Serialização de Dados

Grupo de Sistemas Distribuídos
Universidade do Minho

1 Objetivos

Uso de facilidades de biblioteca em Java relativas à serialização de tipos de dados simples para implementação de serialização recursiva. Comunicação entre cliente e servidor através de mensagens em binário através das facilidades desenvolvidas.

2 Mecanismos

Package `java.io`:

- classe `DataInputStream`:
  - construtor `DataInputStream(InputStream in)`
  - métodos `readInt()`, `readLong()`, `readUTF()`, `readBoolean()`
- classe `DataOutputStream`:
  - construtor `DataOutputStream(OutputStream out)`
  - métodos: `writeInt(int v)`, `writeLong(long v)`, `writeUTF(String str)`, `writeBoolean()`, `flush()`

3 Exercícios propostos

Partindo dos excertos de código disponibilizados, resolva as seguintes exercícios.

1 Implemente os métodos que permitem fazer a serialização da classe `Contact`, nomeadamente:

```
void serialize (DataOutputStream out) throws IOException { ... }
static Contact deserialize (DataInputStream in)}throws IOException { ... }
```

2 Implemente um servidor que permite fazer a gestão de uma lista de contactos. O servidor deverá atender clientes concorrentemente. Cada pedido corresponde à criação (se ainda não existe) ou actualização (se já existe o nome) de um contacto (classe `Contact`), devendo este ser recebido em formato binário. A conexão é terminada quando se der um *end of file* no stream de leitura.

3 Implemente um cliente para o servidor desenvolvido no exercício anterior. O cliente lê continuamente do `stdin` a informação de um contacto em representação textual, conforme apresentado a seguir.

```
John 20 253123456 Company john@mail.com john.business@mail.com
```

4 Modifique o cliente e o servidor de forma a que, após se conectar ao servidor, o cliente receba a lista de todos os contactos existentes. Os contactos deverão ser enviados utilizando serialização de dados no formato binário. Para tal, complete a classe `ContactList` para ser usada na serialização de uma lista de contactos. Tenha em conta potenciais problemas de acesso concorrentemente durante a serialização da lista, e pondere os objectos `Contact` serem imutáveis, para poderem ser facilmente partilhados entre threads.

4 Exercícios adicionais

1 Estenda a funcionalidade do servidor de modo a simular uma *rede social*. Cada contacto pode agora ter uma lista de *amigos*, que aponta para outros contactos. Na sua implementação, tenha em conta que podem haver *ciclos de amizade*. Por exemplo, assumindo que notação $A \rightarrow B$ indica que o contacto A é amigo do contacto B, é possível que haja o ciclo $A \rightarrow B \rightarrow C \rightarrow A$. Modifique o servidor e os clientes de forma a suportar este novo requisito, explorando várias alternativas protocolares.