

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

Konstruktor **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.

Anfrage **boolean isOpen()**

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

Anfrage **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.

Auftrag **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.

Auftrag **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.

Auftrag

**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.

Auftrag

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.

Auftrag

**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.

Auftrag

**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.

Auftrag

**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.