URL IO

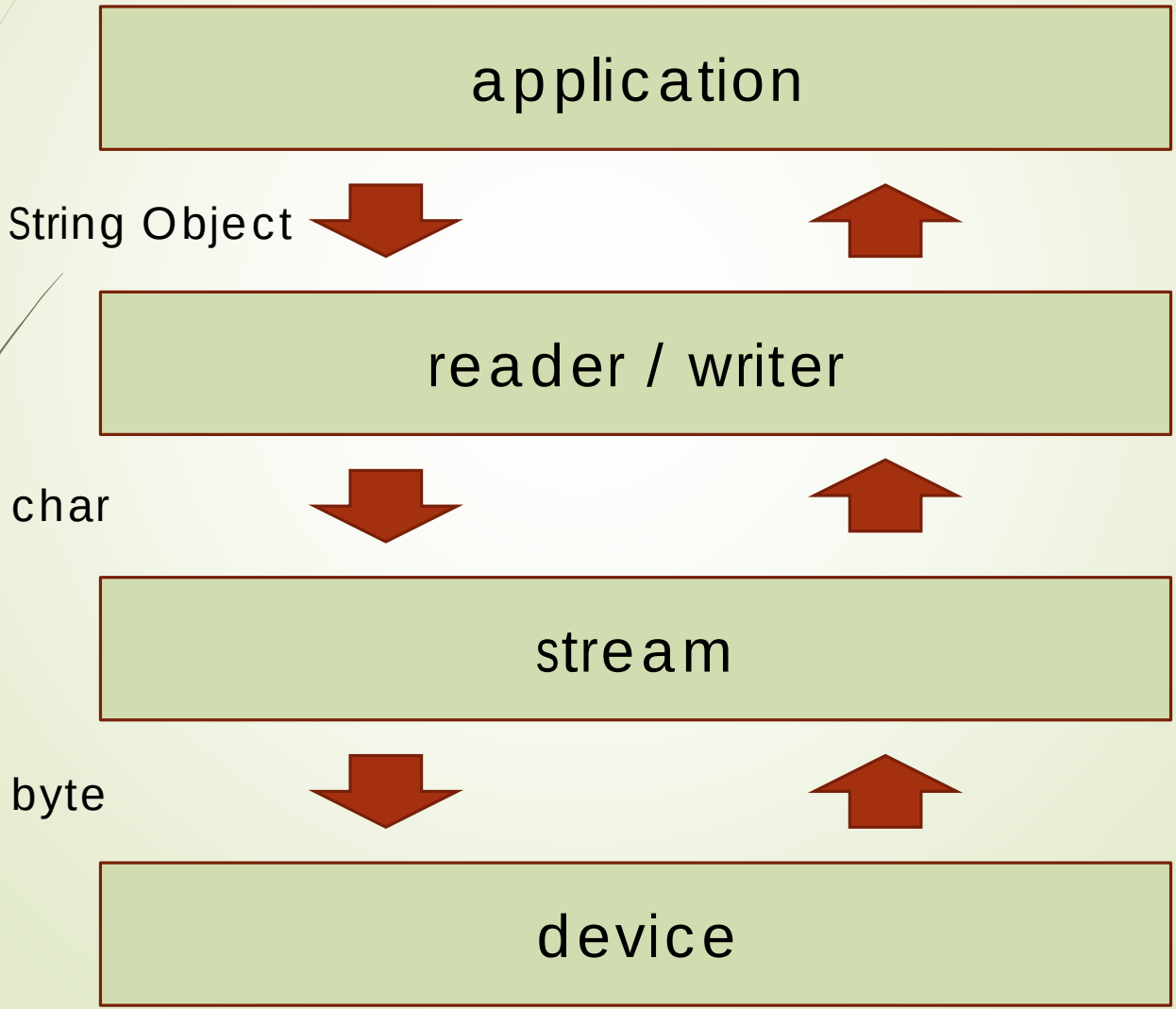
オブジェクト指向プログラミング特論

2020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
  - Uniform Resource Locator


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);
```

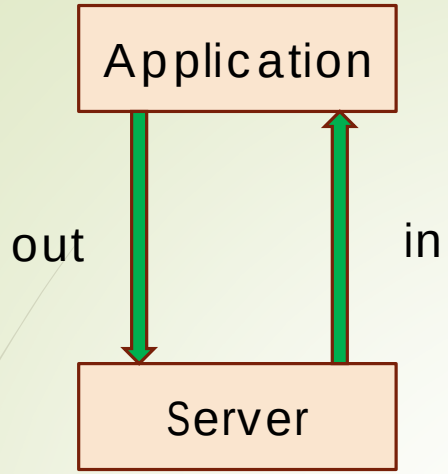
- サーバからの応答：Reader

```
BufferedReader in = new BufferedReader(
    new inputStreamReader(
        server.getInputStream()));
```

リモートサーバへの接続

- サーバへの送信：Writer

```
PrintWriter pout =
    new PrintWriter(
        server.getOutputStream(), true);
```



HTMLを読み出す例

```
public static void main(String[] args) throws IOException {
  Socket server = new Socket("aoba.cc.saga-u.ac.jp", 80);
  //サーバからの応答を得るReaderを開く
  //サーバへの送信を送るWriterを開く
  try (BufferedReader in
        = new BufferedReader(
            new InputStreamReader(server.getInputStream()));
        PrintWriter out
        = new PrintWriter(server.getOutputStream(), true)) {

     out.println("GET /");//サーバへのコマンド送信

     String line;
     //サーバからの応答を一行毎に印刷
     while ((line = in.readLine()) != null) {
        System.out.println(line);
     }
  }
}
```

FileIOSamples/URL/Simplest.java

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
  = urlConnection.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++;
  }
```