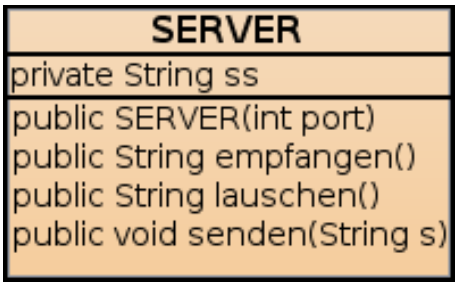
1-Server-1-Client-Kommunikation

Hier wird das einfachste Szenario beschreiben, bei dem eine Netzwerk-Kommunikation zwischen zwei Rechnern nach dem Client-Server-Prinzip statt findet.

Dabei fungiert ein Rechner als Server und der andere als Client. Der Server wird gestartet und wartet dann so lange, bis sich ein Client mit ihm verbindet. Von nun an können beide Seiten beliebige Strings an die Gegenstelle senden bzw. von dort empfangen.

Der Server



Konstruktor

```
Server( int port )
```

Erzeugt eine Server-Instanz auf diesem Rechner, welche auf dem gegebenen Port auf die Verbindungs-Anfrage eines Clients wartet.

Kommunikations-Methoden

Nachdem sich ein Client mit dem Server verbunden hat, stehen folgende Methoden zur Verfügung:

```
senden( String nachricht )
```

Sendet die Nachricht an den Client. Diese landet dort automatisch in einem Puffer.

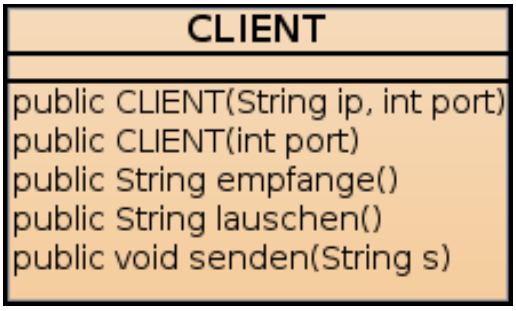
```
empfangen( )
```

Liest die nächste Nachricht als String aus dem eigenen Puffer.
Ist der Puffer leer, wird automatisch die Methode `lauschen()` aufgerufen.

```
lauschen( )
```

Wartet (und blockiert das Programm), bis vom Client ein weiterer `String` gesendet wird, außer, der eigene Puffer ist nicht leer, dann wird `empfangen()` aufgerufen.

Der Client



Konstruktoren

```
Client( int port )
```

Erzeugt eine Client-Instanz, welche sich auf `localhost` mit dem Server auf dem angegebenen Port verbindet. *(Netzwerk-Kommunikation auf dem eigenen Rechner)*

```
Client( String ip , int port )
```

Erzeugt eine Client-Instanz, welche sich mit einem Server auf dem Rechner mit der angegebenen IP-Adresse auf dem angegebenen Port verbindet.

(echte Netzwerk-Kommunikation zwischen zwei verschiedenen Rechnern)

Kommunikations-Methoden

Nachdem sich ein Client mit dem Server verbunden hat, stehen dieselben Methoden zur Verfügung wie auf Server-Seite:

```
senden( String nachricht )
```

Sendet die Nachricht an den Server. Diese landet dort automatisch in einem Puffer.

```
empfangen( )
```

Liest die nächste Nachricht als String aus dem eigenen Puffer.
Ist der Puffer leer, wird automatisch die Methode `lauschen()` aufgerufen.

```
lauschen( )
```

Wartet (und blockiert das Programm), bis vom Server ein weiterer `String` gesendet wird, außer, der eigene Puffer ist nicht leer, dann wird `empfangen()` aufgerufen.