URLConnection Class

Hyang-Won Lee
Department of CSE, Konkuk University
leehw@konkuk.ac.kr
https://sites.google.com/view/leehwko/

URLConnection Class

- Provide more control over interaction with server (than URL class)
  - can inspect header sent by server and respond accordingly
  - can set header fields used in client request
  - can send data back to server using POST/PUT

- Base java.net.URLConnection class is abstract
  - concrete subclasses are hidden in java.net package hierarchy

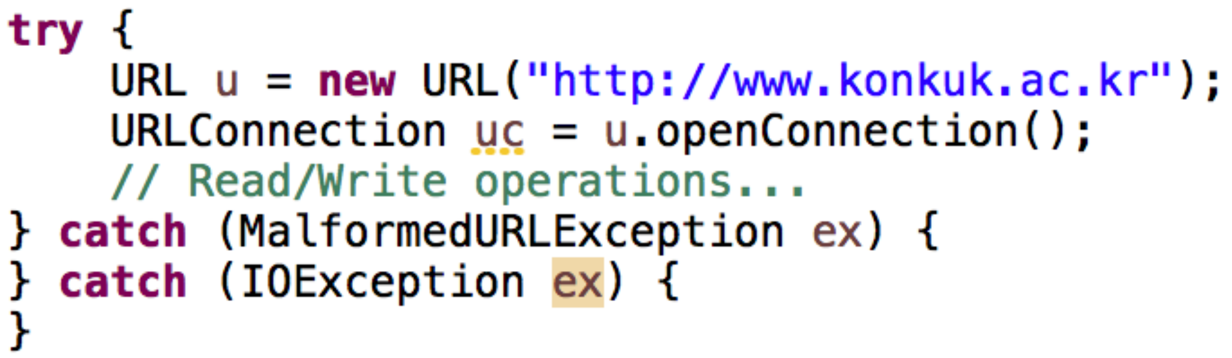
Basic Sequence of Steps

- Construct URL object
- Invoke URL object's openConnection() method to retrieve URLConnection object
- Configure URLConnection
- Read header fields
- Get input stream and read data
- Get output stream and write data
- Close connection

can skip some of the steps depending on your needs

Constructor

- Single constructor
  - protected URLConnection(URL url)
  - cannot be called directly
  - can be called either by subclassing or invoking openConnection() doesn’t establish actual network connection



```
try {
    URL u = new URL("http://www.konkuk.ac.kr");
    URLConnection uc = u.openConnection();
    // Read/Write operations...
} catch (MalformedURLException ex) {
} catch (IOException ex) {
}
```

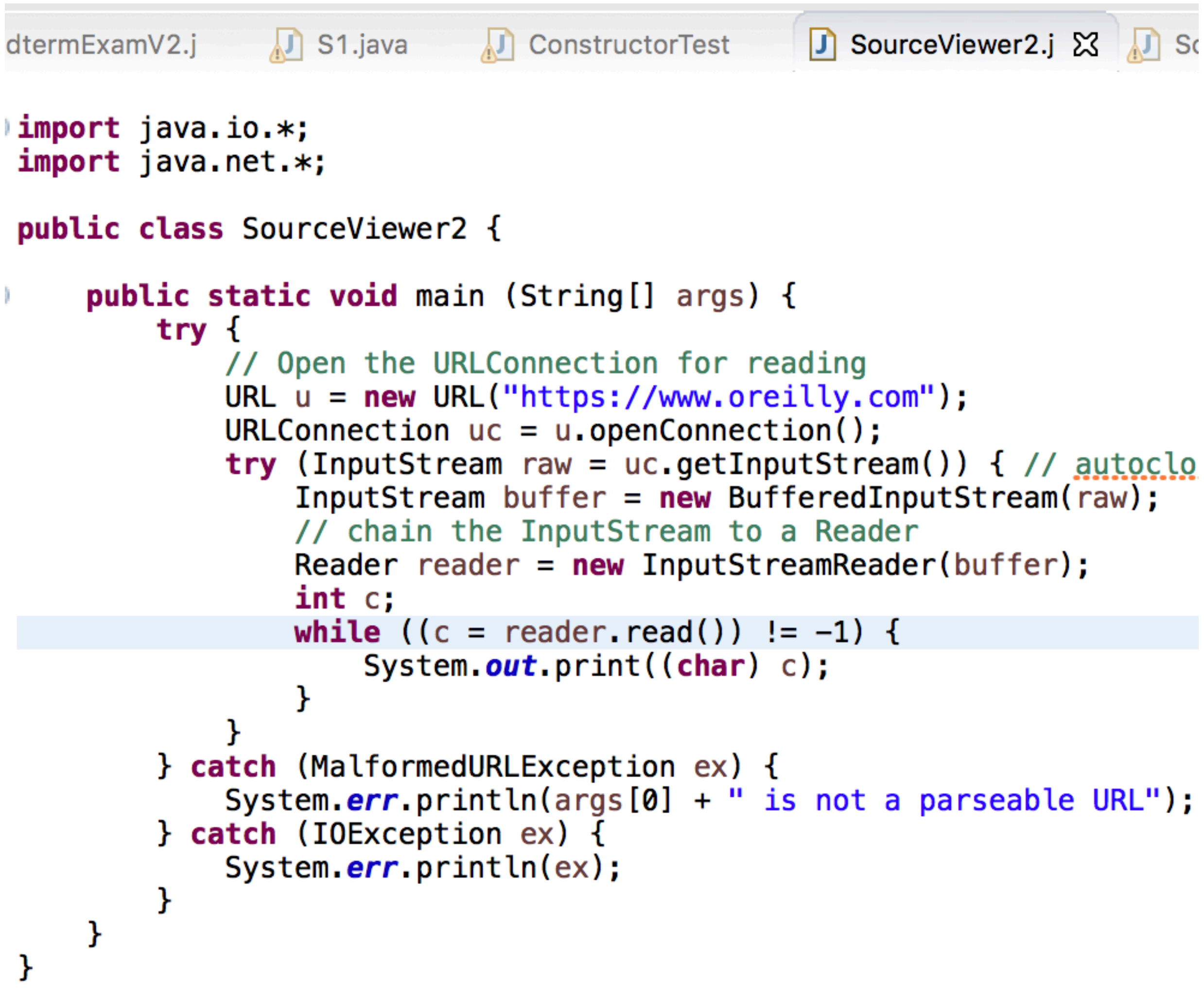
URLConnection Class is Abstract

- All but one of its methods are implemented
- Subclasses must implement connect() method which makes an actual connection to server
  - public abstract void connect() throws IOException
- When URLConnection is first constructed, it is unconnected ⇒ connect() method establishes a connection

Reading Data from a Server

- Procedure
  - construct a URL object
  - Invoke the URL object's openConnection() method to retrieve URLConnection object for that URL
  - Invoke the URLConnection's getInputStream() method
  - Read from the input stream using the usual stream API

Example



```java
import java.io.*;
import java.net.*;

public class SourceViewer2 {

    public static void main (String[] args) {
        try {
            // Open the URLConnection for reading
            URL u = new URL("https://www.oreilly.com");
            URLConnection uc = u.openConnection();
            try (InputStream raw = uc.getInputStream()) { // autoclo
                InputStream buffer = new BufferedInputStream(raw);
                // chain the InputStream to a Reader
                Reader reader = new InputStreamReader(buffer);
                int c;
                while ((c = reader.read()) != -1) {
                    System.out.print((char) c);
                }
            }
        } catch (MalformedURLException ex) {
            System.err.println(args[0] + " is not a parseable URL");
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

- Exercise
  - chain BufferedReader to InputStreamReader and change while loop

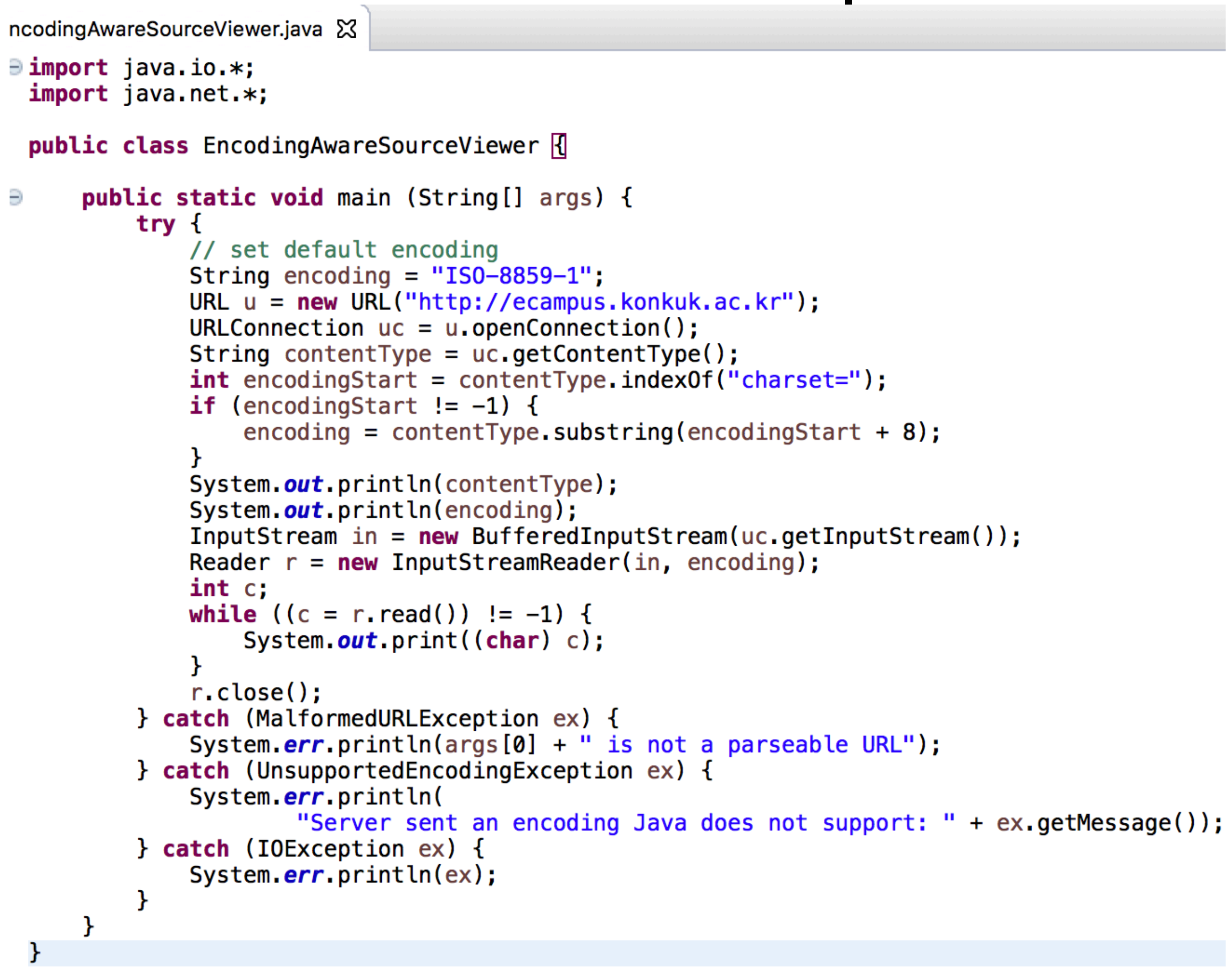
Reading the Header

- Information in HTTP header
  - content type, length, encoding, data and time info,…
- URLConnection provides utility methods to get specific fields in header
  - Content-type
  - Content-length
  - Content-encoding
  - Data
  - Last-modified
  - Expires

Content-type

- Common content types
  - text/html, text/plain, image/gif, application/xml, image/jpeg
- Content-type containing character set part
  - Content-type: text/html; charset=UTF-8
- Method
  - public String getContentType()

Example



ncodingAwareSourceViewer.java

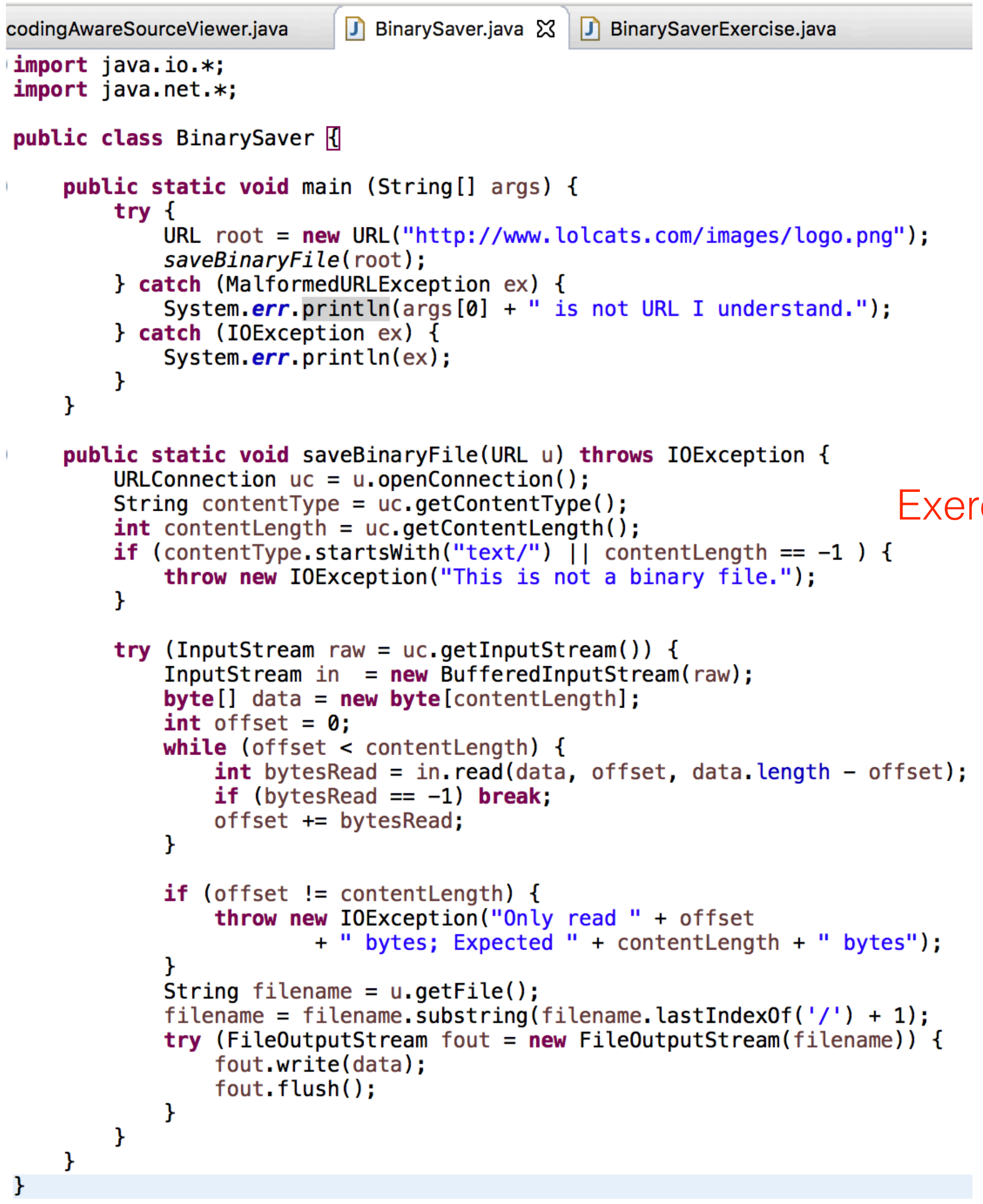
```java
import java.io.*;
import java.net.*;

public class EncodingAwareSourceViewer {

    public static void main (String[] args) {
        try {
            // set default encoding
            String encoding = "ISO-8859-1";
            URL u = new URL("http://ecampus.konkuk.ac.kr");
            URLConnection uc = u.openConnection();
            String contentType = uc.getContentType();
            int encodingStart = contentType.indexOf("charset=");
            if (encodingStart != -1) {
                encoding = contentType.substring(encodingStart + 8);
            }
            System.out.println(contentType);
            System.out.println(encoding);
            InputStream in = new BufferedInputStream(uc.getInputStream());
            Reader r = new InputStreamReader(in, encoding);
            int c;
            while ((c = r.read()) != -1) {
                System.out.print((char) c);
            }
            r.close();
        } catch (MalformedURLException ex) {
            System.err.println(args[0] + " is not a parseable URL");
        } catch (UnsupportedEncodingException ex) {
            System.err.println(
                "Server sent an encoding Java does not support: " + ex.getMessage());
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

- Exercise
  - Find out when the actual connection to server is established

Content-length

- Method
  - public int getContentLength
  - public long getContentLengthLong

Example



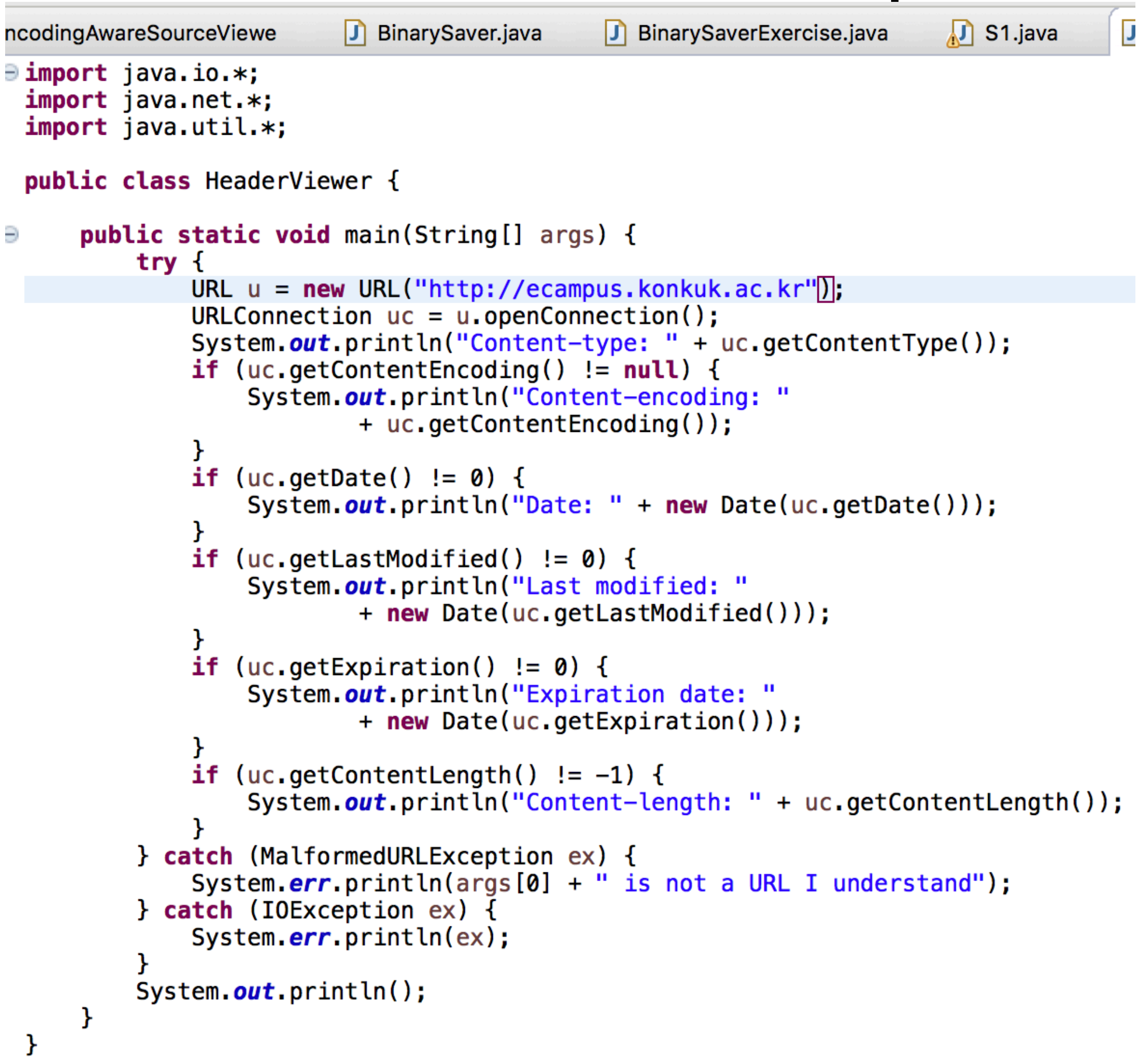
```java
import java.io.*;
import java.net.*;

public class BinarySaver {

    public static void main (String[] args) {
        try {
            URL root = new URL("http://www.lolcats.com/images/logo.png");
            saveBinaryFile(root);
        } catch (MalformedURLException ex) {
            System.err.println(args[0] + " is not URL I understand.");
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }

    public static void saveBinaryFile(URL u) throws IOException {
        URLConnection uc = u.openConnection();
        String contentType = uc.getContentType();
        int contentLength = uc.getContentLength();
        if (contentType.startsWith("text/") || contentLength == -1 ) {
            throw new IOException("This is not a binary file.");
        }

        try (InputStream raw = uc.getInputStream()) {
            InputStream in  = new BufferedInputStream(raw);
            byte[] data = new byte[contentLength];
            int offset = 0;
            while (offset < contentLength) {
                int bytesRead = in.read(data, offset, data.length - offset);
                if (bytesRead == -1) break;
                offset += bytesRead;
            }

            if (offset != contentLength) {
                throw new IOException("Only read " + offset
                        + " bytes; Expected " + contentLength + " bytes");
            }
            String filename = u.getFile();
            filename = filename.substring(filename.lastIndexOf('/') + 1);
            try (FileOutputStream fout = new FileOutputStream(filename)) {
                fout.write(data);
                fout.flush();
            }
        }
    }
}
```

Exercise: try to read single bytes directly from InputStream

Other Methods

- public String getContentEncoding()
  - content encoding (different from Content-type)
  - content sent unencoded ⇒ null returned
- public long getDate()
  - when the document was sent (as seen from server)
  - returned value: milliseconds since midnight GMT, 1/1, 1970
  - (conversion) Date d = new Date(uc.getDate())
- public long getExpiration()
  - indicate when document should be deleted from cache and reloaded from server
  - returned value: milliseconds
- public long getLastModified()
  - date when document was last modified
  - returned value: milliseconds

Example

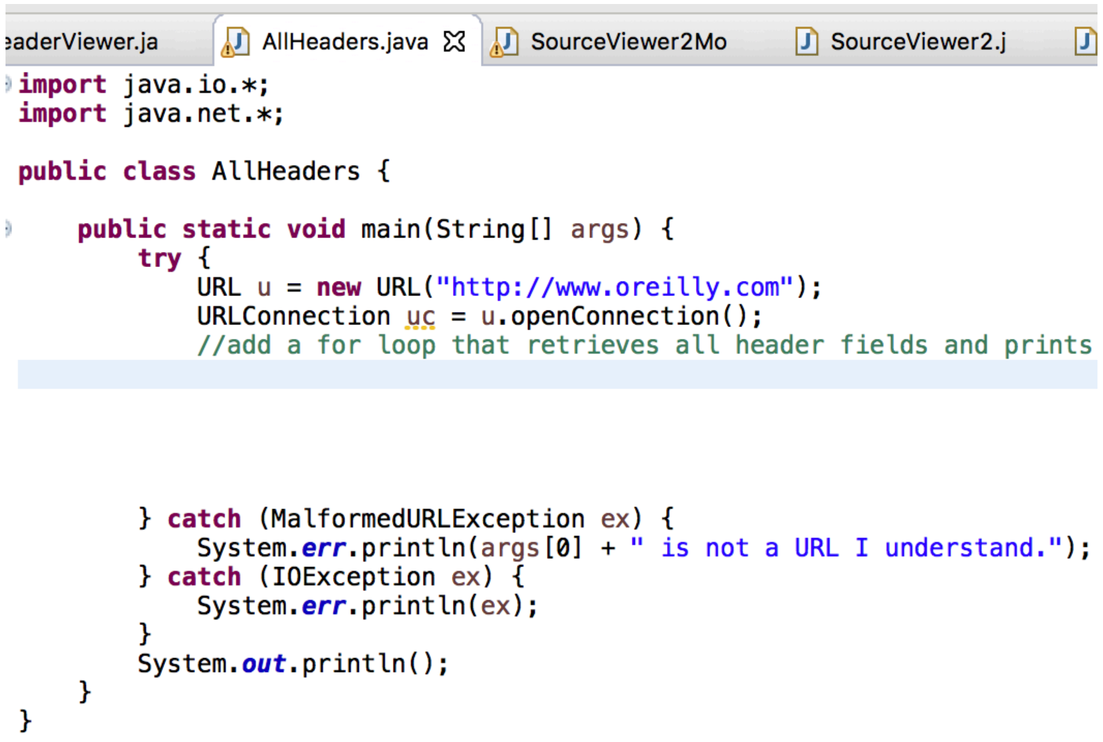


```
import java.io.*;
import java.net.*;
import java.util.*;

public class HeaderViewer {

    public static void main(String[] args) {
        try {
            URL u = new URL("http://ecampus.konkuk.ac.kr");
            URLConnection uc = u.openConnection();
            System.out.println("Content-type: " + uc.getContentType());
            if (uc.getContentEncoding() != null) {
                System.out.println("Content-encoding: "
                        + uc.getContentEncoding());
            }
            if (uc.getDate() != 0) {
                System.out.println("Date: " + new Date(uc.getDate()));
            }
            if (uc.getLastModified() != 0) {
                System.out.println("Last modified: "
                        + new Date(uc.getLastModified()));
            }
            if (uc.getExpiration() != 0) {
                System.out.println("Expiration date: "
                        + new Date(uc.getExpiration()));
            }
            if (uc.getContentLength() != -1) {
                System.out.println("Content-length: " + uc.getContentLength());
            }
        } catch (MalformedURLException ex) {
            System.err.println(args[0] + " is not a URL I understand");
        } catch (IOException ex) {
            System.err.println(ex);
        }
        System.out.println();
    }
}
```

Retrieving Arbitrary Header Fields

- public String getHeaderField(String name)
  - e.g., String ct = uc.getHeaderField(“content-type”);
- public String getHeaderFieldKey(int n)
  - field name of nth header field
- public String getHeaderField(int n)
  - value of nth header field

Example



```java
import java.io.*;
import java.net.*;

public class AllHeaders {

    public static void main(String[] args) {
        try {
            URL u = new URL("http://www.oreilly.com");
            URLConnection uc = u.openConnection();
            //add a for loop that retrieves all header fields and prints




        } catch (MalformedURLException ex) {
            System.err.println(args[0] + " is not a URL I understand.");
        } catch (IOException ex) {
            System.err.println(ex);
        }
        System.out.println();
    }
}
```

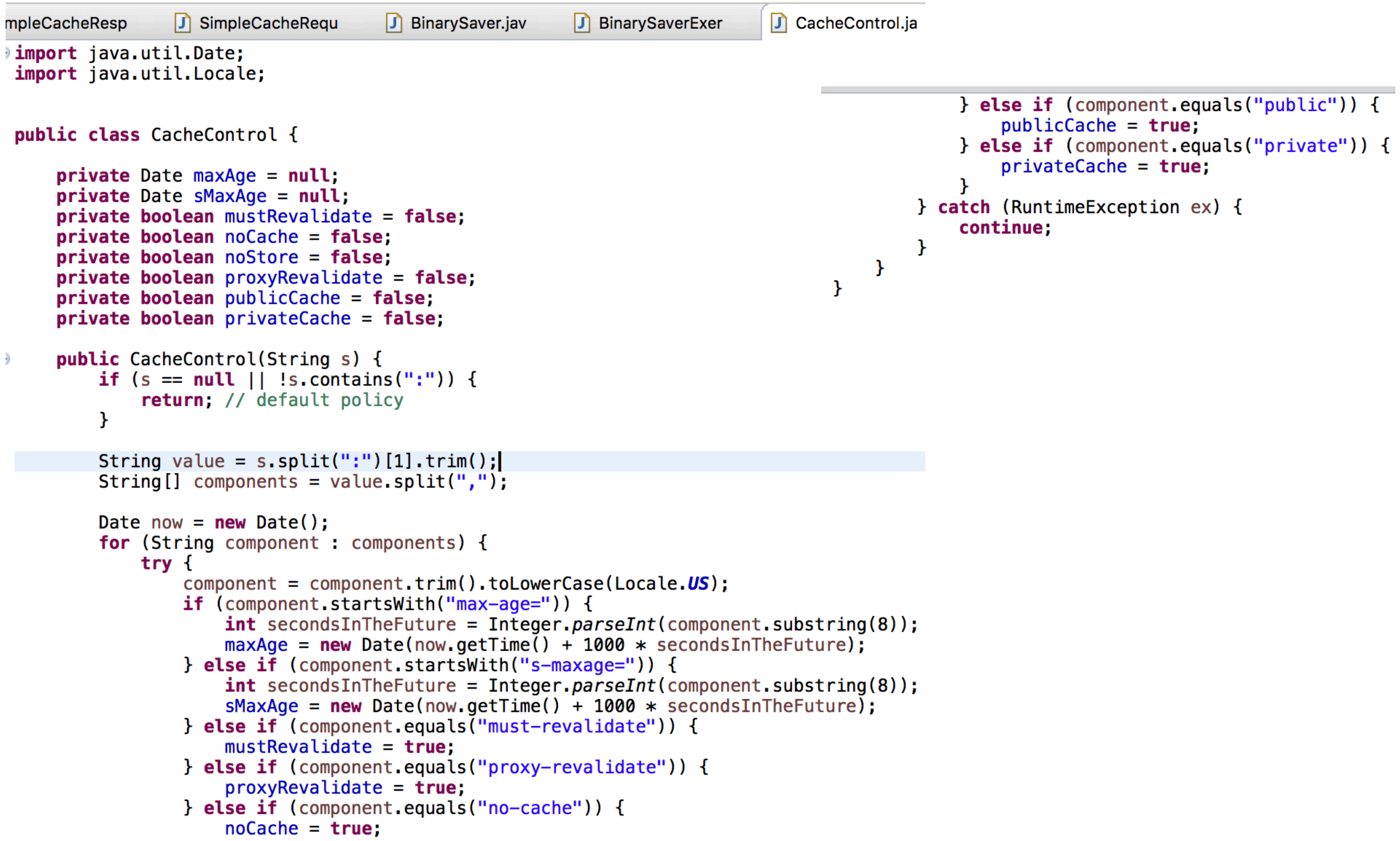
- Exercise
  - add a for loop that retrieves all header fields and prints them on console

Cache

Caches

- Web browsers cache pages and images accessed with GET over HTTP
  - used when reloading the page
- HTTP headers controlling cache
  - Expires
    - indicate that it's ok to cache this representation until specified time
  - Cache-control
    - offer fine-grained cache policies
    - max-age=[seconds]: number of seconds from now before expiration
    - s-maxage=[seconds]: number of seconds from now before expiration (shared cache)
    - public: cache an authenticated response
    - private: only single user caches should store response
    - no-cache: client should reverify (Last-modified) on each access
    - no-store: do not cache

Example



mpleCacheResp | SimpleCacheRequ | BinarySaver.jav | BinarySaverExer | CacheControl.ja

```java
import java.util.Date;
import java.util.Locale;


public class CacheControl {

    private Date maxAge = null;
    private Date sMaxAge = null;
    private boolean mustRevalidate = false;
    private boolean noCache = false;
    private boolean noStore = false;
    private boolean proxyRevalidate = false;
    private boolean publicCache = false;
    private boolean privateCache = false;

    public CacheControl(String s) {
        if (s == null || !s.contains(":")) {
            return; // default policy
        }

        String value = s.split(":")[1].trim();
        String[] components = value.split(",");

        Date now = new Date();
        for (String component : components) {
            try {
                component = component.trim().toLowerCase(Locale.US);
                if (component.startsWith("max-age=")) {
                    int secondsInTheFuture = Integer.parseInt(component.substring(8));
                    maxAge = new Date(now.getTime() + 1000 * secondsInTheFuture);
                } else if (component.startsWith("s-maxage=")) {
                    int secondsInTheFuture = Integer.parseInt(component.substring(8));
                    sMaxAge = new Date(now.getTime() + 1000 * secondsInTheFuture);
                } else if (component.equals("must-revalidate")) {
                    mustRevalidate = true;
                } else if (component.equals("proxy-revalidate")) {
                    proxyRevalidate = true;
                } else if (component.equals("no-cache")) {
                    noCache = true;
                } else if (component.equals("public")) {
                    publicCache = true;
                } else if (component.equals("private")) {
                    privateCache = true;
                }
            } catch (RuntimeException ex) {
                continue;
            }
        }
    }
}
```

Web Cache for Java

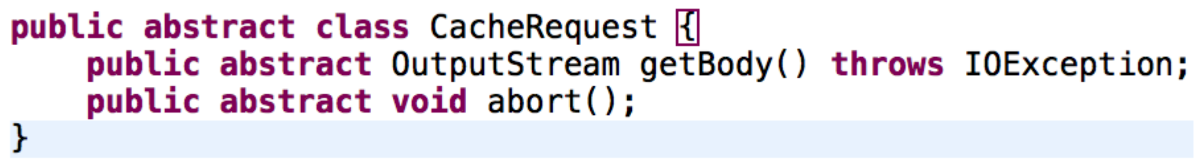
- Java does not cache anything
- To install a system-wide cache for URL class, we need
  - concrete subclass of ResponseCache
  - concrete subclass of CacheRequest
  - concrete subclass of CacheResponse
- ResponseCache.setDefault()
  - subclass of ResponseCache will work with subclasses of CacheRequest and CacheResponse
- JVM can only support a single shared cache

Operations with Cache

- Whenever system tries to load new URL, it first looks into cache
- If cache returns desired content, URLConnection doesn't need to connect to remote server
- If requested data is not in cache, it will be downloaded
  - and response will be cached
- Two abstract methods (in ResponseCache) for storing and retrieving data from cache
  - public abstract CacheResponse get(URI uri, String requestMethod, Map<String, List<String>> requestHeaders) throws IOException
  - public abstract CacheRequest put(URI uri, URLConnection connection) throws IOException

CacheRequest Class

- Abstract class

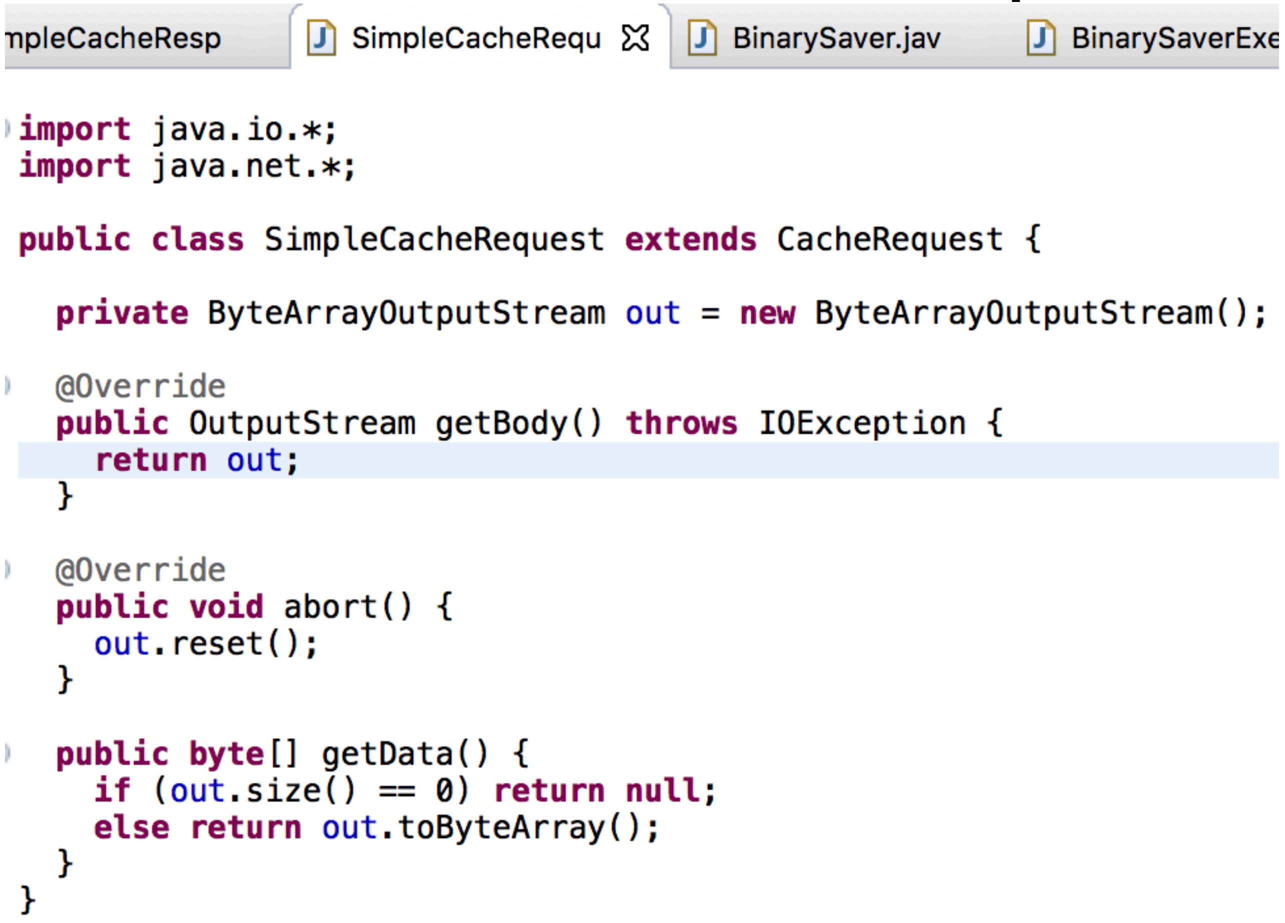


```
public abstract class CacheRequest {
    public abstract OutputStream getBody() throws IOException;
    public abstract void abort();
}
```

- getBody()
  - should return an OutputStream that points into the cache's data for the URI passed to the put() method at the same time
- abort()
  - called when a problem arises while copying (e.g., server unexpectedly closes connection)

CoIn LAB

Example

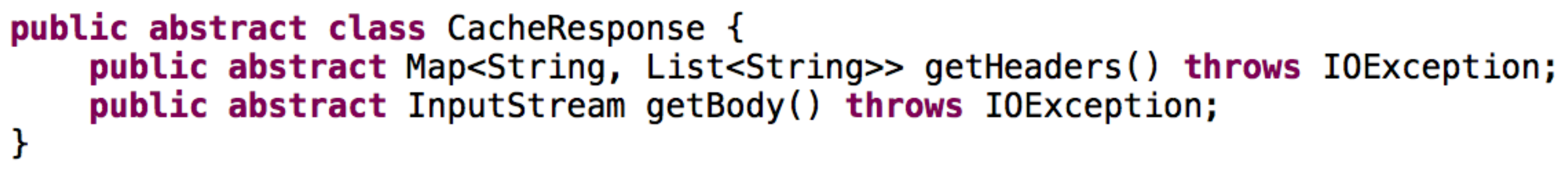


```java
import java.io.*;
import java.net.*;

public class SimpleCacheRequest extends CacheRequest {

  private ByteArrayOutputStream out = new ByteArrayOutputStream();

  @Override
  public OutputStream getBody() throws IOException {
    return out;
  }

  @Override
  public void abort() {
    out.reset();
  }

  public byte[] getData() {
    if (out.size() == 0) return null;
    else return out.toByteArray();
  }
}
```

◉

CacheResponse Class

- Abstract class



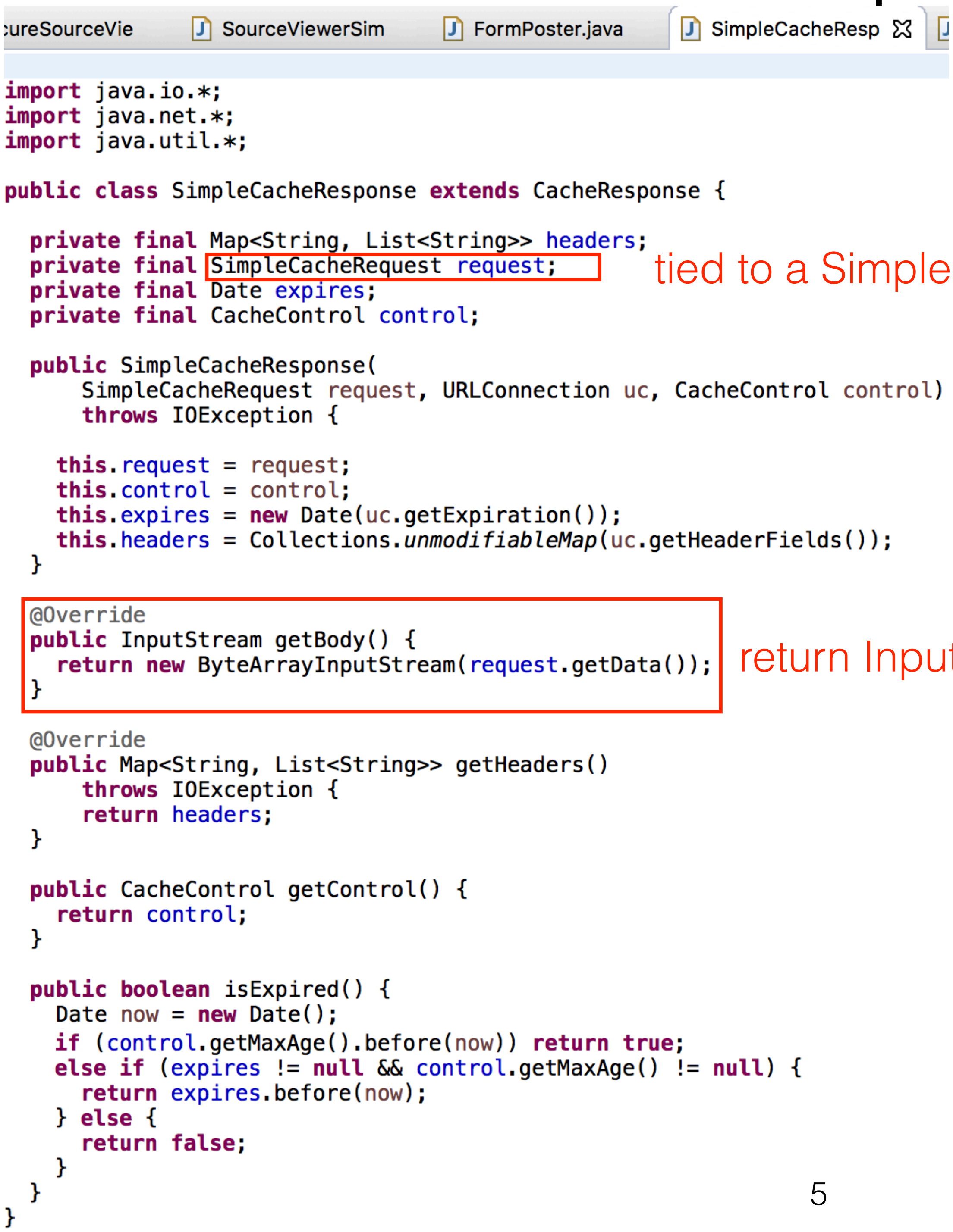
```
public abstract class CacheResponse {
    public abstract Map<String, List<String>> getHeaders() throws IOException;
    public abstract InputStream getBody() throws IOException;
}
```

- getBody()
  - returns an InputStream from which response body can be accessed

- getHeaders()
  - returns a Map from response header field names to lists of field values (~~Content-type: text/html, image/gif, image/png~~)

multiple field values for a field name

Example



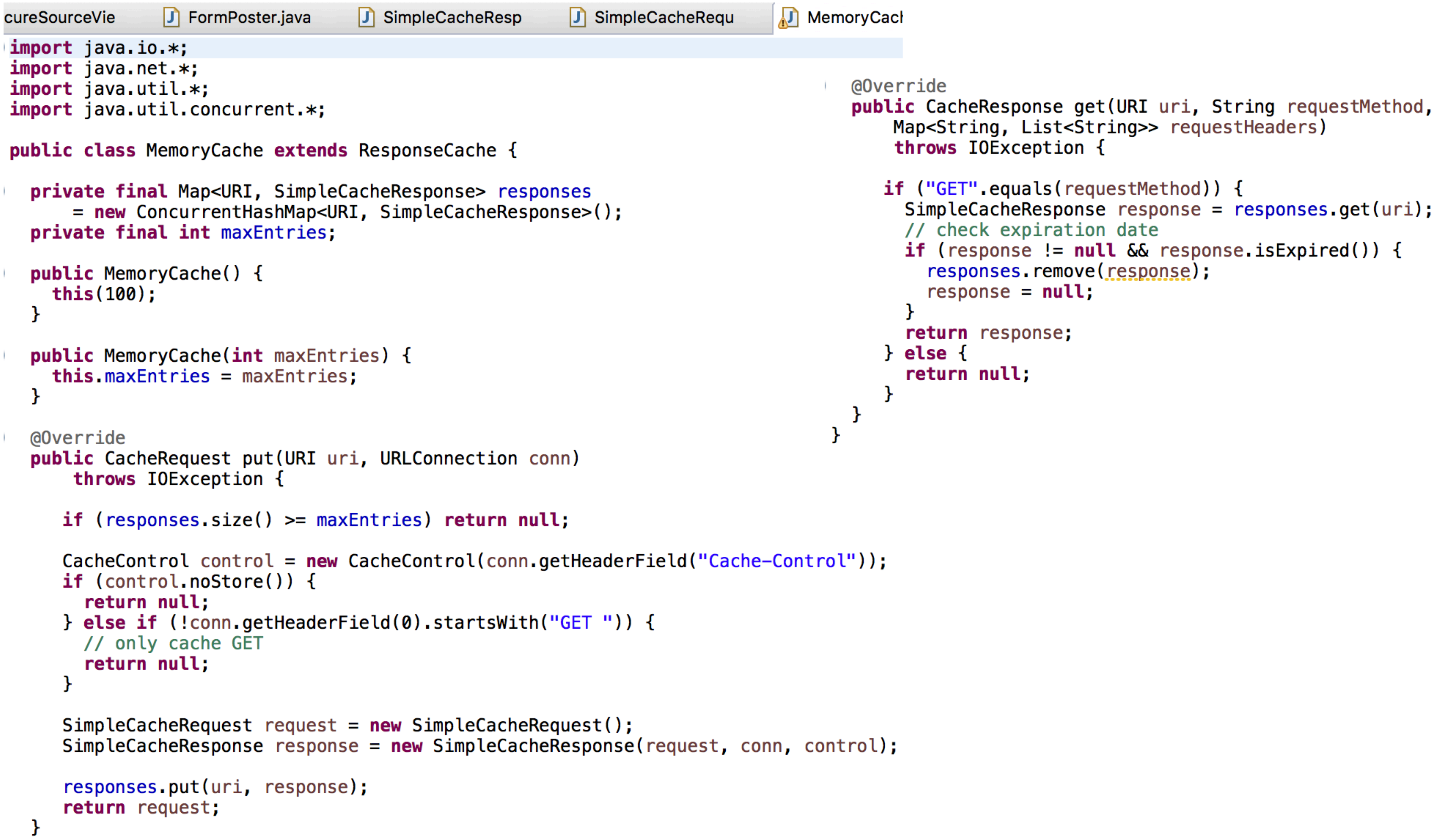
```
import java.io.*;
import java.net.*;
import java.util.*;

public class SimpleCacheResponse extends CacheResponse {

  private final Map<String, List<String>> headers;
  private final SimpleCacheRequest request;
  private final Date expires;
  private final CacheControl control;

  public SimpleCacheResponse(
      SimpleCacheRequest request, URLConnection uc, CacheControl control)
      throws IOException {

    this.request = request;
    this.control = control;
    this.expires = new Date(uc.getExpiration());
    this.headers = Collections.unmodifiableMap(uc.getHeaderFields());
  }

  @Override
  public InputStream getBody() {
    return new ByteArrayInputStream(request.getData());
  }

  @Override
  public Map<String, List<String>> getHeaders()
      throws IOException {
      return headers;
  }

  public CacheControl getControl() {
    return control;
  }

  public boolean isExpired() {
    Date now = new Date();
    if (control.getMaxAge().before(now)) return true;
    else if (expires != null && control.getMaxAge() != null) {
      return expires.before(now);
    } else {
      return false;
    }
  }
}
```

tied to a SimpleCacheRequest

return InputStream

Example: ResponseCache

store and retrieve cached values as requested



```java
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.concurrent.*;

public class MemoryCache extends ResponseCache {

  private final Map<URI, SimpleCacheResponse> responses
      = new ConcurrentHashMap<URI, SimpleCacheResponse>();
  private final int maxEntries;

  public MemoryCache() {
    this(100);
  }

  public MemoryCache(int maxEntries) {
    this.maxEntries = maxEntries;
  }

  @Override
  public CacheRequest put(URI uri, URLConnection conn)
      throws IOException {

     if (responses.size() >= maxEntries) return null;

     CacheControl control = new CacheControl(conn.getHeaderField("Cache-Control"));
     if (control.noStore()) {
       return null;
     } else if (!conn.getHeaderField(0).startsWith("GET ")) {
       // only cache GET
       return null;
     }

     SimpleCacheRequest request = new SimpleCacheRequest();
     SimpleCacheResponse response = new SimpleCacheResponse(request, conn, control);

     responses.put(uri, response);
     return request;
  }
```

```java
  @Override
  public CacheResponse get(URI uri, String requestMethod,
      Map<String, List<String>> requestHeaders)
      throws IOException {

    if ("GET".equals(requestMethod)) {
      SimpleCacheResponse response = responses.get(uri);
      // check expiration date
      if (response != null && response.isExpired()) {
        responses.remove(response);
        response = null;
      }
      return response;
    } else {
      return null;
    }
  }
}
```

Installing/Changing Cache

- Java only allows one URL cache at a time
- Two methods
  - public static ResponseCache getDefault()
  - public static void setDefault(ResponseCache responseCache)
    - set the single cache used by all programs running within the same Java virtual machine
- Example
  - ResponseCache.setDefault(new MemoryCache());
    - HTTP URLConnections always use MemoryCache

Configuring Connections

Fields

- Fields that define how client makes request to server
  - protected URL url;
  - protected boolean doInput = true;
  - protected boolean doOutput = false;
    - if true, can write to and read from server
  - protected boolean allowUserInteraction = defaultAllowUserInteraction;
    - e.g., id/passwd requested
  - protected boolean useCaches = defaultUseCaches;
  - protected long ifModifiedSince = 0;
  - protected boolean connected = false;
    - cannot be set explicitly and accessed

modify these fields only before connection is established

Setter/Getter Methods

- public URL getURL()
- public void setDoInput(boolean doInput)
- public boolean getDoInput()
- public void setDoOutput(boolean doOutput)
- public boolean getDoOutput()
- public void setAllowUserInteraction(boolean allowUserInteraction)
- public boolean getAllowUserInteraction()
- public void setUseCaches(boolean useCaches)
- public boolean getUseCaches()
- public void setIfModifiedSince(long ifModifiedSince)
- public long getIfModifiedSince()

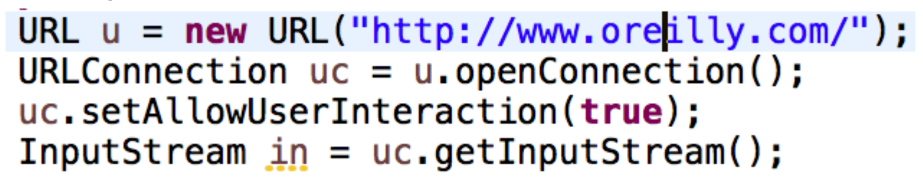
Setter/Getter for Default Behavior

- public void setDefaultUseCaches(boolean useCaches)
- public boolean getDefaultUseCaches()
- public static void setDefaultAllowUserInteraction(boolean allowUserInteraction)
- public static boolean getDefaultAllowUserInteraction()
- public static FileNameMap getFileNameMap()
- public static void setFileNameMap(FileNameMap map)

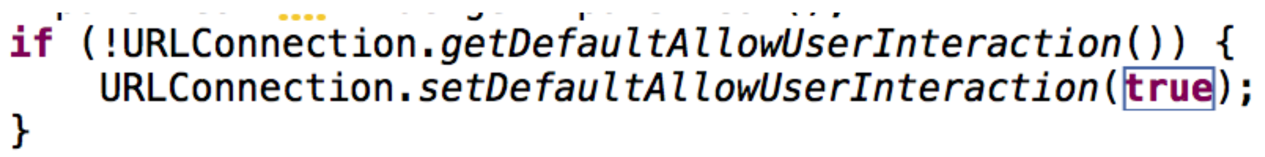
- These methods can be invoked at any time
  - new defaults apply to only new URLConnection objects

User Interaction

- protected boolean allowUserInteraction
  - specify whether user interaction is allowed
  - true: user interaction allowed
  - default: false
- Example



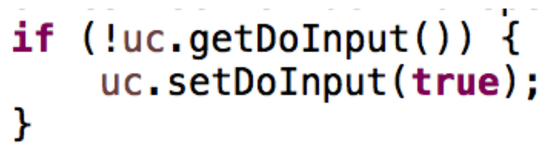
```
URL u = new URL("http://www.oreilly.com/");
URLConnection uc = u.openConnection();
uc.setAllowUserInteraction(true);
InputStream in = uc.getInputStream();
```



```
if (!URLConnection.getDefaultAllowUserInteraction()) {
    URLConnection.setDefaultAllowUserInteraction(true);
}
```

Reading

- protected boolean doInput
    - specify whether URLConnection can be used for reading from server
    - true: URLConnection is used for reading
    - default: true
- Example



```
if (!uc.getDoInput()) {
    uc.setDoInput(true);
}
```

Reading + Writing

- protected boolean doOutput
  - true: URLConnection can be used for writing
  - default: false

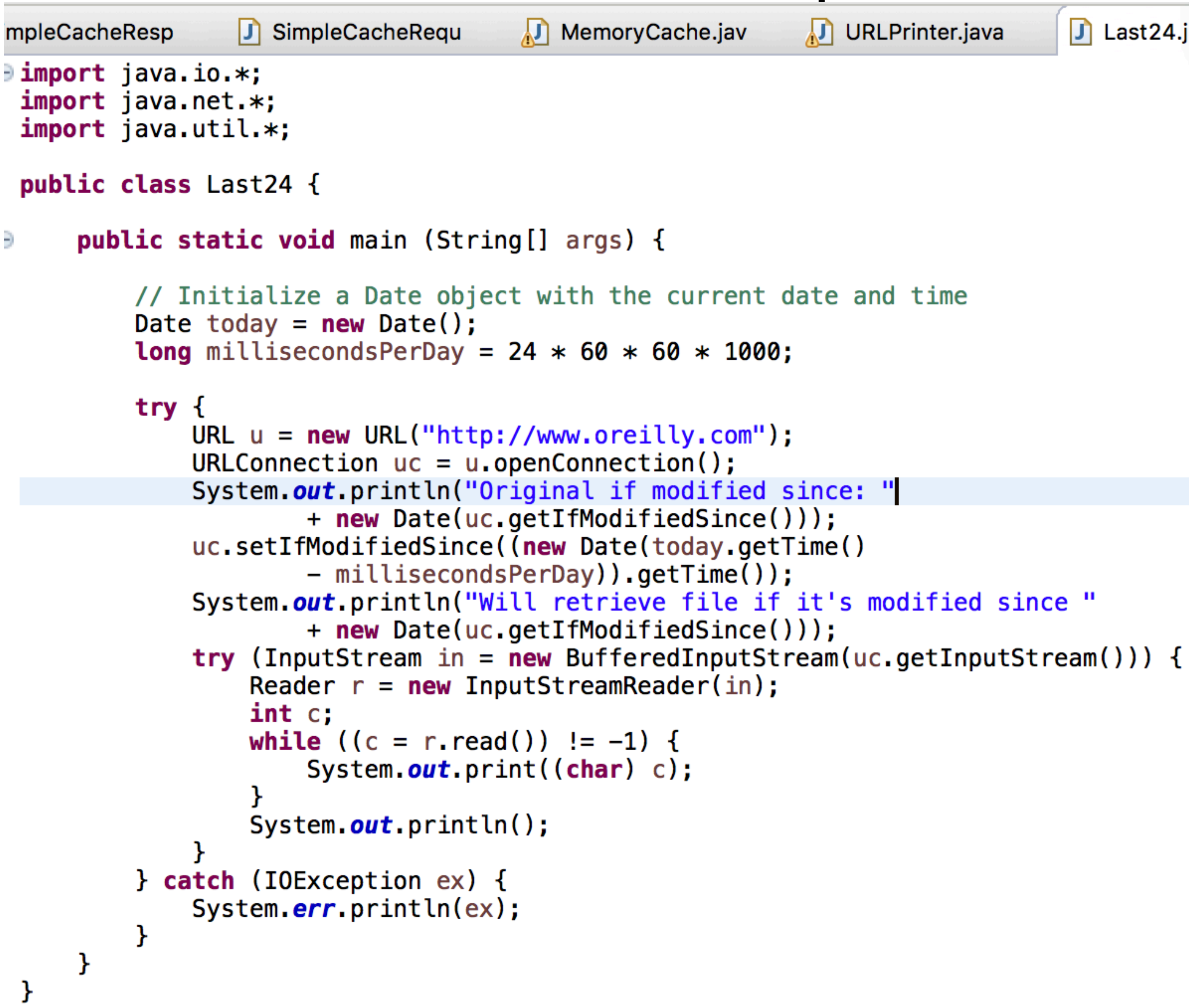
- Example

```
if (!uc.getDoOutput()) {
    uc.setDoOutput(true);
}
```

If-Modified-Since

- HTTP field
  - If-Modified-Since: Fri, 31 Oct 2014 19:22:07 GMT
    - if the document has changed since that time, server should send it
    - otherwise, it should not (client reads from cache)
  - HTTP/1.1 304 Not Modified
    - document has not changed since the time client provided in header
- protected long ifModifiedSince
  - specify the date (milliseconds since midnight, Greenwich Mean Time, 1/1, 1970), which will be placed in the If-Modified-Since header field

Example



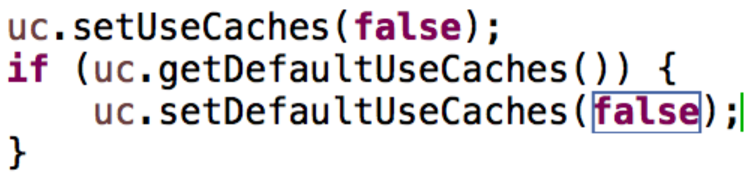
```
import java.io.*;
import java.net.*;
import java.util.*;

public class Last24 {

    public static void main (String[] args) {

        // Initialize a Date object with the current date and time
        Date today = new Date();
        long millisecondsPerDay = 24 * 60 * 60 * 1000;

        try {
            URL u = new URL("http://www.oreilly.com");
            URLConnection uc = u.openConnection();
            System.out.println("Original if modified since: "
                    + new Date(uc.getIfModifiedSince()));
            uc.setIfModifiedSince((new Date(today.getTime()
                    - millisecondsPerDay)).getTime());
            System.out.println("Will retrieve file if it's modified since "
                    + new Date(uc.getIfModifiedSince()));
            try (InputStream in = new BufferedInputStream(uc.getInputStream())) {
                Reader r = new InputStreamReader(in);
                int c;
                while ((c = r.read()) != -1) {
                    System.out.print((char) c);
                }
                System.out.println();
            }
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

Exercise

- Change reader to BufferedReader and use readLine() method
- Try to estimate the last time the content of "https://www.oreilly.com" changed

Using Caches

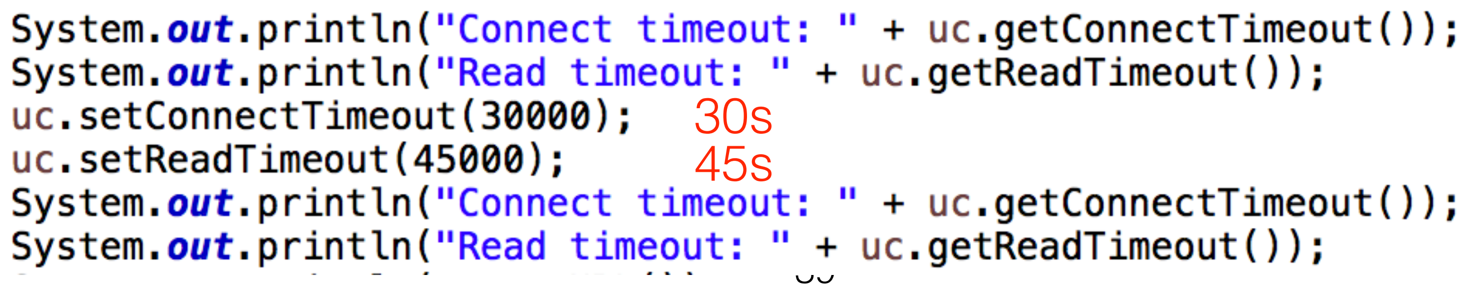
- protected boolean useCaches
  - determine whether cache will be used if it's available
  - default: true
- Example



```
uc.setUseCaches(false);
if (uc.getDefaultUseCaches()) {
    uc.setDefaultUseCaches(false);
}
```

Timeouts

- How long the underlying socket will wait for a response from remote end host before throwing a SocketTimeoutException
  - setConnectTimeout(int t): if timeout expires before connection is established, SocketTimeoutException is thrown
  - setReadTimeout(int t): if time expires before there is data available for read, SocketTimeoutException is thrown
  - 0 interpreted as no timeout
- Example



```
System.out.println("Connect timeout: " + uc.getConnectTimeout());
System.out.println("Read timeout: " + uc.getReadTimeout());
uc.setConnectTimeout(30000);    30s
uc.setReadTimeout(45000);       45s
System.out.println("Connect timeout: " + uc.getConnectTimeout());
System.out.println("Read timeout: " + uc.getReadTimeout());
```

Changing Header Fields

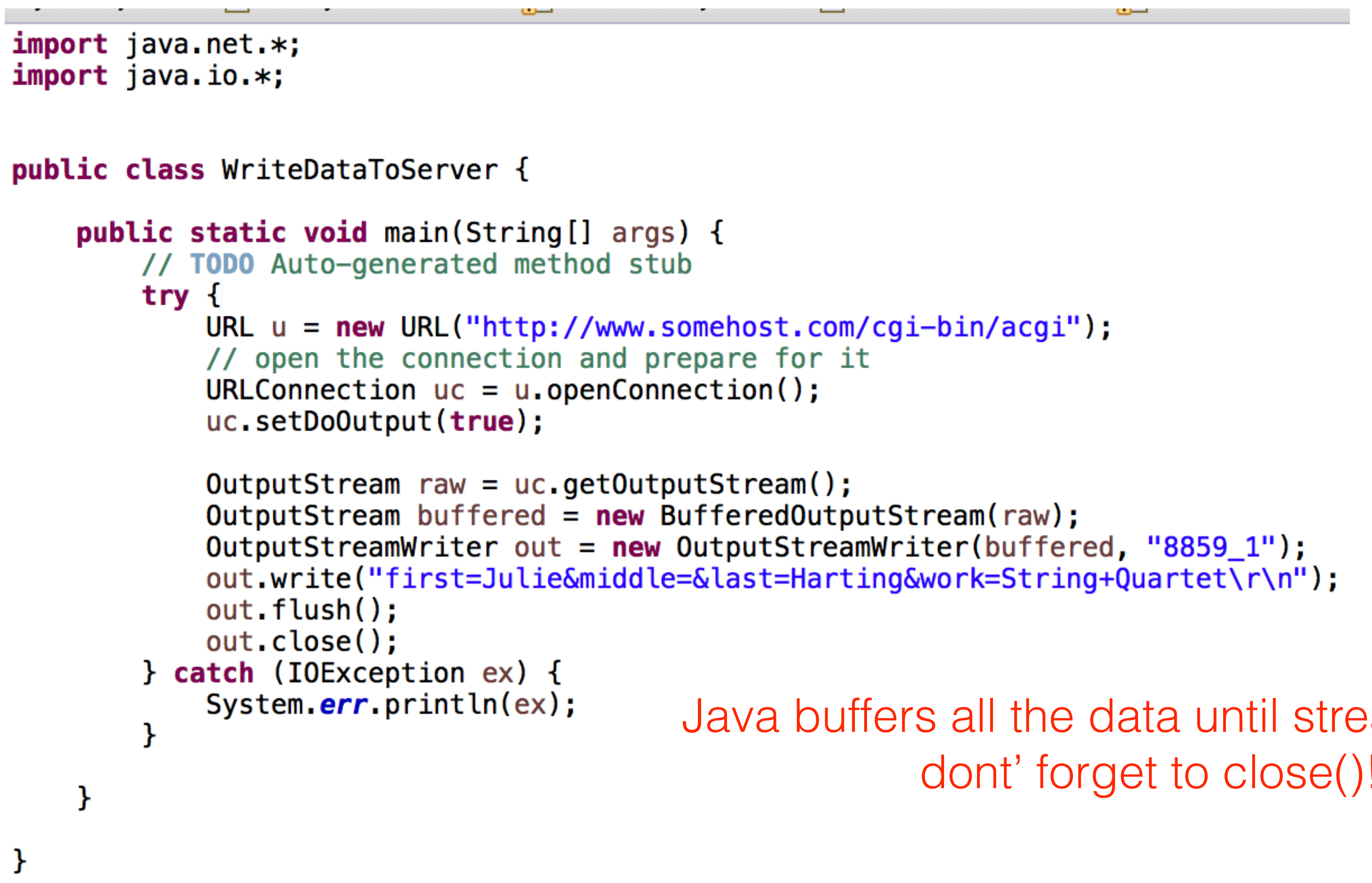
- HTTP has nearly no restrictions on header field names and values
  - no white space in name
  - no line breaks in value
  - this all depends on server
- Four key methods
  - public void setRequestProperty(String name, String value)
    - add new or change existing field
  - public void addRequestProperty(String name, String value)
    - add
  - public String getRequestProperty(String name)
  - public Map<String, List<String>> getRequestProperties()

<u>Writing Data to Server</u>

Writing Data to Server

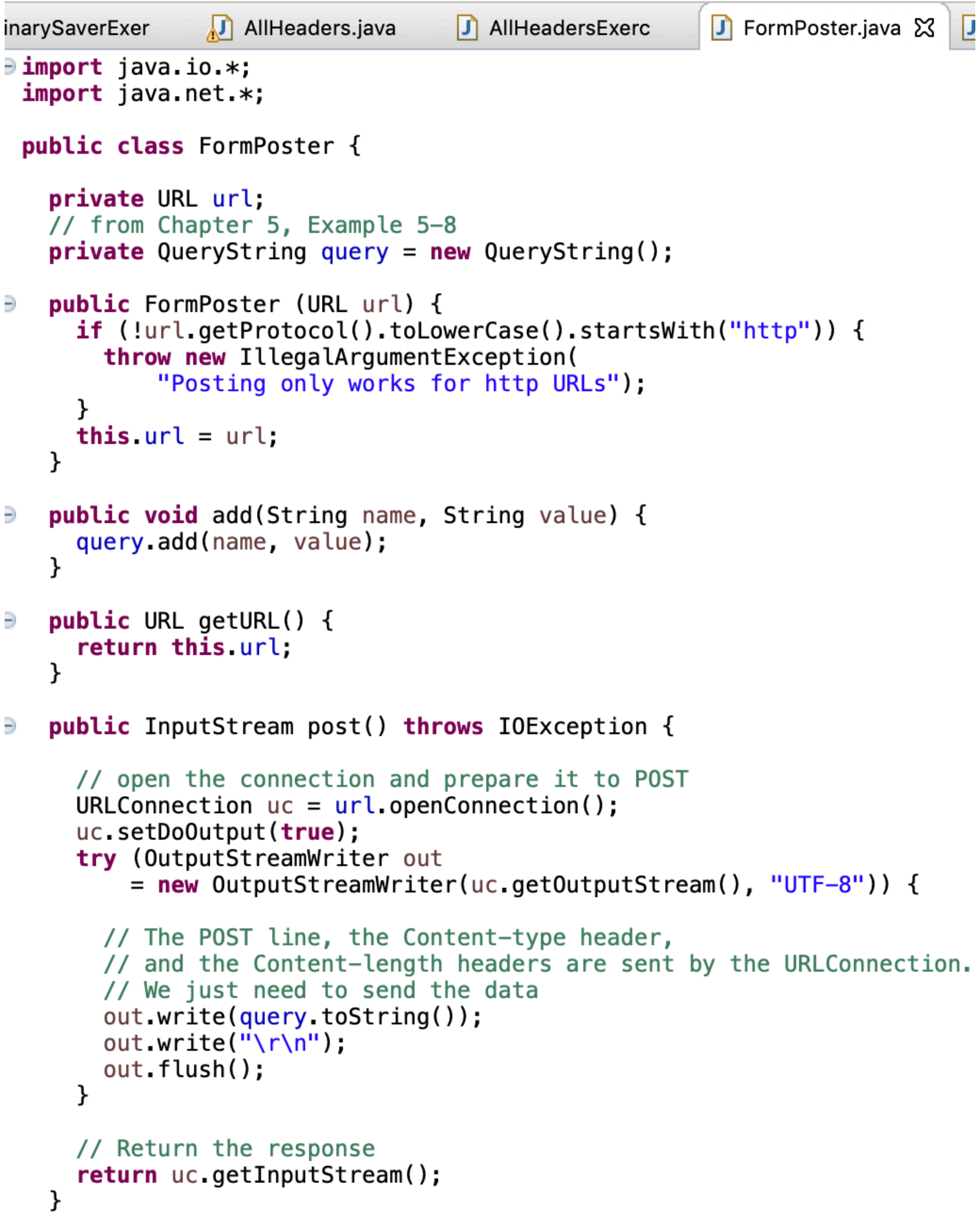
- Two ways
  - POST: submitting a form
  - PUT: uploading a file
    - supported by HttpURLConnection (subclass of URLConnection)
- Key methods
  - public OutputStream getOutputStream()
    - return output stream to which data can be written
  - public void setDoOutput (boolean b)
    - URLConnection can be used for writing if b==true

Example



```java
import java.net.*;
import java.io.*;


public class WriteDataToServer {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            URL u = new URL("http://www.somehost.com/cgi-bin/acgi");
            // open the connection and prepare for it
            URLConnection uc = u.openConnection();
            uc.setDoOutput(true);

            OutputStream raw = uc.getOutputStream();
            OutputStream buffered = new BufferedOutputStream(raw);
            OutputStreamWriter out = new OutputStreamWriter(buffered, "8859_1");
            out.write("first=Julie&middle=&last=Harting&work=String+Quartet\r\n");
            out.flush();
            out.close();
        } catch (IOException ex) {
            System.err.println(ex);
        }

    }

}
```

Java buffers all the data until stream is closed
dont' forget to close()!!



```
import java.io.*;
import java.net.*;

public class FormPoster {

  private URL url;
  // from Chapter 5, Example 5-8
  private QueryString query = new QueryString();

  public FormPoster (URL url) {
    if (!url.getProtocol().toLowerCase().startsWith("http")) {
      throw new IllegalArgumentException(
          "Posting only works for http URLs");
    }
    this.url = url;
  }

  public void add(String name, String value) {
    query.add(name, value);
  }

  public URL getURL() {
    return this.url;
  }

  public InputStream post() throws IOException {

    // open the connection and prepare it to POST
    URLConnection uc = url.openConnection();
    uc.setDoOutput(true);
    try (OutputStreamWriter out
        = new OutputStreamWriter(uc.getOutputStream(), "UTF-8")) {

      // The POST line, the Content-type header,
      // and the Content-length headers are sent by the URLConnection.
      // We just need to send the data
      out.write(query.toString());
      out.write("\r\n");
      out.flush();
    }

    // Return the response
    return uc.getInputStream();
  }
```

```java
  public static void main(String[] args) {
    URL url;
    if (args.length > 0) {
      try {
        url = new URL(args[0]);
      } catch (MalformedURLException ex) {
        System.err.println("Usage: java FormPoster url");
        return;
      }
    } else {
      try {
        url = new URL(
            "http://www.cafeaulait.org/books/jnp4/postquery.phtml");
      } catch (MalformedURLException ex) { // shouldn't happen
        System.err.println(ex);
        return;
      }
    }

    FormPoster poster = new FormPoster(url);
    poster.add("name", "Elliotte Rusty Harold");
    poster.add("email", "elharo@ibiblio.org");

    try (InputStream in = poster.post()) {
      // Read the response
      Reader r = new InputStreamReader(in);
      int c;
      while((c = r.read()) != -1) {
        System.out.print((char) c);
      }
      System.out.println();
    } catch (IOException ex) {
      System.err.println(ex);
    }
  }
}
```

HttpURLConnection

HttpURLConnection

Abstract class extending URLConnection

- Provide methods to
  - get and set HTTP request method
  - decide whether to follow redirects
  - get response code and message
  - figure out whether proxy server is used
- Contain several mnemonic constants representing HTTP response codes
- Constructing class instance

```
URL u = new URL("http://lesswrong.com/");
URLConnection uc = u.openConnection();
HttpURLConnection http = (HttpURLConnection) uc;
```

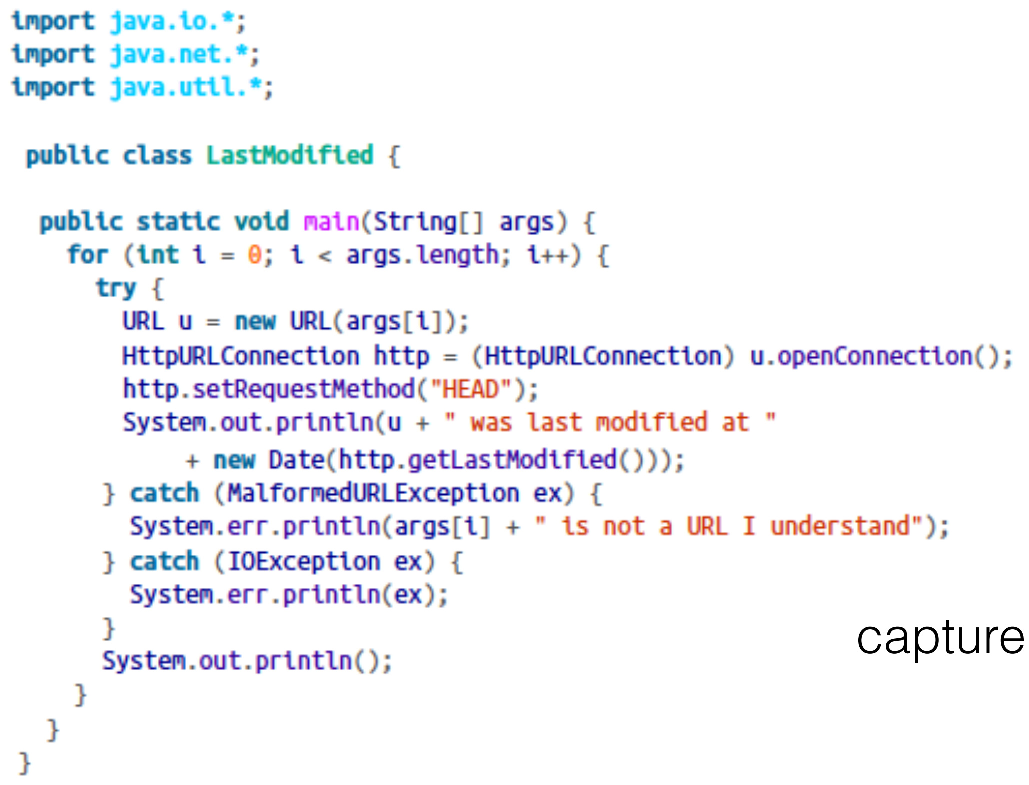
Setting Request Method

- Method
  - public void setRequestMethod(String method) throws ProtocolException
- Seven (case-sensitive) strings method
  - GET: default
  - POST
  - HEAD
  - PUT
  - DELETE
  - OPTIONS
  - TRACE

CoIn LAB

HEAD

- Tell server to only send header
- Example



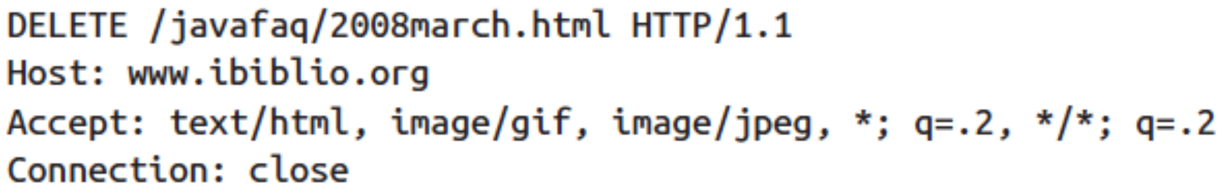
```java
import java.io.*;
import java.net.*;
import java.util.*;

 public class LastModified {

  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        HttpURLConnection http = (HttpURLConnection) u.openConnection();
        http.setRequestMethod("HEAD");
        System.out.println(u + " was last modified at "
             + new Date(http.getLastModified()));
      } catch (MalformedURLException ex) {
        System.err.println(args[i] + " is not a URL I understand");
      } catch (IOException ex) {
        System.err.println(ex);
      }
      System.out.println();
    }
  }
}
```

capture packet to see the difference

- You can do this with GET method as well. What's the difference?

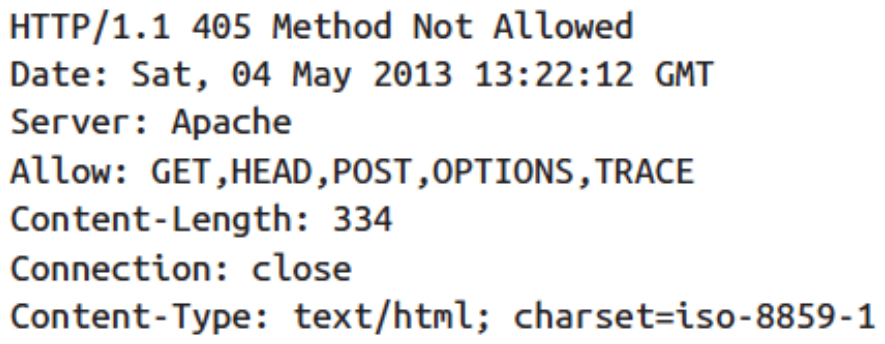
DELETE

- Remove a file
- Not all servers are configured to support
- Some sort of authentication needed
- Example
  - request



```
DELETE /javafaq/2008march.html HTTP/1.1
Host: www.ibiblio.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: close
```

  - response



```
HTTP/1.1 405 Method Not Allowed
Date: Sat, 04 May 2013 13:22:12 GMT
Server: Apache
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 334
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Modify "LastModified.java" to test DELETE

PUT

- Upload a file
- Example: HTML editor putting a file on a server

```
PUT /blog/wp-app.php/service/pomdoros.html HTTP/1.1
Host: www.elharo.com
Authorization: Basic ZGFmZnk6c2VjZXJldA==
Content-Type: application/atom+xml;type=entry
Content-Length: 329
If-Match: "e180ee84f0671b1"
```

OPTIONS

- Ask server what options are supported
- Example
  - request

```
OPTIONS /xml/ HTTP/1.1
Host: www.ibiblio.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: close
```

  - response

```
HTTP/1.1 200 OK
Date: Sat, 04 May 2013 13:52:53 GMT
Server: Apache
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Style-Type: text/css
Content-Length: 0
Connection: close
Content-Type: text/html; charset=utf-8
```

Disconnecting from Server

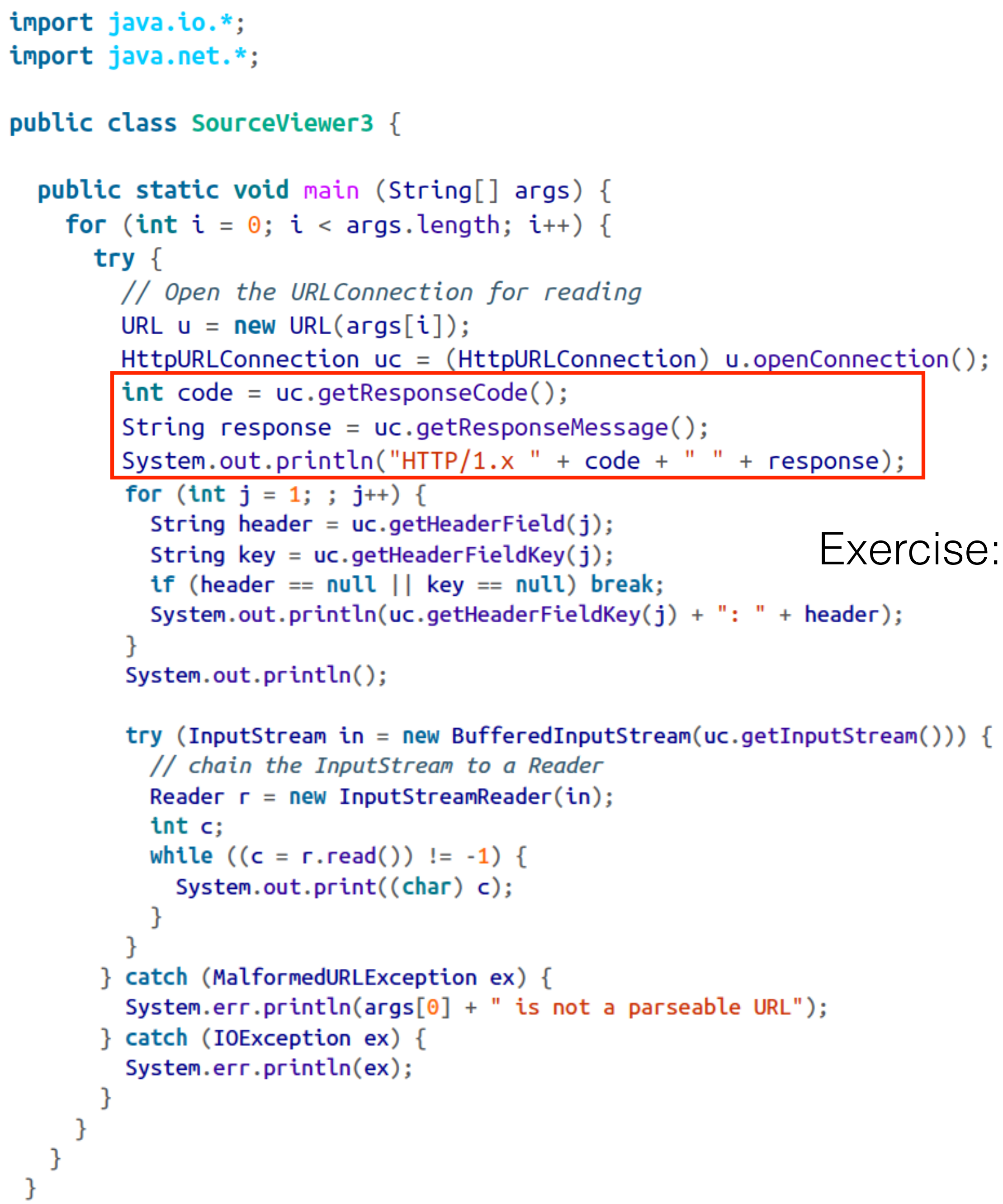
- Disconnect from server when
  - you do not want persistent connection
  - you want to disconnect upon completed transmission
- Method
  - public abstract void disconnect()

Handling Server Responses

- Methods
  - `public int getResponseCode() throws IOException`
  - `public String getResponseMessage() throws IOException`

- HTTP response codes in Java
  - HttpURLConnection.OK
  - HttpURLConnection.NOT_FOUND, …

Example



```java
import java.io.*;
import java.net.*;

public class SourceViewer3 {

  public static void main (String[] args) {
    for (int i = 0; i < args.length; i++) {
      try {
        // Open the URLConnection for reading
        URL u = new URL(args[i]);
        HttpURLConnection uc = (HttpURLConnection) u.openConnection();
        int code = uc.getResponseCode();
        String response = uc.getResponseMessage();
        System.out.println("HTTP/1.x " + code + " " + response);
        for (int j = 1; ; j++) {
          String header = uc.getHeaderField(j);
          String key = uc.getHeaderFieldKey(j);
          if (header == null || key == null) break;
          System.out.println(uc.getHeaderFieldKey(j) + ": " + header);
        }
        System.out.println();

        try (InputStream in = new BufferedInputStream(uc.getInputStream())) {
          // chain the InputStream to a Reader
          Reader r = new InputStreamReader(in);
          int c;
          while ((c = r.read()) != -1) {
            System.out.print((char) c);
          }
        }
      } catch (MalformedURLException ex) {
        System.err.println(args[0] + " is not a parseable URL");
      } catch (IOException ex) {
        System.err.println(ex);
      }
    }
  }
}
```

Exercise: add a code printing HTTP version

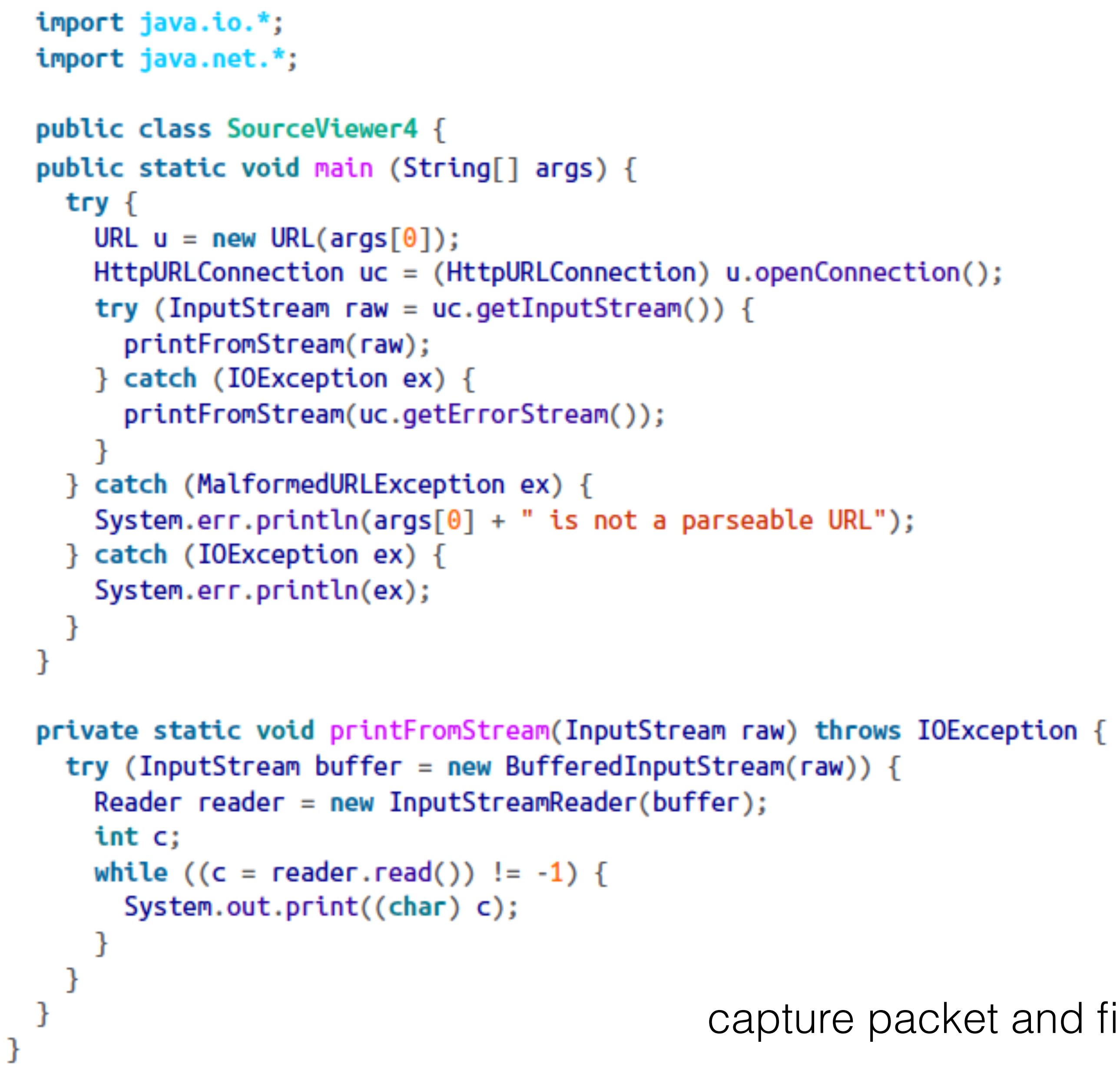
Error Conditions

- 404 NotFound response can be delivered with a page helping user about missing page
- Method to get the help page

```
public InputStream getErrorStream()
```

  - return InputStream containing help page or null if no error encountered or no data returned
- Usage
  - invoke getErrorStream() inside catch block after getInputStream() has failed

Example



```java
  import java.io.*;
  import java.net.*;

  public class SourceViewer4 {
  public static void main (String[] args) {
    try {
      URL u = new URL(args[0]);
      HttpURLConnection uc = (HttpURLConnection) u.openConnection();
      try (InputStream raw = uc.getInputStream()) {
        printFromStream(raw);
      } catch (IOException ex) {
        printFromStream(uc.getErrorStream());
      }
    } catch (MalformedURLException ex) {
      System.err.println(args[0] + " is not a parseable URL");
    } catch (IOException ex) {
      System.err.println(ex);
    }
  }

  private static void printFromStream(InputStream raw) throws IOException {
    try (InputStream buffer = new BufferedInputStream(raw)) {
      Reader reader = new InputStreamReader(buffer);
      int c;
      while ((c = reader.read()) != -1) {
        System.out.print((char) c);
      }
    }
  }
}
```

capture packet and find response code 404