Networking (1)

Podstawowe pojęcia dotyczące sieci

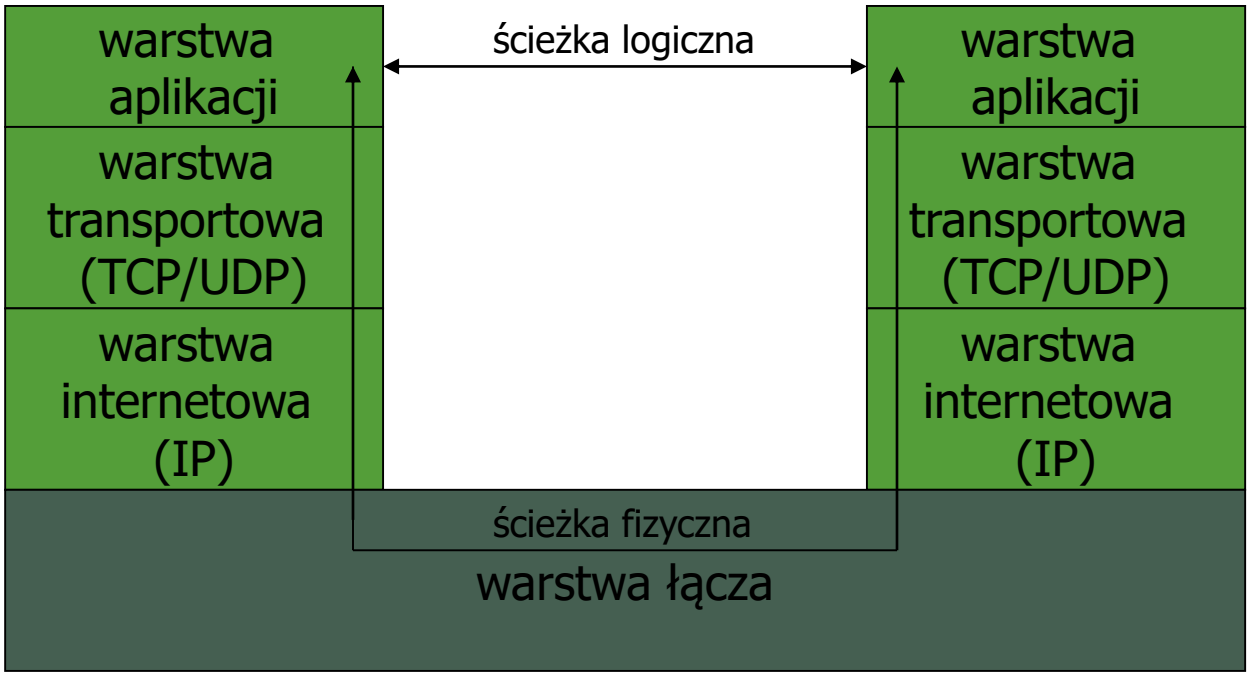
- *Sieć* to zbiór komputerów i innych urządzeń, które mogą się ze sobą komunikować w czasie rzeczywistym za pomocą transmisji danych. Urządzenia w sieci są ze sobą połączone (kablami, światłowodami, urządzeniami bezprzewodowymi).
- Każda maszyna (komputery, rutery, drukarki, terminale, itp) znajdująca się w sieci nazywa się *węzłem*. Węzły, które są w pełni funkcjonalnymi komputerami nazywane są *hostami*.
- Każdy węzeł w sieci ma swój *adres*.

Podstawowe pojęcia dotyczące sieci

- Wszystkie współczesne sieci komputerowe są sieciami *komutacji pakietów*.
- Każdy pakiet oprócz fragmentu danych zawiera informację o tym kto i dokąd go wysłał.
- Zestaw reguł według których komputery i urządzenia komunikują się ze sobą nazywa się *protokołem*.

Warstwy sieci

- Przesyłanie danych przez sieć to skomplikowana operacja:



Warstwy sieci

- *Warstwa łącza* definiuje konkretny interfejs sieciowy (karta ethernetowa czy łącze PPP) i przesyła datagramy IP fizycznym łączem (do sieci lokalnej i w świat) – Java nie ma dostępu do tej warstwy.
- *Warstwa internetowa* odpowiada za grupowanie danych w pakiety oraz za schemat adresowania, w którym różne maszyny mogą się odnaleźć – Java zna tylko protokół IP dla tej warstwy (jest on najpowszechniej stosowany).

Warstwy sieci

- *Warstwa transportowa* odpowiada za to, aby pakiety były odbierane w tej samej kolejności w jakiej zostały wysłane, oraz aby żaden z nich nie został uszkodzony ani zagubiony – Java umie obsłużyć dwa protokoły tej warstwy:
  - TCP (ang. *Transmission Control Protocol*) niezawodny,
  - UDP (ang. *User Datagram Protocol*) zawodny ale szybki.
- *Warstwa aplikacji* dostarcza dane użutkownikowi – znane protokoły tej warstwy to HTTP, SMTP, POP, IMAP, FTP, NFS, NNTP oraz wiele innych.

Adresy IP

- Protokół Internetowy IP jest niezależny od platformy, automatycznie wyznacza trasę rutingu.
- Każdy komputer w sieci IP jest identyfikowany za pomocą swojego unikatowego 32-bitowego (IPv4) albo 128-bitowego (IPv6) adresu.
- DNS (ang. *Domain Name System*) to usługa, która tłumaczy nazwy mnemoniczne adresów na nazwy liczbowe.
- Pakiety, które przychodzą do określonego hosta mogą trafiać do różnych aplikacji czy serwisów dzięki *portom*. Jest ich 65535 dla protokołów TCP i UDP (porty o numerach 1-1023 są zarezerwowane dla usług standardowych).

Adresy IP

- W pakiecie `java.net` jest zdefiniowana klasa `InetAddress`, która reprezentuje adres IP.
- Klasa `InetAddress` ma dwie podklasy `Inet4Address` i `Inet6Address` reprezentujące odpowiednio adresy protokułu internetowego w standardach IPv4 i IPv6.
- Klasa ta jest wykorzystywana przez inne klasy sieciowe: `URL`, `Socket`, `ServerSocket`, `DatagramSocket`, `DatagramPacket`.

Adresy IP

- Klasa `InetAddress` pozwala tworzyć obiekty tych klas za pomocą metod statycznych:
```
getByAddress (byte[] addr)
getByName (String host)
getLocalHost ()
```
- Z obiektu `InetAddress` można wydobyć szczegółowe informacje o adresie IP za pomocą metod:
```
getHostAddress ()
getHostName ()
getCanonicalHostName ()
toString ()
isAnyLocalAddress ()
isReachable (int timeout)
```

Przykład

```
BufferedReader stdin = new BufferedReader(
    new InputStreamReader(System.in));
System.err.print("Nazwa hosta: ");
String host = stdin.readLine().trim();
try {
    InetAddress address = InetAddress.getByName(host);
    System.out.println(address);
}
catch (UnknownHostException ex) {
    System.out.println(
        "Nie można zlokalizować hosta " + host + "!");
}
```

Adres URL

- URL (ang. *Uniform Resource Locator*) to referencja do zasobu w Internecie.
- URL składa się z nazwy protokołu i nazwy zasobu, na przykład: http://www.oracle.com:80/index.html#ref?x=4&y=8
- Nazwa zasobu może składać się z nazwy hosta, portu, ścieżki, pliku, referencji i zapytania.

Klasa `URL`

- Klasa `URL` reprezentuje adres URL w sieci WWW.
- Obiekt URL można utworzyć na kilka sposobów:

```
new URL (String spec)
new URL (String prot, String host,
  String file)
new URL (String prot, String host,
  int port, String file)
new URL (URL context, String spec)
```

- Podczas tworzenia obiektu `URL` może zostać zgłoszony wyjątek `MalformedULRException`.

Klasa `URL`

- Klasa `URL` udostępnia wiele metod odczytywania parametrów adresu URL:

```
getProtocol ()
getHost ()
getPort ()
getPath ()
getQuery ()
getRef ()
```

- W klasie `URL` istnieje metoda, która potrafi nawiązać połączenie z podanym zasobem w sieci i otworzyć dla niego strumień do czytania:

```
InputStream openStream ()
```

Przykład

```
URL url = new URL(URLName);
BufferedReader in = new BufferedReader(
    new InputStreamReader(url.openStream()));
String line;
while ((line=in.readLine()) != null) {
    System.out.println(line);
}
in.close();
```

Klasa `URLConnection`

- Klasa `URLConnection` ma zapewnić łatwiejszą w użyciu, wysokopoziomową abstrakcję połączenia sieciowego.
- Klasa `URLConnection` wykorzystuje klasę `Socket` do zapewnienia łączności sieciowej.
- Klasa `URLConnection` jest mocno związana z protokołem HTTP i zakłada, że każdy przesyłany plik jest poprzedzony nagłówkiem MIME.

Klasa URLConnection

- Otwieranie połączeń URLConnection:

```
// konstrukcja URL
URL url = new URL("http://…");
// pozyskanie URLConnection
URLConnection uc = url.openConnection();
// konfiguracja URLConnection …
// odczytanie pól nagłówka …
// pobranie strumienia wejściowego …
// pobranie strumienia wyjściowego …
// zamknięcie połączenia …
```

Klasa `URLConnection` – parametry połączenia

- Serwery HTTP dostarczają sporo informacji w nagłówkach MIME.
- Klasa `URLConnection` posiada kilka metod do odczytywania najważniejszych informacji z nagłówka MIME:
  `getContentType()`
  `getContentLength()`
  `getContentEncoding()`
  `getDate()`
  `getExpiration()`
  `getLastModified()`
- Klasa `URLConnection` posiada też kilka ogólnych metod do odczytywania informacji z nagłówka MIME:
  `getHeaderFieldKey(int)`
  `getHeaderField(int)`
  `getHeaderField(String)`

Klasa `URLConnection` – konfiguracja połączenia

- Klasa `URLConnection` może konfigurować połączenie za pomocą metod:
  `setDoInput(boolean)`
  `setDoOutput(boolean)`
  `setAllowUserInteraction(boolean)`
  `setUseCaches(boolean)`
  `setIfModifiedSince(long)`
- Klasa `URLConnection` może pobrać treść (obiekt `Object`) metodą `getContent()`.
- Klasa `HttpURLConnection` jest podklasą `URLConnection` zawiera pewne dodatkowe metody przydatne do pracy z adresami URL typu http.

Klasa `URLConnection` – czytanie i pisnie

- Czytanie danych:

```
URL url = new URL("http://…");
URLConnection uc
  = url.openConnection();
InputStream is = uc.getInputStream();
```

- Pisanie danych:

```
URL url = new URL("http://…");
URLConnection uc
  = url.openConnection();
uc.setDoOutput(true);
InputStream is = uc.getInputStream();
OutputStream os = uc.getOutputStream();
```

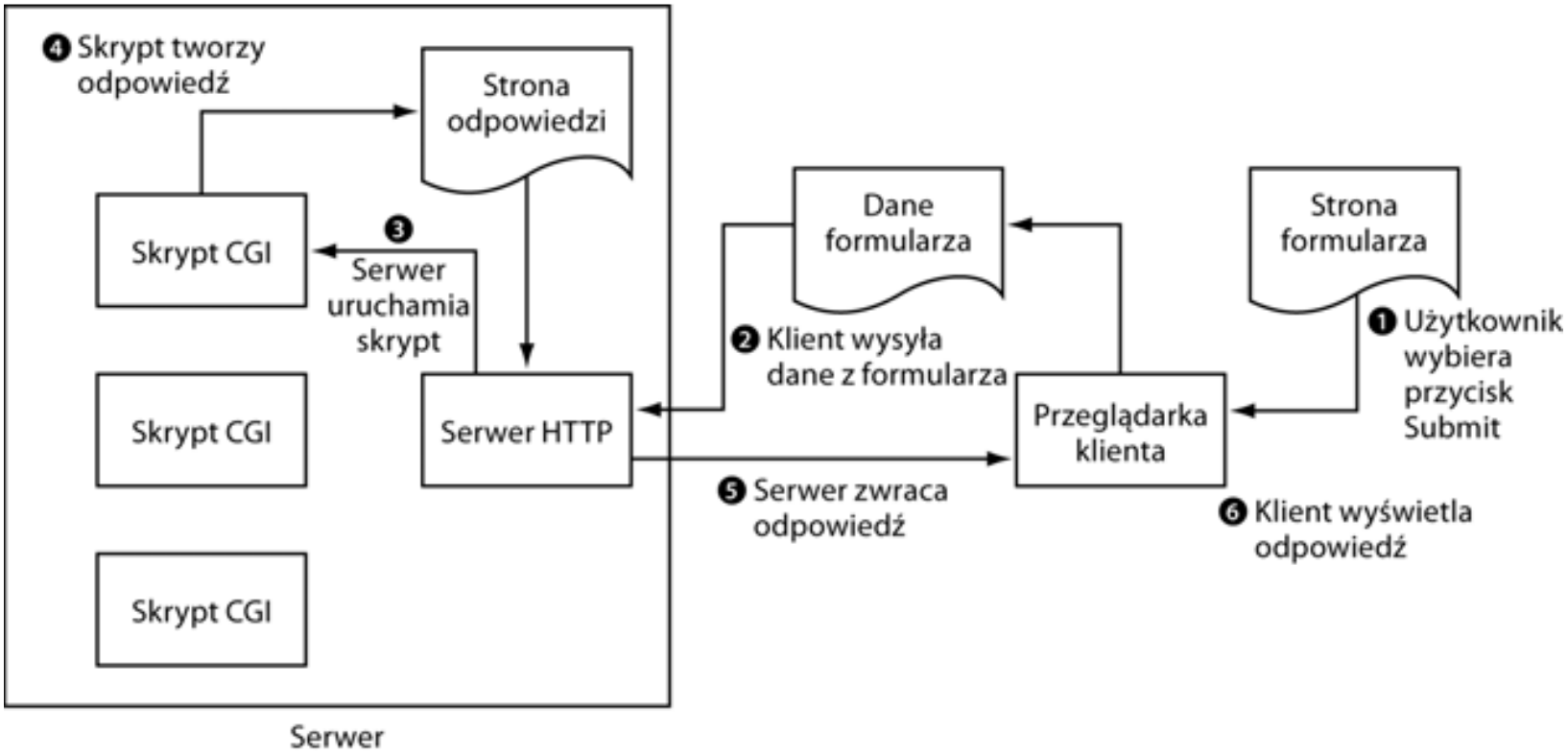
Klasa `URLConnection` – czytanie

- Z obiektu URLConnection można pobrać strumień wejściowy – połączenie jest otwierane niejawnie przez wywołanie getInputStream().
- Przykład:

```
URL url = new URL("http://.../...");
URLConnection conn = url.openConnection();
BufferedReader in = new BufferedReader(
    new InputStreamReader(
        conn.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null)
    System.out.println(inputLine);
in.close();
```

Klasa `URLConnection` – komunikacja z serwerem

- Sparametryzowana komunikacja z serwerem (serwer generuje odpowiedź w oparciu o przesłane dane):



Klasa `URLConnection` – komunikacja z serwerem

- Istnieje wiele technologii umożliwiających serwerom Web wykonywanie programów/skryptów – do najbardziej znanych należą servlety Javy, JavaServer Faces czy skrypty CGI.
- Serwer WWW odbiera dane (na przykład z formularza) uruchamia skrypt do przetworzenia tych danych a następnie tworzy nową stronę w języku HTML, którą odsyła z powrotem do przeglądarki.
- Dane przesyłane są w standardowym formacie do serwera, który następnie zajmuje się przekazaniem ich skryptowi, który przygotuje odpowiedź.

Klasa `URLConnection` – komunikacja z serwerem

- Istnieją dwa sposoby wysyłania informacji do serwera: `GET` oraz `POST`.
- Metoda `GET` polega na dołączeniu parametrów na końcu łańcucha URL:
  http://host/skrypt?parametry
- Każdy z parametrów w URL ma postać:
  nazwa=wartość
- Parametry są oddzielone od siebie znakiem &.
- Wartości parametrów muszą być zakodowane zgodnie ze schematem kodowania URL.

Klasa `URLConnection` – komunikacja z serwerem

- Metoda `POST` nie dołącza parametrów do adresu URL.
- Wykorzystuje ona strumień wyjściowy `URLConnection`, do którego zapisuje pary nazwa-wartość.
- Pary te muszą być poddane kodowaniu URL i rozdzielone znakiem `&`.
- Aby wysłać dane przez strumień, musimy po kolei:
- URLConnection.

Klasa `URLConnection` – komunikacja z serwerem

- Aby wysłać dane przez strumień, musimy po kolei:
  - tworzymy obiekt `URLConnection`:
    ```
    URL url = new URL("http://.../...");
    URLConnection conn = url.openConnection();
    ```
  - Następnie wywołujemy metodę `setDoOutput()`:
    ```
    conn.setDoOutput(true);
    ```
  - Tworzymy strumień do wysyłania danych:
    ```
    PrintWriter out =
      new PrintWriter(conn.getOutputStream());
    ```

Klasa `URLConnection` – komunikacja z serwerem

- Aby wysłać dane przez strumień, musimy po kolei (c.d.):
  - Wysłamy dane do serwera:

```
out.print(name1 + "=„
  + URLEncoder.encode(value1, "UTF-8") + "&");
out.print(name2 + "=„
  + URLEncoder.encode(value2, "UTF-8"));
```

  - Zamykamy strumień:

```
out.close();
```

  - Na końcu wywołujemy metodę `getInputStream()` i odczytujemy odpowiedź serwera korzystając ze strumienia wejściowego.

Literatura

- E.R.Harold: *Java. Programowanie sieciowe.* Wydawnictwo RM, Warszawa 2001.
- C.S.Horstmann, G.Cornell: *Java – techniki zaawansowane. Wydanie 9. Rozdział 3: Programowanie aplikacji sieciowych.* Wydawnictwo HELION, Gliwice 2013.
- Custom Networking (Java Tutorial): https://docs.oracle.com/javase/tutorial/networking/