Netzklassen

Die Klasse `Connection`

Objekte der Klasse `Connection` ermöglichen eine Netzwerkverbindung zu einem Server mittels TCP/IP-Protokoll. Nach Verbindungsaufbau können Zeichenketten (`Strings`) zum Server gesendet und von diesem empfangen werden. Zur Vereinfachung geschieht dies zeilenweise, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfang wird dieser entfernt. Es findet nur eine rudimentäre Fehlerbehandlung statt, so dass z. B. der Zugriff auf unterbrochene oder bereits getrennte Verbindungen nicht zu einem Programmabbruch führt. Eine einmal getrennte Verbindung kann nicht reaktiviert werden.

Dokumentation der Klasse `Connection`

`Connection(String pServerIP, int pServerPort)`

Ein Objekt vom Typ `Connection` wird erstellt. Dadurch wird eine Verbindung zum durch `pServerIP` und `pServerPort` spezifizierten Server aufgebaut, so dass Daten (Zeichenketten) gesendet und empfangen werden können. Kann die Verbindung nicht hergestellt werden, kann die Instanz von `Connection` nicht mehr verwendet werden.

`void send(String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner ergänzt – an den Server gesendet. Schlägt der Versand fehl, geschieht nichts.

`String receive()`

Es wird beliebig lange auf eine eingehende Nachricht vom Server gewartet und diese Nachricht anschließend zurückgegeben. Der vom Server angehängte Zeilentrenner wird zuvor entfernt. Während des Wartens ist der ausführende Prozess blockiert. Wurde die Verbindung unterbrochen oder durch den Server unvermittelt geschlossen, wird `null` zurückgegeben.

`void close()`

Die Verbindung zum Server wird getrennt und kann nicht mehr verwendet werden. War die Verbindung bereits getrennt, geschieht nichts.

Die Klasse `Client`

Objekte von Unterklassen der abstrakten Klasse `Client` ermöglichen Netzwerkverbindungen zu einem Server mittels TCP/IP-Protokoll. Nach Verbindungsaufbau können Zeichenketten (`Strings`) zum Server gesendet und von diesem empfangen werden, wobei der Nachrichtenempfang nebenläufig geschieht. Zur Vereinfachung finden Nachrichtenversand und -empfang zeilenweise statt, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfang wird dieser entfernt. Jede empfangene Nachricht wird einer Ereignisbehandlungsmethode übergeben, die in Unterklassen implementiert werden muss. Es findet nur eine rudimentäre Fehlerbehandlung statt, so dass z. B. Verbindungsabbrüche nicht zu einem Programmabbruch führen. Eine einmal unterbrochene oder getrennte Verbindung kann nicht reaktiviert werden.

Dokumentation der Klasse `Client`

`Client(String pServerIP, int pServerPort)`

Es wird eine Verbindung zum durch `pServerIP` und `pServerPort` spezifizierten Server aufgebaut, so dass Daten (Zeichenketten) gesendet und empfangen werden können. Kann die Verbindung nicht hergestellt werden, kann der Client nicht zum Datenaustausch verwendet werden.

`boolean isConnected()`

Die Anfrage liefert den Wert `true`, wenn der Client mit dem Server aktuell verbunden ist. Ansonsten liefert sie den Wert `false`.

`void send(String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner ergänzt – an den Server gesendet. Schlägt der Versand fehl, geschieht nichts.

`void close()`

Die Verbindung zum Server wird getrennt und der Client kann nicht mehr verwendet werden. Ist Client bereits vor Aufruf der Methode in diesem Zustand, geschieht nichts.

`void processMessage(String pMessage)`

Diese Methode wird aufgerufen, wenn der Client die Nachricht `pMessage` vom Server empfangen hat. Der vom Server ergänzte Zeilentrenner wurde zuvor entfernt. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Client` überschrieben werden, so dass auf den Empfang der Nachricht reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

Die Klasse `Server`

Objekte von Unterklassen der abstrakten Klasse `Server` ermöglichen das Anbieten von Serverdiensten, so dass Clients Verbindungen zum Server mittels TCP/IP-Protokoll aufbauen können. Zur Vereinfachung finden Nachrichtenversand und -empfang zeilenweise statt, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfang wird dieser entfernt. Verbindungsannahme, Nachrichtenempfang und Verbindungsende geschehen nebenläufig. Auf diese Ereignisse muss durch Überschreiben der entsprechenden Ereignisbehandlungsmethoden reagiert werden. Es findet nur eine rudimentäre Fehlerbehandlung statt, so dass z. B. Verbindungsabbrüche nicht zu einem Programmabbruch führen. Einmal unterbrochene oder getrennte Verbindungen können nicht reaktiviert werden.

Dokumentation der Klasse `Server`

`Server(int pPort)`

Ein Objekt vom Typ `Server` wird erstellt, das über die angegebene Portnummer einen Dienst anbietet an. Clients können sich mit dem Server verbinden, so dass Daten (Zeichenketten) zu diesen gesendet und von diesen empfangen werden können. Kann der Server unter der angegebenen Portnummer keinen Dienst anbieten (z. B. weil die Portnummer bereits belegt ist), ist keine Verbindungsaufnahme zum Server und kein Datenaustausch möglich.

`boolean isOpen()`

Die Anfrage liefert den Wert `true`, wenn der Server auf Port `pPort` einen Dienst anbietet. Ansonsten liefert die Methode den Wert `false`.

`boolean isConnectedTo(String pClientIP, int pClientPort)`

Die Anfrage liefert den Wert `true`, wenn der Server mit dem durch `pClientIP` und `pClientPort` spezifizierten Client aktuell verbunden ist. Ansonsten liefert die Methode den Wert `false`.

`void send(String pClientIP, int pClientPort, String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an den durch `pClientIP` und `pClientPort` spezifizierten Client gesendet. Schlägt der Versand fehl, geschieht nichts.

`void sendToAll(String pMessage)`

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an alle mit dem Server verbundenen Clients gesendet. Schlägt der Versand an einen Client fehl, wird dieser Client übersprungen.

`void closeConnection(String pClientIP, int pClientPort)`

Die Verbindung des Servers zu dem durch `pClientIP` und `pClientPort` spezifizierten Client wird getrennt. Zuvor wird die Methode `processClosingConnection` mit IP-Adresse und Port des jeweiligen Clients aufgerufen. Ist der Server nicht mit dem in der Parameterliste spezifizierten Client verbunden, geschieht nichts.

`void close()`

Alle bestehenden Verbindungen zu Clients werden getrennt und der Server kann nicht mehr verwendet werden. Ist der Server bereits vor Aufruf der Methode in diesem Zustand, geschieht nichts.

`void processNewConnection(String pClientIP, int pClientPort)`

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn sich ein Client mit IP-Adresse `pClientIP` und Portnummer `pClientPort` mit dem Server verbunden hat. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf den Neuaufbau der Verbindung reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

`void processMessage(String pClientIP, int pClientPort, String pMessage)`

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn der Server die Nachricht `pMessage` von dem durch `pClientIP` und `pClientPort` spezifizierten Client empfangen hat. Der vom Client hinzugefügte Zeilentrenner wurde zuvor entfernt. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf den Empfang der Nachricht reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

`void processClosingConnection(String pClientIP, int pClientPort)`

Sofern der Server die Verbindung zu dem durch `pClientIP` und `pClientPort` spezifizierten Client trennt, wird diese Ereignisbehandlungsmethode aufgerufen, unmittelbar bevor die Verbindungstrennung tatsächlich erfolgt. Wird die Verbindung unvermittelt unterbrochen oder hat der in der Parameterliste spezifizierte Client die Verbindung zum Server unvermittelt getrennt, erfolgt der Methodenaufruf nach der Unterbrechung / Trennung der Verbindung. Die Methode ist abstrakt und muss in einer Unterklasse der Klasse `Server` überschrieben werden, so dass auf das Ende der Verbindung zum angegebenen Client reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.