Helder da Rocha
www.argonavis.com.br

Sobre este módulo

- Este módulo pretende apenas cobrir conceitos essenciais sobre programação em rede com Java
  - Como criar um servidor e um cliente TCP/IP
  - Como abrir uma conexão TCP/IP
  - Como ler de uma conexão
  - Como escrever para uma conexão
- Classes abordadas do pacote java.net
  - Socket e ServerSocket
  - InetAddress
  - URL
- Para maiores detalhes, consulte as referências no final do capítulo

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);                    (1)
PrintWriter os = new PrintWriter(
   new OutputStreamWriter(
      sock.getOutputStream()));
BufferedReader is = new BufferedReader(
   new InputStreamReader(
      sock.getInputStream());
os.println("GET / HTTP/1.0\n\n");                             (2)
String linha = "";
while ((linha = is.readLine()) != null) {
   System.out.println(linha);                                 (3)
} // ... feche o socket com sock.close();
```

(1) Abre socket para servidor Web, (2) envia comando e (3) imprime resposta

TCP/IP no pacote java.net

- A comunicação via protocolo **TCP** (Transfer Control Protocol), confiável, é suportada pelas classes
  - Socket (soquete de dados)
  - ServerSocket (soquete do servidor).
- A comunicação via **UDP** (Unreliable Datagram Protocol), não-confiável, é suportada pelas classes
  - DatagramSocket (soquete de dados UDP),
  - DatagramPacket (pacote UDP)
  - MulticastSocket (soquete UDP para difusão).
- Endereçamento
  - InetAddress (representa um endereço na Internet)
  - URL (representa uma URL)

URL

- *Representa uma **URL***
- *Principais métodos*
  - *openStream() obtém um InputStream para os dados*
  - *openConnection(): retorna um objeto URLConnection que contém métodos para ler o cabeçalho dos dados*
  - *getContent(): retorna os dados diretamente como Object se conteúdo for conhecido (texto, imagens, etc.)*
- *Para imprimir a página HTML de um site*

```
try {
   URL url = new URL("http://www.site.com");
   InputStreamReader reader =
      new InputStreamReader(url.openStream());
   BufferedReader br = new BufferedReader(reader);
   String linha = "";
   while ( (linha = br.readLine()) != null) {
      System.out.println(linha);
   }
} catch (MalformedURLException e) { ... }
```

InetAddress

- Representa um endereço Internet
- Principais métodos estáticos construtores
  - getLocalHost() retorna InetAddress
  - getByName(String host) retorna InetAddress
- Principais métodos de instância
  - getHostAddress() retorna String com IP do InetAddress
  - getHostName() retorna String com nome no InetAddress
- Para descobrir o IP e nome da máquina local:

```
InetAddress address = InetAddress.getLocalHost();
String ip = address.getHostAddress();
String nome = address.getHostName();
```

Socket

- *Um dos lados de uma conexão bidirecional TCP*
- *Principais métodos servem para obter fluxos de entrada e saída*
  - *getInputStream()*
  - *getOutputStream()*
  - *close()*
- *Exemplo*

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

- *Depois de obtido os fluxos, basta ler ou enviar dados*

Socket (2)

- *Para ler ou gravar caracteres ao invés de bytes, pode-se decorar os fluxos obtidos de um socket com as classes Reader e Writer:*

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

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

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

// Use aqui os fluxos de dados

con.close();
```

ServerSocket

- *Com ServerSocket pode-se implementar um servidor que fica escutando uma porta a espera de um cliente*
- *Principal método*
  - *accept(): aceita a conexão e retorna o seu socket*
- *Exemplo de servidor dedicado*

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

Exceções de rede

- Várias exceções podem ocorrer em um ambiente de rede
  - O programa deve tomar medidas para reduzir o impacto das exceções inevitáveis, como rede fora do ar ou conexão recusada
  - O compilador irá informar, durante o desenvolvimento, as exceções que precisam ser declaradas ou tratadas
- As exceções mais comuns do pacote java.net são
  - SocketException
  - MalformedURLException
  - UnknownHostException
  - ProtocolException
- Operações de timeout, liberação de threads, sincronização, transações, etc. devem ser implementados pelo programador em aplicações de rede
  - Não há exceções tratando esses problemas automaticamente

Exercícios

- 1. Escreva um programa que descubra e imprima 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
- 3. Servidor dedicado: escreva um servidor simples que responda a comandos da forma "GET arquivo".
  - Localize o arquivo e imprima-o no OutputStream
  - Escreva um cliente que receba o arquivo e grave-o localmente

Exercícios (2)

- 4. Servidor multithreaded: 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 irá tratar as requisições do cliente.

  O servidor deverá imprimir na tela todos os comandos enviados por todos os clientes.
  - Os clientes enviam mensagens de texto, como um chat.
- 5. Crie um cliente para a aplicação acima e teste-o em várias máquinas diferentes.

Curso J100: Java 2 Standard Edition

Revisão 17.0

(helder@acm.org)

argonavis.com.br