

# 6. Basic I/O

## 4. Networking

# What Is a URL?

- URL is an acronym for *Uniform Resource Locator* and is a reference (an address) to a resource on the Internet
- A URL has two main components:
  - **Protocol identifier**: For the URL `http://example.com`, the protocol identifier is `http`.
  - **Resource name**: For the URL `http://example.com`, the resource name is `example.com`.

# Creating a URL

- The easiest way to create a URL object is from a String:  

```
URL myURL = new URL("http://example.com/");
```
- The URL object created above represents an *absolute URL*
- An absolute URL contains all of the information necessary to reach the resource in question

# Creating a URL Relative to Another

- A relative URL contains only enough information to reach the resource relative to another URL
- The following code creates relative URLs:

```
URL myURL = new URL("http://example.com/pages/");
```

```
URL page1URL = new URL(myURL, "page1.html");
```

```
URL page2URL = new URL(myURL, "page2.html");
```

- This code snippet uses the URL constructor that lets you create a URL object from another URL object (the base) and a relative URL specification

# URL addresses with Special characters

- Some URL addresses contain special characters, for example the space character. Like this:

`http://example.com/hello world/`

- To make these characters legal they need to be encoded before passing them to the URL constructor.

```
URL url = new URL("http://example.com/hello%20world");
```

# URI

- The `java.net.URI` class automatically takes care of the encoding characters:

```
URI uri = new URI("http", "example.com", "/hello world/", "");
```

- And then convert the URI to a URL:

```
URL url = uri.toURL();
```

# MalformedURLException

- Each of the four URL constructors throws a MalformedURLException if the arguments to the constructor refer to a null or unknown protocol:

```
try { URL myURL = new URL(...); }  
catch (MalformedURLException e) {  
    // exception handler code here  
}
```

# Reading Directly from a URL

- After you've successfully created a URL, you can call the URL's `openStream()` method to get a stream from which you can read the contents of the URL.
- The `openStream()` method returns a `java.io.InputStream` object, so reading from a URL is as easy as reading from an input stream.



# Reading Example

```
public static void main(String[] args) throws Exception {  
    URL oracle = new URL("http://www.oracle.com/");  
    BufferedReader in = new BufferedReader( new  
        InputStreamReader(oracle.openStream()));  
    String inputLine;  
    while ((inputLine = in.readLine()) != null)  
        System.out.println(inputLine);  
    in.close();  
}
```

# Connecting to a URL

- URL object's `openConnection` method allows to get a `URLConnection` object for a communication link between your Java program and the URL
- `URLConnection` has a set of protocol specific subclasses, e.g. `java.net.HttpURLConnection`

# Open Connection Example

```
try {  
    URL myURL = new URL("http://example.com/");  
    URLConnection myURLConnection =  
    myURL.openConnection();  
    myURLConnection.connect();  
}  
catch (MalformedURLException e) {  
    // new URL() failed ...  
}  
catch (IOException e) {  
    // openConnection() failed ...  
}
```

# Reading from a URLConnection

- Reading from a URLConnection instead of reading directly from a URL might be more useful: you can use the URLConnection object for other tasks (like writing to the URL) at the same time.

# Reading Example

```
public static void main(String[] args) throws Exception {  
    URL oracle = new URL("http://www.oracle.com/");  
    URLConnection yc = oracle.openConnection();  
    BufferedReader in = new BufferedReader(new  
        InputStreamReader( yc.getInputStream()));  
    String inputLine;  
    while ((inputLine = in.readLine()) != null)  
        System.out.println(inputLine);  
    in.close();  
}
```

# Exercise: Read Statistics I

- Read file from

[http://www.ukrstat.gov.ua/express/expr2012/09\\_12/234.zip](http://www.ukrstat.gov.ua/express/expr2012/09_12/234.zip)

and save it in test.zip file

# Exercise: Read Statistics II

```
public static void main(String[] args) throws  
    Exception{  
    URL expr = new  
    URL("http://www.ukrstat.gov.ua/express/expr2012/09_12/234.zip");  
    URLConnection conn = expr.openConnection();  
    InputStream in = conn.getInputStream();  
    FileOutputStream out = null;
```

# Exercise: Read Statistics III

```
try {  
    out = new FileOutputStream("test.zip");  
    int c = -1;  
    while ((c = in.read()) != -1) { out.write(c); }  
}  
finally { if (in != null) in.close();  
         if (out != null) out.close();  
         in.close();  
}  
}
```



# Exercise: Read Statistics IV

- See 641GetWebFile project for the full text.

# Providing Data to the Server

1. Create a URL.
2. Retrieve the URLConnection object.
3. Set output capability on the URLConnection.
4. Open a connection to the resource.
5. Get an output stream from the connection.
6. Write to the output stream.
7. Close the output stream.

# Manuals

- <http://docs.oracle.com/javase/tutorial/networking/TOC.html>