

                System.out.println("has closed the connection!");
                socket.close();
                break;
            }
            index++;
            socket.sendUrgentData(0xFF); // send heartbeat to check whether the socket is still connec
            System.out.println("[heartbeat] Currently connected!");
            Thread.sleep( millis: 12 * 1000);//thread sleep 12 seconds
        }
    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
    }
}
```


assignment2 — java AggregationServer.java — 86×32
AggregationServer
assignment2 — java ContentServer1.java — 100×32
ContentServer1

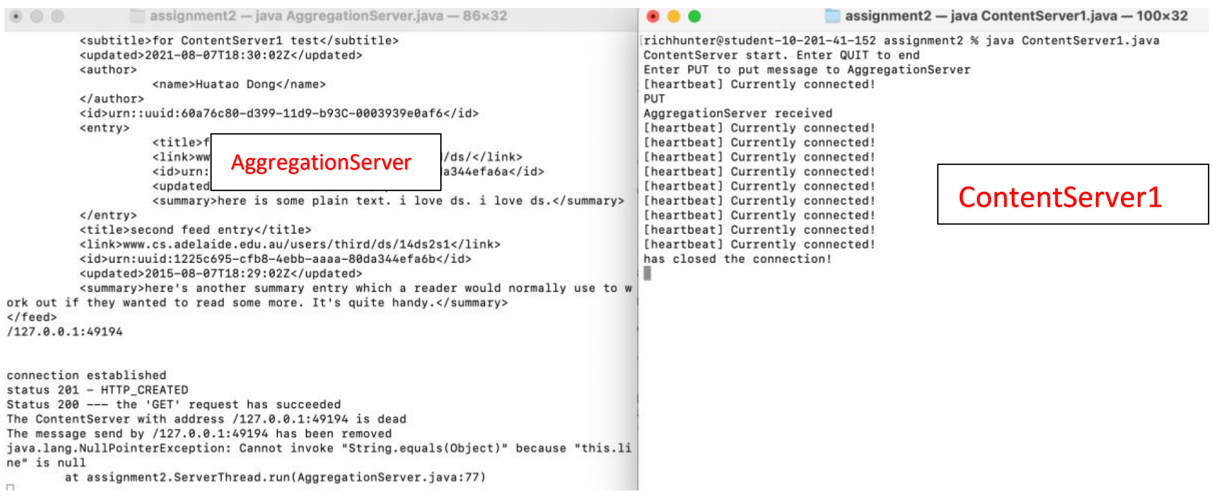
```
connection established
status 201 - HTTP_CREATED
Status 200 -- the 'PUT' request has succeeded


/127.0.0.1:49194
Lamport Clock Timestamp: CS1: 14
PUT /atom.xml HTTP/1.1
User-Agent: ATOMClient/1/0
Content-Type: XML
Content-Length: 22
<?xml version="1.0" encoding="ISO-8859-1"?>
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom">
          <title>Hello World</title>
          <subtitle>for ContentServer1 test</subtitle>
          <updated>2021-08-07T18:30:02Z</updated>
          <author>
                    <name>Huatao Dong</name>
          </author>
          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
          <entry>
                    <title>for entry test</title>
                    <link>www.cs.adelaide.edu.au/users/third/ds/</link>
                    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
                    <updated>2021-08-07T18:30:02Z</updated>
                    <summary>here is some plain text. i love ds. i love ds.</summary>
          </entry>
          <title>second feed entry</title>
          <link>www.cs.adelaide.edu.au/users/third/ds/14ds2s1</link>
          <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b</id>
          <updated>2015-08-07T18:29:02Z</updated>
```

```
richhunter@student-10-201-41-152 assignment2 % java ContentServer1.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
PUT
AggregationServer received
[heartbeat] Currently connected!
```



```
richhunter@student-10-201-41-152 assignment2 % java GETClient2.java
GETClient start. Enter QUIT to end
Enter GET to get message from AggregationServer
GET
------------------------------------
title:Hello World
subtitle:for ContentServer1 test
author:Huatao Dong
id:urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
entry
link : www.cs.adelaide.edu.au/users/third/ds/
title : Hello World
id : urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6
updated : 2021-08-07T18:30:02Z
summary : here is some plain text. i love ds. i love ds.
entry
title:second feed entry
link:www.cs.adelaide.edu.au/users/third/ds/14ds2s1
id:urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b
summary:here's another summary entry which a reader would normally use to work out if they wan
ted to read some more. It's quite handy.
```

After send 10 times heartbeat, it closed the connection. The AggregationServer also noted and removed message send by CS1.


assignment2 — java AggregationServer.java — 86×32
AggregationServer
assignment2 — java ContentServer1.java — 100×32
ContentServer1

```
          <subtitle>for ContentServer1 test</subtitle>
          <updated>2021-08-07T18:30:02Z</updated>
          <author>
                    <name>Huatao Dong</name>
          </author>
          <id>urn::uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
          <entry>
                    <title>f
                    <link>ww                                  /ds/</link>
                    <id>urn:                                  a344efa6a</id>
                    <updated
                    <summary>here is some plain text. i love ds. i love ds.</summary>
          </entry>
          <title>second feed entry</title>
          <link>www.cs.adelaide.edu.au/users/third/ds/14ds2s1</link>
          <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6b</id>
          <updated>2015-08-07T18:29:02Z</updated>
          <summary>here's another summary entry which a reader would normally use to w
ork out if they wanted to read some more. It's quite handy.</summary>
</feed>
/127.0.0.1:49194


connection established
status 201 - HTTP_CREATED
Status 200 --- the 'GET' request has succeeded
The ContentServer with address /127.0.0.1:49194 is dead
The message send by /127.0.0.1:49194 has been removed
java.lang.NullPointerException: Cannot invoke "String.equals(Object)" because "this.li
ne" is null
        at assignment2.ServerThread.run(AggregationServer.java:77)
```

```
richhunter@student-10-201-41-152 assignment2 % java ContentServer1.java
ContentServer start. Enter QUIT to end
Enter PUT to put message to AggregationServer
[heartbeat] Currently connected!
PUT
AggregationServer received
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
[heartbeat] Currently connected!
has closed the connection!
```

The heartbeat works very well. Test passed.

THANK YOU!!!!!!