# Other IO examples
# Internet resources and Processes

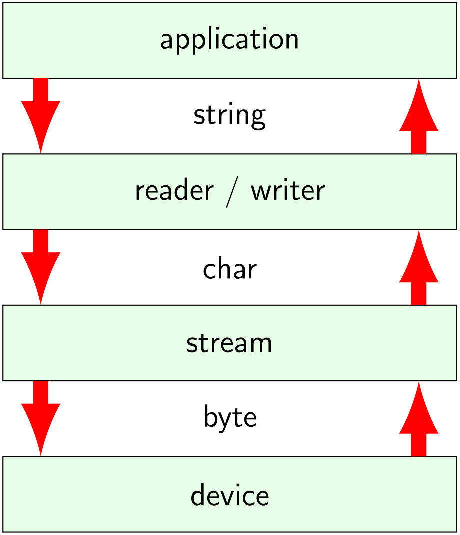
Object Oriented Programming
2022 First Semester
Shin-chi Tadaki (Saga University)

## Today's sample programs

- `https://github.com/oop-mc-saga/FileIOSamples`

The same repository of the previous

## Hierarchical structure of IO



## The previous scheme available for other resources

- Internet connections
- Web accesses
- External processes

## Connecting Internet resources

- Open TCP socket

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

- reading responses from the server

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

- sending messages to the server

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

```java
public static void main(String[] args) throws IOException {
    Socket server = new Socket("aoba.cc.saga-u.ac.jp", 80);
    //Open Reader for receiving response from server
    //Open Writer for sending message to server
    try (BufferedReader in
            = new BufferedReader(
                    new InputStreamReader(server.getInputStream()));
            PrintWriter out
            = new PrintWriter(server.getOutputStream(), true)) {

        out.println("GET /");//Send message to server

        String line;
        //Print responses from server
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }
    }
}
```

URL/Simplest.java

## Understanding HTTP (Hypertext Transfer Protocol)

- Using port 80
- Get contents by `GET` command
- Responses from the server
  - Code 200: success
  - Error codes
    - 403: access forbidden
    - 404: resource not found
    - server shows special pages for errors

## HTTP in java

- URL class

```java
URL url = new ULR(urlString);
```

- HttpURLConnection class

```java
HttpURLConnection connection
    = (HttpURLConnection)url.openConnection();
```

- HTTP status code

```java
int code = connection.getResponseCode();
```

- Content type

```java
String type = connection.getContentType();
```

- header data described by meta tags

```java
Map<String, List<String>> headerFields
    = connection.getHeaderFields();
```

## Reading HTML

- Using Reader

```
BufferedReader in = new BufferedReader(
    new InputStreamReader(connection.getInputStream())
);
```

- Reading HTML line by line
- For analyzing structure of HTML, matching patterns describing tags.

## Example: get title

```java
String tp = "<title>(.+)</title>";
//Make compatible for multilines and case insensitive
Pattern pattern = Pattern.compile(tp,
        Pattern.MULTILINE | Pattern.CASE_INSENSITIVE);
Matcher m = pattern.matcher(htmlContent);
if (m.find()) {
    return m.group(1);
}
```

- `Pattern.MULTILINE`: analyze multiple lines
- `Pattern.CASE_INSENSITIVE)`: compatible with capital and small tag characters

`URL/ReadURL.java`

## Example: get headers

```java
//Make compatible for multilines
Pattern pattern = Pattern.compile("<h(\\d+)>(.+)</h\\1>",
        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);
}
```

## Process class

- `java.lang.Process`: provide control of external processes
- Sending commands to the processes
- Receiving responses / errors from the processes

## Example: gnuplot script with data

```
set terminal pdfcairo enhanced color solid size 29cm,21cm font
↪  "Times-New-Roman" fontscale 1.2
set xrange [0:20]
set yrange [-1:1]
set title "TITLE"
set output "sample.pdf"
plot "-" title "data", sin(x) with line notitle
0.0 0.0
0.2 0.19866933079506122
0.4 0.3894183423086505
0.6000000000000001 0.5646424733950355
0.8 0.7173560908995228
#omitted below
```

- Specify the standard input as `"-"` in gnuplot scripts.
- Data can be added after the `plot` command.

## Creating process

- `ProcessBuilder`

```java
ProcessBuilder processBuilder = new ProcessBuilder("gnuplot").
directory(new File("."));
Process process = processBuilder.start();
```

- `process.getOutputStream()`: stream for sending messages
- `process.getErrorStream()`: stream for receiving errors

## Gnuplot class for drawing graph

```java
public Gnuplot() throws IOException {
    //Create new gnuplot process
    ProcessBuilder processBuilder = new ProcessBuilder("gnuplot").
            directory(new File("."));
    process = processBuilder.start();
    //writer for sending gnuplot commands
    writer = new BufferedWriter(
            new OutputStreamWriter(process.getOutputStream()));
    //reader for receiving gnuplot errors
    err = new BufferedReader(
            new InputStreamReader(process.getErrorStream()));
}
```

`process/Gnuplot.java`