URL IO

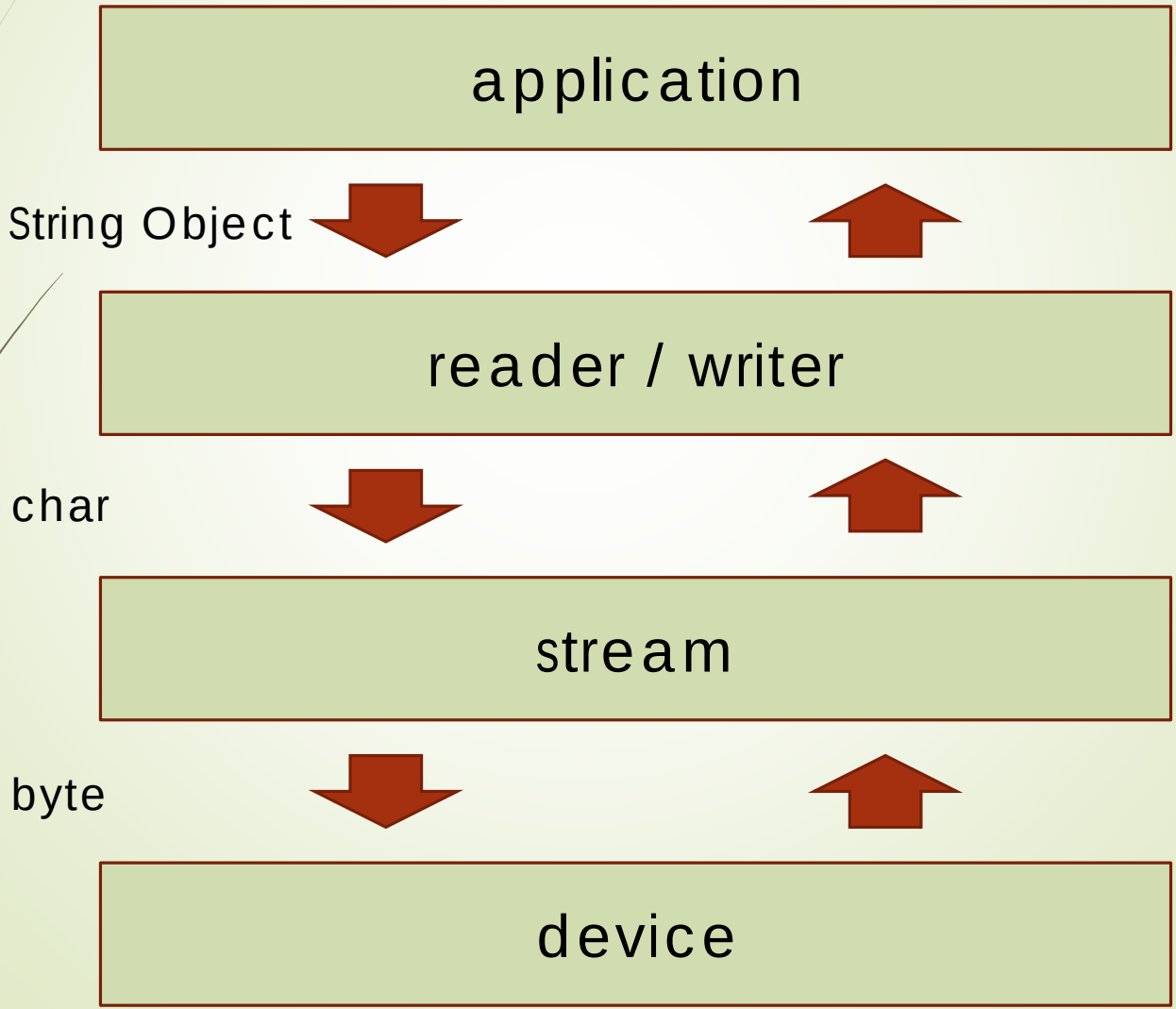
オブジェクト指向プログラミング特論

2018年度

只木進一：工学系研究科

ネットワークへのアクセス

- ネットワークへの接続
  - TCP：Socket利用
  - UDP：DatagramSocket利用
- URLへのアクセス


application
String Object
reader / writer
char
stream
byte
device

階層化されたIOの利点

- アプリケーションからは、ReaderやWriterあるいはstreamを操作する
- デバイスはファイルに限らない
  - キーボード・ディスプレイ
  - ネットワークを隔てた遠隔システム

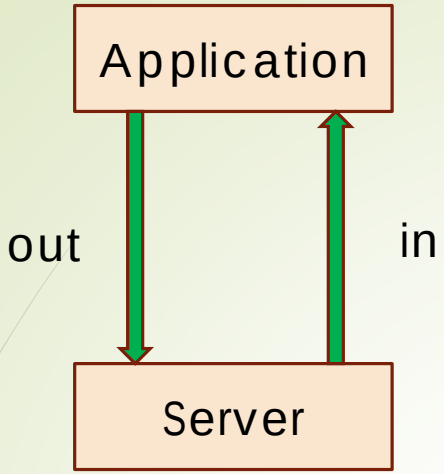
リモートサーバへの接続

- ソケットを開く(TCP通信)

```
Socket server = new Socket(serverAddress, port);
```

- Reader/Writerを開く

```
BufferedReader in = new BufferedReader(
  new inputStreamReader(server.getInputStream()));
PrintWriter pout =
  new PrintWriter(server.getOutputStream(), true);
```




```
public static void main(String[] args) throws IOException {
  Socket server = new Socket("aoba.cc.saga-u.ac.jp", 80);
  try (BufferedReader in = new BufferedReader(
        new InputStreamReader(server.getInputStream()))) {
    PrintWriter out = new PrintWriter(server.getOutputStream(), true);
    out.println("GET /");
    String line;
    while ((line = in.readLine()) != null) {
      System.out.println(line);
    }
  }
}
```

HTTP(Hypertext Transfer Protocol)の仕組み

- サーバの80番ポートへ接続する
- GETコマンドでページを要求する
- サーバから返信が来る
  - 正しい返信：200
  - エラーがあった場合
    - 403 : Forbidden
    - 404 : Not Found
    - エラーを表示するHTMLファイル

javaでのHTTP接続

- URLクラス

URL url = new URL(urlString)

- HttpURLConnectionクラス

HttpURLConnection c =
(HttpURLConnection)url.openConnection();

- HTTPステータスコード

int code = urlConnection.getResponseCode();

- コンテンツタイプ

String type = urlConnection.getContentType()

- ヘッダ情報
  - metaタグで記述されたもの

Map<String, List<String>> headerFields = rlConnection.getHeaderFields()

HTMLファイルの読み込み

- Readerとして

```
BufferedReader reader = new BufferedReader(
new InputStreamReader(urlConnection.getInputStream()))
```

- 一行毎に読み込み、タグを分析
  - 正規表現の応用問題

HTMLタグの分析例

- タイトルを取得

```
String tp = "<title>([^<]+)</title>";
Pattern pattern = Pattern.compile(tp,
      Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);
Matcher m = pattern.matcher(htmlContent);
if (m.find()) {
   return m.group(1);
}
```

  - 複数行に分割されている場合への対応
  - タグの大文字小文字を無視

例：ヘッダの切り出し

```
Pattern pattern = Pattern.compile("<h(¥¥d+)>([^<]+)</h(¥¥d+)>",
        Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);
  Matcher m = pattern.matcher(htmlContent);
  while (m.find()) {
     int level = Integer.valueOf(m.group(1));
     String title = m.group(2);
     HTMLHeader header = new HTMLHeader(level);
     header.setTitle(title);
     headerList.add(header);
     n++;
  }
```

Simplest.java

```
package URL;

import java.io.*;
import java.net.Socket;

/**
 *
 * @author tadaki
 */
public class Simplest {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        Socket server = new Socket("aoba.cc.saga-u.ac.jp", 80);
        try (BufferedReader in = new BufferedReader(
                new InputStreamReader(server.getInputStream()))) {
            PrintWriter out = new PrintWriter(server.getOutputStream(),
true);
            out.println("GET /");
            String line;
            while ((line = in.readLine()) != null) {
                System.out.println(line);
            }

        }
    }
}
```

URLMain.java

```java
package URL;

import java.io.IOException;
import java.util.List;

/**
 *
 * @author tadaki
 */
public class URLMain {

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {
        String urlString = "http://aoba.cc.saga-u.ac.jp/";
        ReadURL readURL = new ReadURL(urlString);
        System.out.println(readURL.showHeaderFields());
        System.out.println("title: "+readURL.getTitle());
        readURL.readHeaders();
        List<HTMLHeader> headerList = readURL.getHeaderList();
        headerList.stream().forEachOrdered(h->System.out.println(h));
    }

}
```

ReadURL.java

```
package URL;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author tadaki
 */
public class ReadURL {

    private final List<HTMLHeader> headerList;
    private final HttpURLConnection urlConnection;
    private final Map<String, List<String>> headerFields;
    public static final String nl = System.getProperty("line.separator");
    private String htmlContent = null;

    /**
     * コンストラクタ
     *
     * @param urlString URLを表す文字列
     * @throws java.io.IOException
     */
    public ReadURL(String urlString) throws IOException {
        URL url = new URL(urlString);
        urlConnection = (HttpURLConnection)url.openConnection();
        urlConnection.setRequestProperty("Accept-Language", "ja");
        headerFields = urlConnection.getHeaderFields();
        headerList = Collections.synchronizedList(new ArrayList<>());
    }

    public String showHeaderFields() {
        StringBuilder sb = new StringBuilder();
        headerFields.keySet().forEach(
                s -> {
                    sb.append(s).append("->");
                    sb.append(headerFields.get(s)).append(nl);
```

```
                });
        return sb.toString();
    }

    /**
     * text/htmlの場合にコンテンツを読み込む
     *
     * @return
     * @throws IOException
     */
    public String readPage() throws IOException {
        if (!isHTML()) {
            System.err.println("not a html file");
            return null;
        }
        int code = urlConnection.getResponseCode();
        if (code != 200) {
            System.err.println("Incorrect URL:code="+code);
            return null;
        }

        //urlをストリームとして開く
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(urlConnection.getInputStream()));
        //内容を一つの文字列として保存
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line).append(nl);
        }
        return sb.toString();
    }

    /**
     * content-typeがtext/htmlならば真
     *
     * @return
     * @throws IOException
     */
    public boolean isHTML() throws IOException {
        Pattern pattern = Pattern.compile("html", Pattern.CASE_INSENSITIVE);
        Matcher m = pattern.matcher(urlConnection.getContentType());
        return m.find();
    }

    public String getTitle() throws IOException {
```

```
        if (htmlContent == null) {
            htmlContent = readPage();
        }
        if (htmlContent == null) {
            return null;
        }
        String tp = "<title>([^<]+)</title>";
        //複数行対応、及び大文字小文字を区別しない
        Pattern pattern = Pattern.compile(tp,
                Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);
        Matcher m = pattern.matcher(htmlContent);
        if (m.find()) {
            return m.group(1);
        }
        return null;
    }

    public int readHeaders() throws IOException {
        if (htmlContent == null) {
            htmlContent = readPage();
        }
        if (htmlContent == null) {
            return 0;
        }
        int n = 0;
        //複数行にまたがる正規表現として、ヘッダを定義
        Pattern pattern = Pattern.compile("<h(¥¥d+)>([^<]+)</h(¥¥d+)>",
                Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);
        Matcher m = pattern.matcher(htmlContent);
        while (m.find()) {
            int level = Integer.valueOf(m.group(1));
            String title = m.group(2);
            HTMLHeader header = new HTMLHeader(level);
            header.setTitle(title);
            headerList.add(header);
            n++;
        }
        return n;
    }

    public List<HTMLHeader> getHeaderList() {
        return headerList;
    }

}
```

HTMLHeader.java

```java
package URL;

/**
 *
 * @author tadaki
 */
public class HTMLHeader {

    private final int level;
    private String title;

    public HTMLHeader(int level) {
        this.level = level;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }


    public String toString() {
        StringBuilder sb=new StringBuilder();
        sb.append("<H").append(level).append(">");
        sb.append(title);
        sb.append("</H").append(level).append(">");
        return sb.toString();
    }
}
```