

Самостійна робота № 9

Тема: Пакет java.net.http

Мета: Вивчення складу та принципів застосування пакету java.net.http

Для роботи з ресурсами, які задаються адресами URL (Uniform Resource Locator — уніфікований вказівник ресурсу), у бібліотеці класів Java є достатньо розвинений клас з відповідним ім'ям URL. За допомогою визначених в ньому конструкторів та методів можна отримувати та перевіряти окремі компоненти ресурсу: протокол, адресу вузла, номер порту, ім'я файлу. Також можна відкрити потік, зв'язаний з ресурсом та прочитати ресурс для відображення, обробки або для копіювання в інший потік.

У класі URL реалізовано 6 конструкторів.

Перший конструктор створює об'єкт URL для мережевого ресурсу за адресою, що йому надається через єдиний параметр:

```
public URL(String spec);
```

У процесі створення об'єкту перевіряється наданий URL у вигляді рядка *spec*. Якщо адресу вказано невірно, то виникає виключення `MalformedURLException`. При спробі використати протокол, з яким система не може працювати, виникає те саме виключення.

Другий конструктор

```
public URL(String protocol, String host, int port, String file);
```

дозволяє окремо задавати протокол *protocol*, адресу вузла *host*, номер порту *port*, а також ім'я файлу *file*, на основі яких створює об'єкт. Значення *host* може бути задано або як ім'я, або його IP-адресою. У разі використання адреси IPv6, то вона розміщується у квадратні дужки [], відповідно до RFC 2732, або у відповідності до RFC 2373. При задані у якості номеру порту -1 буде використовуватись порт за замовчуванням для заданого протоколу.

Цей конструктор не виконує перевірку введених даних.

Наступний варіант конструктору при створенні відповідного об'єкту дозволяє використання номеру порту за замовчуванням:

```
public URL(String protocol, String host, String file);
```

Цей конструктор подібний попередньому конструктору, якщо в ньому аргумент порту задано як -1.

Ще один конструктор цього класу

```
public URL(String protocol, String host, int port, String file, URLStreamHandler handler);
```

створює об'єкт URL із зазначеного протоколу *protocol*, вузла *host*, номера порту *port*, файлу *file* та обробника *handler*. Якщо вказати номер порту -1, це означає, що URL-адреса має використовувати порт за замовчуванням для протоколу. Зазначення обробника null вказує на те, що URL-адреса має використовувати обробник потоку за замовчуванням для протоколу. Цей конструктор також не виконує перевірку введених даних.

Наступний конструктор дозволяє вказувати контексту адреси URL та її символічний запис *spec*:

```
public URL(URL context, String spec);
```

Об'єкт URL створюється відповідно до RFC2396 і має загальний вигляд:

```
<scheme>://<authority><path>?<query>#<fragment>
```

Якщо компонент *<scheme>* визначено в наведеній специфікації і не відповідає *<scheme>* контексту, то новий URL створюється як абсолютна URL-адреса тільки на основі цієї специфікації. Інакше компонент *<scheme>* успадковується від контекстної URL-адреси. Якщо компонент *<authority>* присутній у специфікації, тоді специфікація розглядається як абсолютна, а специфікації *<authority>* та *<path>* замінюють *<authority>* та *<path>* контексту. Якщо компонент *<authority>* відсутній у специфікації, *<authority>* нового URL буде успадковано із контексту.

Якщо компонент специфікації *<path>* починається із символу косої риски «/», тоді *path* розглядається як абсолютний, а *<path>* специфікації замінює контекстний *<path>*. В іншому випадку *<path>* розглядається як відносний *path* і додається до контекстного *<path>*, як описано у RFC2396. Крім того, у цьому випадку *path* канонізується шляхом видалення змін до каталогу, внесених через «..» і «.».

Конструктор

```
public URL(URL context, String spec, URLStreamHandler handler);
```

створює об'єкт URL із аналізом заданої специфікації за допомогою вказаного обробника в межах зазначеного контексту. Якщо обробник має значення null, розбір відбувається як у конструкторі з двома аргументами.

У класі URL значна кількість різноманітних методів, серед яких можна виділити наступні.

Метод `openStream` дозволяє створювати вхідний потік для читання файлу ресурсу, який зв'язано із створеним об'єктом класу URL:

```
public final InputStream openStream();
```

Для виконання операції читання із створеного таким чином потоку можна використовувати метод `read`, який визначено у класі `InputStream`.

За допомогою методу `getHost` можна визначити ім'я вузла, що відповідає об'єкту URL, для якого він викликається:

```
public String getHost();
```

Метод `getFile` дозволяє отримати ім'я файлу, який зв'язано із об'єктом `URL`:

```
public String getFile();
```

Метод `getPort` призначено для визначення номеру порту, через який виконується зв'язок до об'єкту `URL`:

```
public int getPort();
```

За допомогою метода `getProtocol` визначається протокол, за допомогою якого встановлюється з'єднання з ресурсом, що задано об'єктом `URL`:

```
public String getProtocol();
```

Метод `getRef` повертає текстовий рядок посилання на ресурс, що відповідає об'єкту `URL`:

```
public String getRef();
```

Метод `hashCode` повертає hash-код об'єкту `URL`:

```
public int hashCode();
```

Метод `equals` використовується для визначення ідентичності адрес `URL`, які задано двома об'єктами класу `URL`:

```
public boolean equals(Object obj);
```

Якщо адреси `URL` є ідентичними, то метод `equals` поверне значення `true`, а якщо ні, то значення `false`.

Метод `toExternalForm` повертає текстовий рядок зовнішнього представлення адреси `URL`, що визначається об'єктом класу `URL`:

```
public String toExternalForm();
```

Метод `toString` повертає текстовий рядок, що представляє об'єкт класу `URL`:

```
public String toString();
```

Метод `openConnection` призначено для створення каналу між додатком та мережевим ресурсом, який представлено об'єктом класу `URL`:

```
public URLConnection openConnection();
```

Нижче наведено приклад використання конструкторів та методів класу `URL`.

Приклад 1

```
class Example {
    public static void main(String args[]) throws MalformedURLException {
        URL hp = new URL("https://www.ukr.net/news/main.html");
        System.out.println("Protocol: " + hp.getProtocol());
        System.out.println("Port: " + hp.getPort());
        System.out.println("Host: " + hp.getHost());
        System.out.println("File: " + hp.getFile());
        System.out.println("Ext:" + hp.toExternalForm());
    }
}
```

Клас URLConnection

Клас `URLConnection` є класом загального призначення і використовується для доступу до атрибутів віддаленого ресурсу. Як тільки буде встановлено з'єднання з віддаленим сервером, клас `URLConnection` можна використовувати для отримання властивостей віддаленого об'єкту перед тим, як переносити його локально. Ці атрибути розкриваються у специфікації мережевого протоколу HTTP і мають значення тільки для об'єктів типу `URL`, що використовують протокол HTTP.

Клас має один конструктор

```
protected URLConnection(URL url);
```

Який забезпечує створення URL-з'єднання за вказаною URL-адресою. З'єднання із об'єктом, на який посилається URL-адреса, не створюється.

Клас забезпечує наступні методи:

```
getLength() - повертає довжину вмісту у байтах;  
getLengthLong() - повертає довжину вмісту у байтах;  
getContentType() - повертає тип вмісту, це значення заголовку content-  
type;  
getDate() - повертає час та дату у мілісекундах з 1 січня 1970;  
getExpiration() - повертає час та дату терміну дії ресурсу у мілісекундах  
з 1 січня 1970;  
getHeaderField(int index) - повертає значення поля заголовку за  
вказаним індексом;  
getHeaderField(String name) - повертає значення поля заголовку за  
вказаним ім'ям;  
getHeaderFieldKey(int index) - повертає ключ поля заголовку за  
вказаним індексом;
```

Map<String , List<String>> getHeaderFields() - повертає відображення усіх полів;

getLastModified() - повертає час та дату модифікації ресурсу у мілісекундах з 01/01/1970;

InputStream getInputStream() - повертає потік введення, який зв'язано з ресурсом.

Нижче наведено приклад, що дозволяє використати потік для отримання вмісту ресурсу.

Приклад 2

```
try{
    int c;
    URL ukr = new URL("http://www.education.zp.ua");
    URLConnection ukrCon = ukr.openConnection();
    long d = ukrCon.getDate();
    if(d==0)
        System.out.println("No date information.");
    else
        System.out.println("Date: " + new Date(d));
    System.out.println("Content-Type: " + ukrCon.getContentType());
    d = ukrCon.getExpiration();
    if(d==0)
        System.out.println("No expiration information.");
    else
        System.out.println("Expires: " + new Date(d));
    d = ukrCon.getLastModified();
    if(d==0)
        System.out.println("No last-modified information.");
    else
        System.out.println("Last-Modified: " + new Date(d));
    long len = ukrCon.getContentLengthLong();
    if(len == -1)
        System.out.println("Content length unavailable.");
    else
        System.out.println("Content-Length: " + len);
    if(len != 0) {
        System.out.println("=== Content ===");
        InputStream input = ukrCon.getInputStream();
        while (((c = input.read()) != -1)) {
            System.out.print((char) c);
        }
        input.close();
    } else {
        System.out.println("No content available.");
    }
} catch (Exception e){
    System.out.println(e);
}
```

```
Map<String, List<String>> ukrHeader = ukrCon.getHeaderFields();
ukrHeader.forEach((aHead,ahList)->System.out.println("title
                                     head:"+aHead+" > "+ahList));
```

Клас `URLConnection`

Клас `URLConnection` є похідним від класу `URLConnection` і його призначено для підтримки з'єднання за мережевим протоколом прикладного рівня HTTP.

Для отримання об'єкту класу `URLConnection` необхідно викликати метод `openConnection()` у об'єкту класу `URL`, а результат привести до типу `URLConnection`. Після отримання посилання на об'єкт класу `URLConnection` можна використовувати методи, що наслідувались від класу `URLConnection`, а також будь-які методи класу `URLConnection`. Найчастіше використовуються наступні методи цього класу:

`static boolean getFollowRedirects()` - повертає логічне значення `true`, якщо автоматично виконано переадресацію, інакше — `false`;

`String getRequestMethod()` - повертає рядкове представлення методу, за допомогою якого виконується запит за URL (за замовчуванням це метод `GET`);

`int getResponseCode() throws IOException` - повертає код відповіді по протоколу HTTP; якщо код відповіді не може бути отриманим, то повертає 1 (при роз'єднанні генерується виключення `IOException`);

`String getResponseMessage() throws IOException` - повертає повідомлення, яке зв'язане з кодом відповіді; якщо повідомлення відповіді відсутнє, то повертається значення `null`;

`static void setFollowRedirects(boolean set)` - якщо параметр `set` отримує логічне значення `true`, то переадресація виконується автоматично, якщо отримує логічне значення `false`, то переадресації не відбувається (за замовчуванням переадресація виконується автоматично);

`void setRequestMethod (String method) throws ProtocolException` - задає метод для запитів по протоколу HTTP у відповідності до значення параметра `method` - за замовчуванням це метод `GET`.

Приклад 3

```
public static void main(String args[]) throws Exception {
    URL url = new URL("http://www.google.com");
    HttpURLConnection urlCon = (HttpURLConnection) url.openConnection();
    System.out.println("Request method is " + urlCon.getRequestMethod());
    System.out.println("Response code is " + urlCon.getResponseCode());
    System.out.println("Response Message is " + urlCon.getResponseMessage());
    Map<String, List<String>> urlMap = urlCon.getHeaderFields();
    Set<String> urlField = urlMap.keySet();
    System.out.println("\nHere is the header:");
```

```
for(String k : urlField) {  
    System.out.println("Key: " + k + " Value: " + urlMap.get(k));  
}  
}
```

Завдання самостійної роботи

1. Виконайте приклад 1. Наведіть скрин-шот та надайте пояснення отриманому результату.
2. Випробуйте конструктори та методи класу URL, що наведено у теоретичній частині. Наведіть програмний код, скрин-шоти та результати випробувань. Надайте пояснення отриманим результатам.
3. Реалізуйте приклад 2. Наведіть скрин-шот та надайте пояснення отриманому результату.
4. Реалізуйте приклад 3. Наведіть скрин-шот та надайте пояснення отриманому результату.
5. Підготуйте та надайте звіт.