Licenciatura em Tecnologias da Informação
Tecnologias de Integração
Sockets

Um `Socket` é um ponto terminal numa comunicação bidirecional entre duas aplicações na rede. Existem dois tipos de `Sockets`:

- `Stream Sockets` - utilizam o protocolo TCP e a informação pode circular entre as aplicações enquanto a ligação estiver ativa.
- `Datagram Sockets` - utilizam o protocolo UDP e permitem a transmição de pacotes individuais de informação

Implementação em Java

A package `java.net` possui todas as classes para a comunicação na rede, o que inclui as classes:

- `Socket` que implementa um dos lados da ligação entre a aplicação Java e outra aplicação Java na rede
- `SocketServer` que permite implementar um servidor `socket` que escuta e aceita ligações
- `DatagramSocket` que permite a implementação tanto do cliente como do servidor de um `datagram socket`
- `DatagramPacket` que representa um pacote a enviar

Stream socket: Servidor

1. Criar o objeto **ServerSocket**

```
1 ServerSocket s = new ServerSocket( int portNumber, int queueLength);
```

   - O construtor especifica o porto onde está à espera de ligações de clientes e o número máximo de clientes em espera
   - Apenas pode existir uma ligação a um porto em determinado momento

2. Colocar o servidor à espera de uma tentativa de ligação

```
1 Socket con = s.accept();
```

   - Cada ligação de cliente é gerida por um objeto do tipo **Socket**
   - O Socket permite que o servidor comunique com o cliente
   - As interações com o cliente são efetuadas num porto diferente do porto de escuta

3. Criar os objetos necessários para a troca de mensagens entre cliente e servidor recorrendo aos métodos **getOutputStream** (envio) e **getInputStream** (receção)

```
ObjectInputStream in = new ObjectInputStream(con.getInputStream());
ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream());
```

4. Enviar e receber mensagens e processar informação
   - O cliente e o servidor comunicam recorrendo aos objetos **OutputStream** e **InputStream**
5. Terminar a ligação

```
con.close();
s.close();
```

Stream socket: Cliente

1. Criar Socket para efetuar ligação ao servidor

   ```
   Socket con = new Socket(String server, int port);
   ```

2. Criar os objetos necessários para a troca de mensagens entre cliente e servidor recorrendo aos métodos **getOutputStream** (envio) e **getInputStream** (receção)
3. Enviar e receber mensagens e processar informação
   - O cliente e o servidor comunicam recorrendo aos objetos **OutputStream** e **InputStream**
4. Terminar a ligação quando se conclui a comunicação

Datagram socket

1. Criar um objeto **DatagramSocket**

```
DatagramSocket ds = new DatagramSocket(int portNumber);
```

   - O construtor especifica o porto onde estará à escuta de pacotes
   - Determina o endereço e o porto fonte dos pacotes

2. Enviar pacotes

```
ds.send(DatagramPacket packet);
```

3. Receber pacotes

```
ds.receive(DatagramPacket packet);
```

Datagram packet

1. Pacote a enviar

```
DatagramPacket dp = new DatagramPacket(byte[] buffer, int lenght,
    InetAddress address, int port);
```

   - O construtor especifica a informação a enviar, a sua dimensão, o endereço e o porto de destino

2. Pacote a receber

```
DatagramPacket dp = new DatagramPacket(byte[] buffer, int lenght);
```

   - O construtor especifica a informação a receber e a sua dimensão

Exercícios

Todos os pontos que se seguem devem ser resolvidos utilizando Stream e Datagram Sockets.

1. Crie duas aplicações, uma cliente e outra servidor, que interajam entre si. O cliente deve enviar uma mensagem de texto, aguardar pela resposta e apresentá-la no terminal. O servidor deverá receber a mensagem de texto, apresentá-la no terminal e deverá devolvê-la com os caracteres na ordem inversa.

2. Crie duas aplicações, uma cliente e outra servidor, que interajam entre si de forma a que o cliente solicite a data e hora e apresente a informação após receber a resposta e o servidor responda com um objecto com a data e hora atual.

3. Crie duas aplicações, uma cliente e outra servidor, que interajam entre si. O cliente deve solicitar o download de um ficheiro a partir do servidor. O servidor deverá enviar o ficheiro para o cliente, caso exista, ou a informação de recurso não encontrado, no caso contrário.

4. Crie duas aplicações, uma cliente e outra servidor. O cliente deverá solicitar informação sobre um curso, enviando o código do curso, após receber a informação, deverá apresentá-la no terminal. O servidor deverá receber o código do curso e deverá responder com um objeto que contenha a informação sobre um curso (nome, área científica, número de alunos, ...)