Übersicht der Klassen

# Package Client

|  |  |
| --- | --- |
| AbstractChatClient | Gemeinsame Funktionalität für alle Client-Implementierungen |
| AbstractMessageListenerThread | Abstrakte Klasse mit Basisfunktionalität für clientseitige Message-Processing-Threads |
| AdvancedMessageListenerThreadImpl | Thread wartet auf ankommende Nachrichten vom Server und bearbeitet diese. |
| ClientCommunication (Interface) | Interface zur Kommunikation des Chat-Clients mit dem Chat-Server |
| ClientFxGUI | Chat-GUI |
| ClientImpl | Verwaltet eine Verbindung zum Server |
| ClientModel | Modelldaten für FX-GUI |
| ClientUserInterface (Interface) | Interface zur Ausführung von Aktionen über die Präsentationslogik |
| LoggedInGuiController | Controller für Chat-GUI |
| LogInGuiController | Controller für Login-GUI |
| SharedClientData | Gemeinsame genutzte Daten, die sich der Chat-Client-Thread und die Message-Processing-Threads teilen |
| SimpleMessageListenerThreadImpl | Thread wartet auf ankommende Nachrichten vom Server und bearbeitet diese |

# Package Common

|  |  |
| --- | --- |
| AuditLogImplementationType (Enum) | Implementierungsvarianten des Lasttests mit verschiedenen Transportprotokollen |
| AuditLogPDU | Nachrichtenaufbau für das AuditLog-Protokoll |
| AuditLogPduType (Enum) | Enumeration zur Definition der Audit-PDU-Typen |
| ChatPDU | Nachrichtenaufbau für Chat-Protokoll (für alle Nachrichtentypen: Request, Response, Event, Confirm) |
| ClientConversationStatus (Enum) | UNREGISTERED; REGISTERING; REGISTERED; UNREGISTERING |
| ClientListEntry | Eintrag in der serverseitigen Clientliste zur Verwaltung der angemeldeten User inkl. des Conversation-Status. |
| CpuUtilisationWatch | Ermitteln der durchschnittlich verbrauchten CPU-Zeit eines Prozesses |
| DistributionMetrics | Metriken zur Bewertung der Verteilung der gemessenen RTTs |
| ExceptionHandler |  |
| ImplementationType (Enum) | Implementierungsvarianten des Lasttests mit verschiedenen Transportprotokollen |
| MemoryTest | Testklasse für einen Heap-Test. Klasse dient nur zum Test und ist nicht für die Studienarbeit relevant |
| PduType (Enum) | Enumeration zur Definition der Chat-PDU-Typen |
| SharedClientStatistics | Die Klasse sammelt Statistikdaten zur Ermittlung von Round Trip Times (RTT) für einen Test zur Kommunikation zwischen mehreren Client-Threads und einem Server.  Die Daten werden in einem Array gesammelt, das einen Eintrag für jeden Client enthält.  Jeder Client erhält eine Nummer, die als Zugriffsindex auf das Array verwendet wird. |
| SystemConstants | Bezeichnungen der verschiedenen Server-Implementierungen und Farben der GUI |
| TestNanoTime | Testklasse für das Lesen eines Zeitstempels |

# Package Connection

|  |  |
| --- | --- |
| Connection (Inferface) | Wird vom Client und vom Server zur Kommunikation verwendet |
| ConnectionFactory (Interface) | Erstellt neue Instanzen von {@link Connection};  erstellt eine Verbindung zum Server |
| ConnectionTimeoutException | Timeout-Exception bei Verbindung |
| DecoratingConnectionFactory | Stellt beim Verbindungsaufbau sicher, dass eine {@link Connection} um Logging-Funktionen erweitert wird |
| EndOfFileException | End of File bei Verbindung, verursacht durch einen Verbindungsabbau des Partners |
| LoggingConnectionDecorator | Stattet ein {@link Connection} Objekt mit automatischem Logging aus.  Umschließt eine beliebige Connection-Instanz und bietet dieselbe Schnittstelle an.    Beim Aufruf einer Methode wird zunächst eine Log-Ausgabe getätigt und danach die Methode der umschlossenen Connection aufgerufen.  Anschließend erfolgt eine weitere Log-Ausgabe. |
| ServerSocketInterface (Interface) | Stellt eine einheitliche Schnittstelle für Implementierungen verschiedener Protokolle (z.B. TCP oder UDP) dar |

# Package Server

|  |  |
| --- | --- |
| AbstractChatServer | Gemeinsame Attribute für alle Implementierungen |
| AbstractWorkerThread | Abstrakte Klasse mit Basisfunktionalität für serverseitige Worker-Threads |
| AdvancedChatServerImpl | Advanced-Chat-Server-Implementierung  Der ChatServer wird in einem eigenen Thread gestartet.  Er nimmt alle Verbindungsaufbauwünsche der ChatClients entgegen und startet für jede Verbindung jeweils einen eigenen Worker-Thread |
| AdvancedChatWorkerThreadImpl | Worker-Thread zur serverseitigen Bedienung einer Session mit einem Client.  Jedem Chat-Client wird serverseitig ein Worker-Thread zugeordnet |
| AuditLogConnection | Verwaltet eine logische Verbindung zum AuditLog-Server über UDP oder TCP |
| ChatServerGUI | Benutzeroberfläche zum Starten des Chat-Servers |
| ChatServerGuiInterface (Interface) | Interface, das der ServerGUI bereitstellen muss |
| ChatServerInterface (Interface) | Einheitliche Schnittstelle aller Server |
| ServerFactory | Übernimmt die Konfiguration und Erzeugung bestimmter Server-Typen |
| ServerStartData | Getter und Setter für startTime |
| SharedChatClientList | Liste aller angemeldeten Clients.  Diese Liste wird im Server als Singleton verwaltet (darf nur einmal erzeugt werden). Alle Worker-Threads im Server nutzen diese Liste.    Die Liste wird als HashMap organisiert.  Als Schlüssel wird der Username von Clients verwendet.    Genereller Hinweis: Zur Umgehung von ConcurrentModificationExceptions wird bei der Iteration durch Listen generell eine Kopie der Liste angelegt. |
| SharedServerCounter | Globale Zähler für Logouts, gesendete Events und empfangene Confirms nur zum Test |
| SimpleChatServerImpl | Simple-Chat-Server-Implementierung |
| SimpleChatWorkerThreadImpl | Worker-Thread zur serverseitigen Bedienung einer Session mit einem Client.  Jedem Chat-Client wird serverseitig ein Worker-Thread zugeordnet. |

# Package TCP

|  |  |
| --- | --- |
| TcpConnection | Implementierung der TCP-Verbindung |
| TcpConnectionFactory | Erzeugen von TCP-Verbindungen zum Server |
| TcpServerSocket | Server-Socket Implementierung auf TCP-Basis |

# Package UDP

|  |  |
| --- | --- |
| UdpClientConnection |  |
| UdpClientConnectionFactory |  |
| UdpPseudoConnectionContext | UDP ist ein zustandsloses Protokoll, in dem es eigentlich keine Verbindung gibt.  Hier werden Verbindungsinformationen der ankommenden Datagramme gespeichert, damit der Server seine Antwort an den richtigen Client schicken kann. |
| UdpServerConnection |  |
| UdpServerSocket |  |
| UdpSocket | Klasse UdpSocket    Diese Klasse kapselt die Datagram-Sockets und stellt eine etwas komfortablere Schnittstelle zur Verfügung.    Der Mehrwert dieser Klasse im Vergleich zur Standard-Datagram-Socket-Klasse ist die Nutzung eines Objektstroms zur Kommunikation über UDP.    Achtung: Maximale Datagramlänge: 64 KByte |