# JAVA I SIECI

## ZAGADNIENIA:

- URL,
- Interfejs gniazd,
- transmisja SSL,
- protokół JNLP.

MATERIAŁY:

http://docs.oracle.com/javase/tutorial/networking/index.html

# URL

URL – Unified Resource Locator jest podstawową klasą identyfikującą zasoby w internecie:

```
import java.net.*;
import java.io.*;

public class URLExample {
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://www.google.pl/");

        BufferedReader in = new BufferedReader(new InputStreamReader(
                                                    url.openStream()));

        String s;
        while ((s = in.readLine()) != null)
            System.out.println(s);
        in.close();
    }
}
```

# URLConnection

URLConnection zawiera metody umożliwiające nawiązanie połączenia z zasobem reprezentowanym przez URL.

```
import java.net.*;
import java.io.*;

public class URLConnectionExample {
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://www.google.pl/");
        URLConnection con = url.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(
                                con.getInputStream()));

        String s;
        while ((s = in.readLine()) != null)
            System.out.println(s);
        in.close();
    }
}
```

# URLConnection

URLConnection umożliwia także zapis do wskazanego zasobu przez obiekt URL.

```
import java.io.*;
import java.net.*;
public class URLConnectionWriter {
    public static void main(String[] args) throws Exception {
        URL url = new URL(args[0]);
        URLConnection con = url.openConnection();
        con.setDoOutput(true);

        OutputStreamWriter out = new OutputStreamWriter(
                                                con.getOutputStream());

        for (int i = 1; i < args.length; i++)
            out.write(args[i]);
        out.close();
    }
}
```

# URLConnection

Zarówno w przypadku URL jak I URLConnection komunikacja odbywa się z wykorzystaniem odpowiedniego protokołu. Dokumentacja Javy gwarantuje standardowo obsługę następujących protokołów:

- http,
- https,
- ftp, ftp://login:haslo@serwer:port/katalog/podkatalog/plik
- file,
- jar.

Obsługa innych protokołów wymaga implementacji klasy URLStreamHandler.

# INTERFEJS GNIAZD

Standardowa obsługa sieci w Javie opiera się o tzw. interfejs gniazd. Najważniejsze klasy, umożliwiające komunikację poprzez sieć to:

- Socket – klasa reprezentująca gniazdo służące do nawiązywania połączenia, wysyłania I odbierania danych,
- ServerSocket – klasa reprezentująca gniazdo oczekujące na przychodzące żądania połączeń.

Ponadto istnieją także gniazda SSLSocket I SSLServerSocket obsługujące komunikację szyfrowaną protokołem SSL/TLS.

# PROGRAM KLIENCKI

```
import java.io.*;
import java.net.Socket;

public class ClientExample {

    public static Socket sock;

    public static void main(String[] args) throws IOException{
        // tworzymy gniazdo I nawiazujemy polaczenie z komputerem
        // identyfikowanym przez adres args[0] na porcie args[1]
        sock = new Socket(args[0], Integer.valueOf(args[1]));

        // pobieramy strumienie zwiazane z gniazdem
        OutputStream os = sock.getOutputStream();
        InputStream  is = sock.getInputStream();

        // tworzymy Reader na standardowym wejsciu (klawiaturze)
        BufferedReader br = new BufferedReader(new InputStreamReader(
                                                        System.in));
```

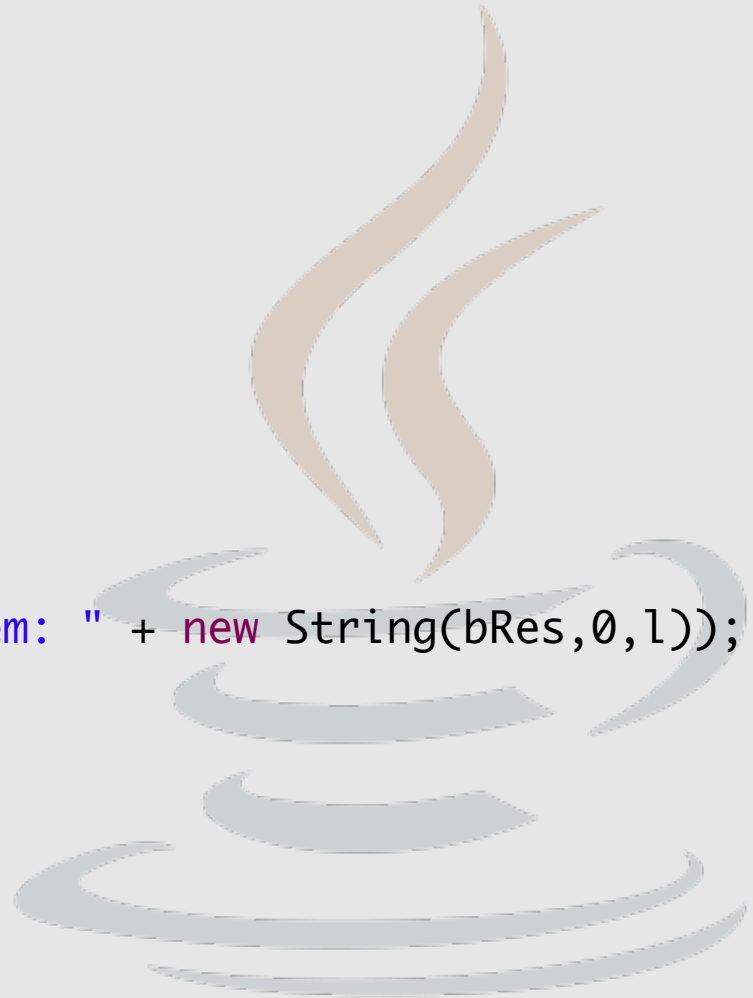
# PROGRAM KLIENCKI

```
        // zmienne pomocnicze
        String sLine;
        byte[] bRes = new byte[100];

        // glowna petla programu, pobieramy dane z klawiatury
        while((sLine=br.readLine())!=null){
            // wysylamy je przez gniazdo
            os.write(sLine.getBytes());
            System.out.println("wyslalem: " + sLine);
            // odbieramy odpowiedz z serwera - to jest zle rozwiazanie
            // dobre rozwiązanie - odbieranie danych w osobnym wątku
            is.read(bRes);
            System.out.println("odebralem" + new String(bRes));
        }
        // zamykamy strumien i gniazdo
        br.close();
        sock.close();
    }
} // koniec programu
```

# PROGRAM KLIENCKI

Osobny wątek do odbioru danych:

```
Thread t = new Thread(new Runnable(){
    public void run(){
        byte[] bRes = new byte[100];
        InputStream is;
        int l;
        try {
            is = sock.getInputStream();
            while(true){
                l = is.read(bRes);
                System.out.println("odebralem: " + new String(bRes,0,l));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
t.start();
```

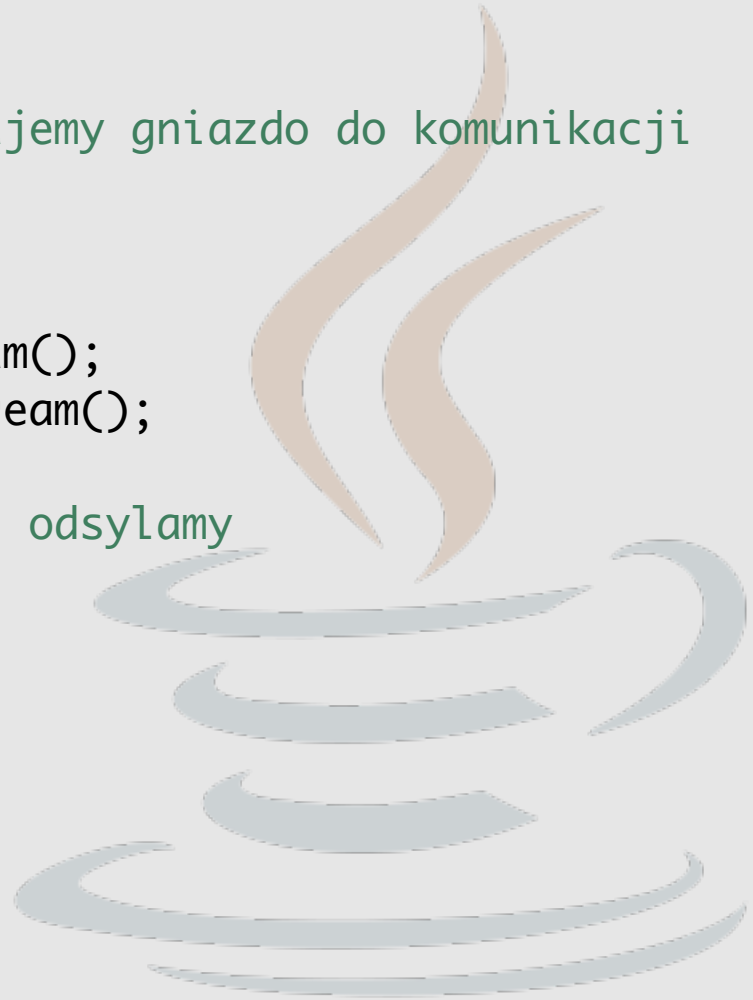


# SERWER ECHO

```
import java.io.*;
import java.net.*;

public class ServerExample {

    // gniazdo oczekujace na polaczenia
    private static ServerSocket ss;

    public static void main(String[] args) throws IOException{

        // tworzymy gniazdo, ktore oczekuje na przychodzace polaczenia
        // na porcie przekazanym jako argument wywolania programu
        ss = new ServerSocket(Integer.valueOf(args[0]));
```

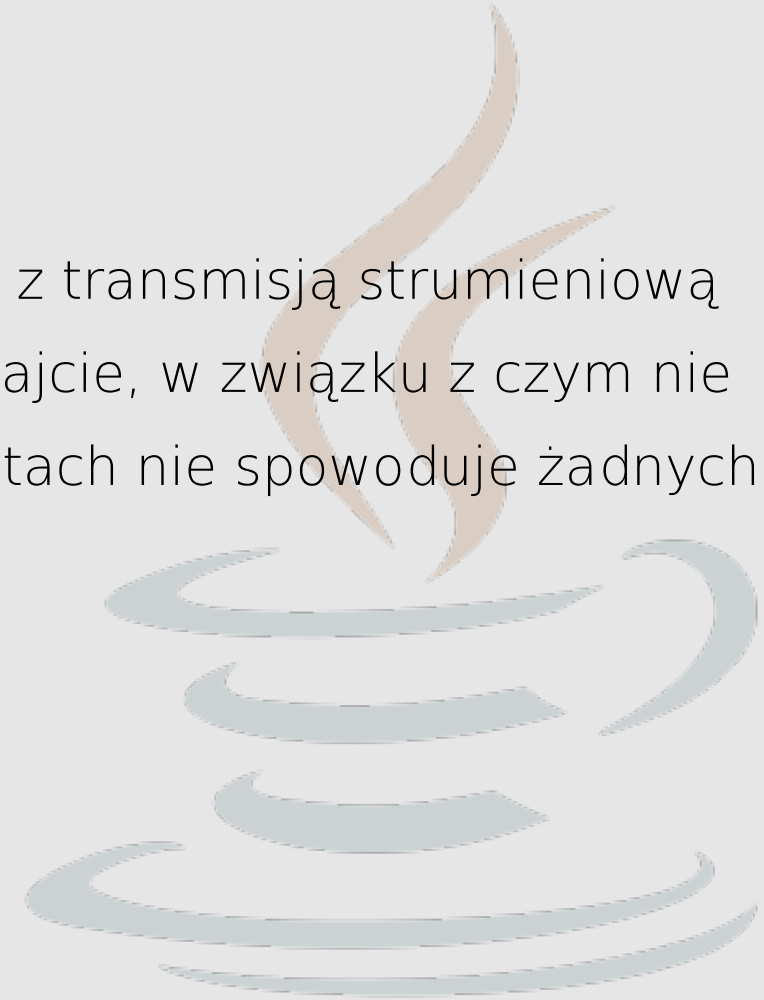
# SERWER ECHO

```
        // nieskonczona petla
        while(true){
            // akceptujemy polaczenie, dostajemy gniazdo do komunikacji
            // z klientem
            Socket s = ss.accept();
            // strumienie
            InputStream is = s.getInputStream();
            OutputStream os = s.getOutputStream();
            int b;
            // czytamy, piszemy na konsoli i odsylamy
            while((b=is.read())!=-1){
                System.out.print((char)b);
                os.write(b);
            }
            s.close();
        }
    }
}
```



# SERWER ECHO

W serwerze nie ma problemu związanego z transmisją strumieniową ponieważ dane są przetwarzane bajt po bajcie, w związku z czym nie sytuacja, gdy dane dotrą w różnych pakietach nie spowoduje żadnych efektów ubocznych.



# SSL W JAVIE

Protokół SSL umożliwia bezpieczną (szyfrowaną) transmisję danych poprzez niezabezpieczoną sieć. Dodatkowo SSL umożliwia autoryzację stron komunikacji. W tym celu wykorzystywany jest mechanizm certyfikatów. Za transmisję z użyciem protokołu SSL odpowiedzialne są klasy zgrupowane w pakiecie javax.net.SSL.

Implementacja SSH jest dostępna poprzez zewnętrzne biblioteki. Jedną z nich jest jsch (http://www.jcraft.com/jsch/).

# SERWER SSL

```
import javax.net.ssl.*;
import java.io.*;

public class EchoServer {
    public static void main(String[] args) throws IOException {
        SSLServerSocketFactory factory = (SSLServerSocketFactory)
        SSLServerSocketFactory.getDefault();
        SSLServerSocket ss = (SSLServerSocket) factory
                                              .createServerSocket(9999);
        SSLSocket s = (SSLSocket) ss.accept();
        BufferedReader br = new BufferedReader(new InputStreamReader(
                                               s.getInputStream()));
        String sTmp = null;
        while ((sTmp = br.readLine()) != null) {
            System.out.println(sTmp);
            System.out.flush();
        }
    }
}
```

# KLIENT SSL

```
import javax.net.ssl.*;
import java.io.*;
public class EchoClient {
    public static void main(String[] args) throws Exception {
        SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory
                                                          .getDefault();
        SSLSocket s = (SSLSocket) factory.createSocket("localhost", 9999);
        BufferedReader br = new BufferedReader(
                                        new InputStreamReader(System.in));
        OutputStreamWriter osw = new OutputStreamWriter(
                                                s.getOutputStream());
        BufferedWriter bw = new BufferedWriter(osw);
        String sTmp = null;
        while ((sTmp = br.readLine()) != null) {
            bw.write(sTmp + '\n');
            bw.flush();
        }
    }
}
```

# SSL W JAVIE

Pierwsza czynność to wygenerowanie klucza:

```
keytool -genkey -keystore mySrvKeystore -keyalg RSA
```

Uruchomienie serwera:

```
java -Djavax.net.ssl.keyStore=mySrvKeystore
-Djavax.net.ssl.keyStorePassword=123456 EchoServer
```

Uruchomienie klienta:

```
java -Djavax.net.ssl.trustStore=mySrvKeystore
-Djavax.net.ssl.trustStorePassword=123456 EchoClient
```

Dodatkowe parametry wywołania pozwolą zobaczyć informacje związane z połączeniem SSL:

```
-Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
-Djavax.net.debug=ssl
```

Przykład ze strony: http://stilius.net/java/java_ssl.php.

# SSL W JAVIE

Domyślnie tylko jedna strona komunikacji (serwer) musi potwierdzać swoją tożsamość. Aby wymusić autoryzację klienta należy użyć metod: `setNeedClientAuth(true)` lub `setWantClientAuth(true)` wywołanych na rzecz obiektu `SSLServerSocket`.

Jeśli chcemy aby żadna ze stron nie musiała potwierdzać swojej tożsamości musimy zmienić domyślne algorytmy kodowania. Najłatwiej zrobić to tworząc własne rozszerzenie klasy `SSLSocketFactory`.

Listę obsługiwanych i domyślnych algorytmów uzyskamy za pomocą metod: `getSuppotredCipherSuites()` oraz `getDefaultCipherSuites()`.

# JAVA WEB START

Technologia Java Web Start jest stosowana do lokalnego uruchamiania programów w Javie umieszczonych w sieci.

JWS:

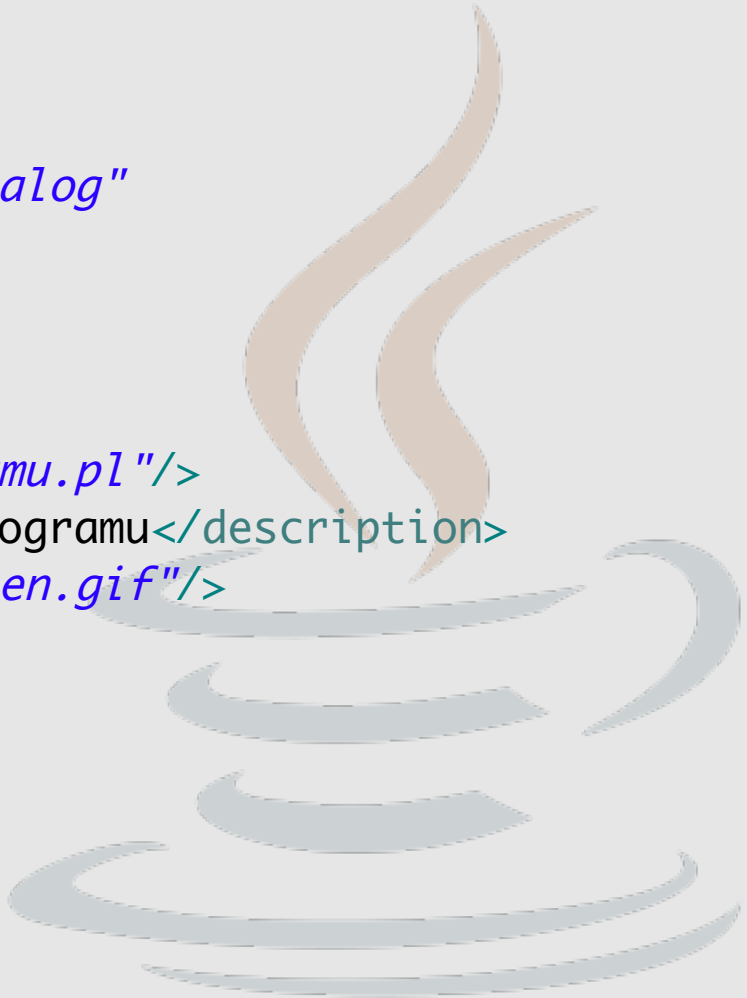
- jest w pełni niezależna od używanych przeglądarek internetowych,
- umożliwia automatyczne pobranie właściwej wersji środowiska JRE,
- pobierane są tylko pliki, które zostały zmienione,
- obsługuje prawa dostępu do zasobów lokalnego komputera (dysk, sieć, itp.),
- do opisu zadania do uruchomienia wykorzystuje pliki jnlp (Java Network Launch Protocol).

Więcej informacji:

http://docs.oracle.com/javase/tutorial/deployment/webstart/index.html

# PROTOKÓŁ JNLP

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"
  codebase="http://www.serwer.w.sieci.pl/katalog"
  href="plik_jws.jnlp">
  <information>
    <title>Nazwa programu</title>
    <vendor>Producent programu</vendor>
    <homepage href="http://www.strona.programu.pl"/>
    <description kind="short">Krotki opis programu</description>
    <icon kind="splash" href="kat/splashscreen.gif"/>
    <icon href="kat/ikona.gif"/>
    <offline-allowed/>
  </information>
  <security>
      <all-permissions/>
  </security>
```



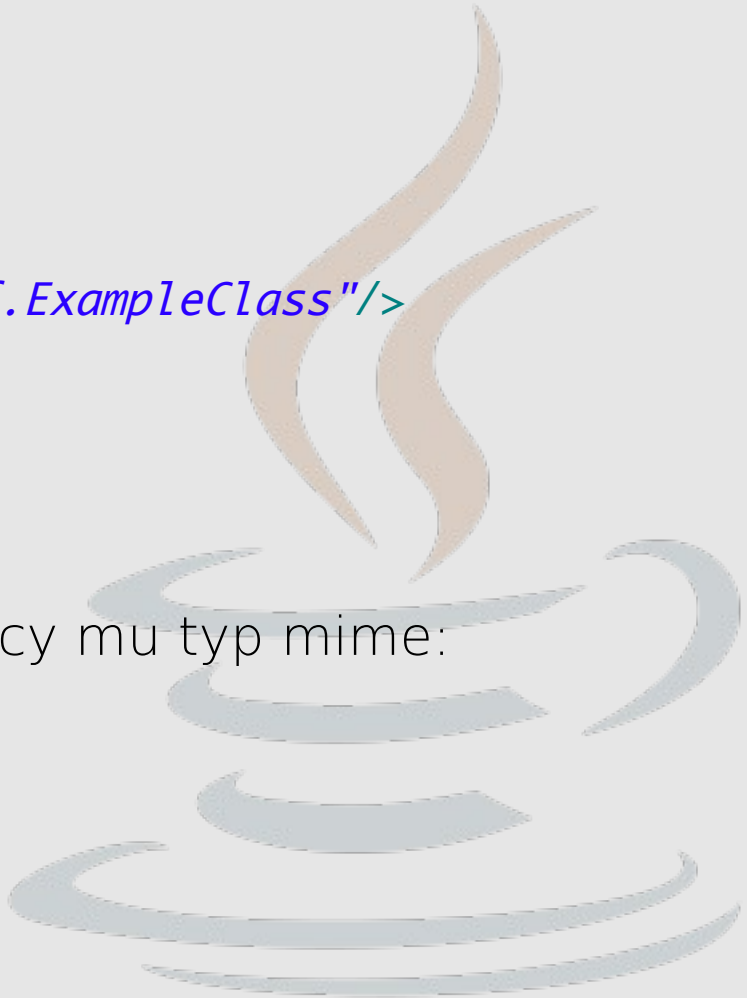
# PROTOKÓŁ JNLP

```
  <resources>
    <j2se version="1.4+"/>
    <jar href="kat/archiwum1.jar"/>
    <jar href=" kat2/lib/biblioteka.jar"/>
  </resources>
  <application-desc main-class="pl.edu.uj.if.ExampleClass"/>
</jnlp>
```

Plik jnlp umieszczamy na serwerze www.

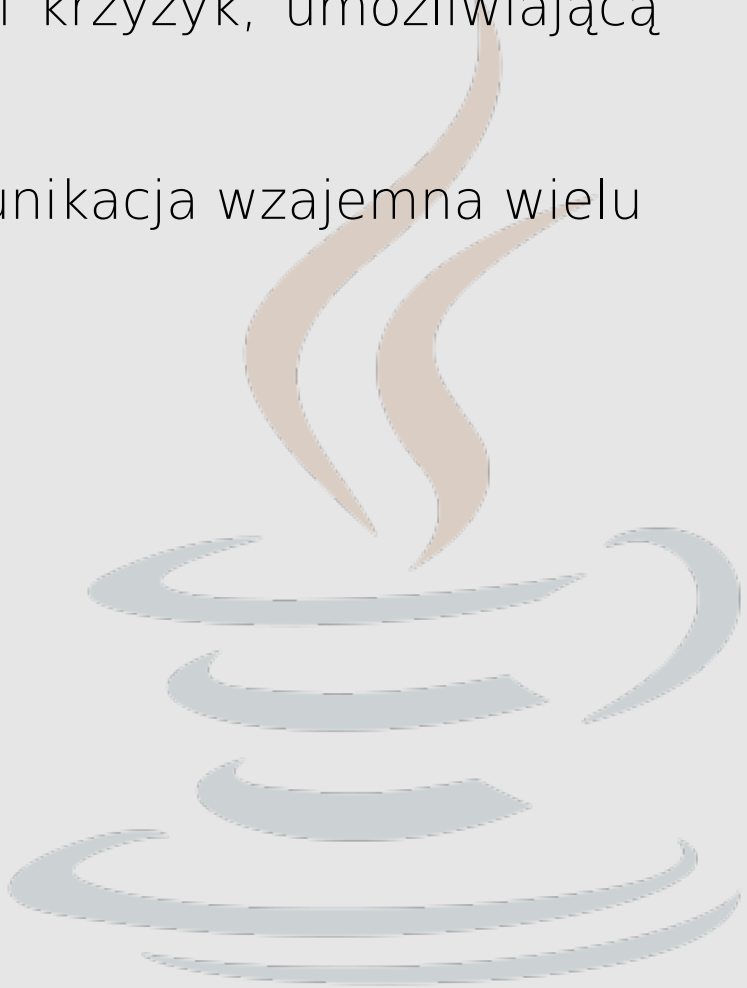
Często należy skonfigurować odpowiadający mu typ mime:

application/x-java-jnlp-file JNLP



# ĆWICZENIA

- Proszę napisać sieciową grę w kółko i krzyżyk, umożliwiającą rozgrywkę sieciową dla dwóch osób.
- Proszę napisać czat internetowy (komunikacja wzajemna wielu osób).



# DZIĘKUJĘ ZA UWAGĘ