Helder da Rocha
www.argonavis.com.br

Pacote java.net

- *O pacote java.net contém classes para implementar comunicação através da rede*
- *Fáceis de usar. Semelhante à criação de arquivos:*

```
Socket sock = new Socket("www.x.com", 80);
PrintWriter os = new PrintWriter(
   new OutputStreamWriter(
      sock.getOutputStream()));
BufferedReader is = new InputStreamReader(
    new BufferedReader(
      sock.getInputStream());
os.println("GET / HTTP/1.0\n\n");
String linha = "";
while ((linha = is.readLine()) != null) {
   System.out.println(linha);
}
```

Principais classes

- Toda a comunicação em rede é fundamentada no esquema máquina/porta: **um soquete**.
  - Na API Java, os soquetes são representados por várias classes, dependendo do seu tipo.
- A comunicação via protocolo **TCP** (Transfer Control Protocol), confiável, suportada pelas classes
  - Socket (soquete de dados)
  - ServerSocket (soquete do servidor).
- A comunicação via **UDP** (Unreliable Datagram Protocol), não-confiável, suportada pelas classes
  - DatagramSocket (soquete de dados UDP),
  - DatagramPacket (pacote UDP)
  - MulticastSocket (soquete UDP para difusão).

URL e URLConnection

- A **URL** (Uniform Resource Locator) java.net.URL.
  - getStream(): para obter um fluxo de dados a partir da URL ou seu método
  - getContent(): para ler um objeto contendo os dados. Se o tipo de dados for suportado, será possível manipulá-lo. Se não for, pode usar as classes ContentHandler e ContentHandlerFactory
- A classe **URLConnection** permite que a informação de cabeçalho da conexão seja recuperada.
  - comprimento dos dados,
  - tipo MIME (Multipart Internet Mail Extensions)
  - cookies
  - ...

Socket

- *Representa um dos lados de uma conexão bidirecional TCP*
- *Para usar um socket é preciso obter seus fluxos de entrada e saída*

```
InetAddress end =
      InetAddress.getByName("info.acme.com");
Socket con = new Socket(end, 80);
InputStream dados = con.getInputStream();
OutputStream comandos =
                        con.getOutputStream();
```

- *Depois basta ler ou enviar dados como qualquer outro stream*

Socket (2)

- *Para ler ou gravar caracteres ao invés de bytes, pode-se usar as classes Reader e Writer:*

```
Socket con = new
        Socket("maquina.com.br", 4444);

Reader = new InputStreamReader(
                    con.getInputStream());

Writer = new OutputStreamWriter(
                    con.getOutputStream());
```

ServerSocket

- *Soquete de servidor*
  - *Fica escutando uma porta a espera de um sinal do cliente*
  - *Quando receber uma conexão deve criar um novo thread copiar o socket para ele*
- *Exemplo*

```
ServerSocket escuta = new ServerSocket(80);
while(true) {
   Socket cliente = escuta.accept();  // espera
   InputStream comandos =
                        cliente.getInputStream();
   OutputStream dados =
                        cliente.getOutputStream();
}
```

Datagramas UDP: envio

```
DatagramSocket udp = new DatagramSocket();

byte[] info = {'o', 'i'};

int porta = 3333;

InetAddress destino =
          InetAddress.getByName("longe.com");

DatagramPacket pacote =
               new DatagramPacket( info,
                                   info.length,
                                   destino,
                                   porta);

udp.send(pacote);
```

Datagramas UDP: recebimento

```
byte info = new byte[256];

DatagramSocket udp = new DatagramSocket(3333);

DatagramPacket pacote =
        new DatagramPacket(info, info.length);

udp.receive(pacote);

InetAddress remetente = udp.getAddress();

int portaRemetente = udp.getPort();

byte[] dados = udp.getData();
```

Exceções

- *Várias exceções podem ser provocadas em uma ambiente de rede.*
- *O compilador irá informar as exceções que precisam ser declaradas ou tratadas*
- *As mais comuns são*
  - *SocketException*
  - *MalformedURLException*
  - *UnknownHostException*
  - *ProtocolException*

Exercícios

- *1. Escreva um programa que descubra o número IP da sua máquina*
- *2. Escreva um programa que*
  - *Conecte-se na porta HTTP (geralmente 80) de um servidor conhecido*
  - *Envie o comando: "GET / HTTP/1.0\n\n"*
  - *Grave o resultado em um arquivo resultado.html*
  - *(você pode usar URLConnection ou Sockets)*

Exercícios (2)

- *3) Escreva um programa que use um ServerSocket para aguardar conexões de um cliente. O programa deverá ter duas partes:*
  - *(1) uma classe principal (Servidor) que fica escutando a porta escolhida (número acima de 1024) e*
  - *(2) uma classe que estende Thread (Conexao) e que contém irá tratar as requisições do cliente.*

  O servidor deverá imprimir na tela todos os comandos enviados por todos os clientes.
- *4) Crie um cliente para a aplicação acima e teste-o em várias máquinas diferentes.*