**TGM Wien**

**JMS Chat**

**DEZSYS-06**

**Thomas Taschner & Michael Weinberger 4AHITT**

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc404722392)

[Beschreibung auf Moodle 2](#_Toc404722393)

[Designüberlegung 3](#_Toc404722394)

[Erster Ansatz 3](#_Toc404722395)

[Konkrete Idee 3](#_Toc404722396)

[Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung) 4](#_Toc404722397)

[Arbeitsdurchführung (Resultate/Niederlagen) 5](#_Toc404722398)

[Testbericht 7](#_Toc404722399)

[Quellenangaben 8](#_Toc404722400)

# Aufgabenstellung

## Beschreibung auf Moodle

Implementieren Sie eine Chatapplikation mit Hilfe des Java Message Service. Verwenden Sie Apache ActiveMQ als Message Broker Ihrer Applikation. Das Programm soll folgende Funktionen beinhalten:

* Benutzer meldet sich mit einem Benutzernamen und dem Namen des Chatrooms an.   
  Beispiel für einen Aufruf:   
    
  vsdbchat <ip\_message\_broker> <benutzername> <chatroom>
* Der Benutzer kann in dem Chatroom (JMS Topic) Nachrichten an alle Teilnehmer eine Nachricht senden und empfangen.   
  Die Nachricht erscheint in folgendem Format:  
    
  <benutzername> [<ip\_des\_benutzers>]: <Nachricht>
* Zusätzlich zu dem Chatroom kann jedem Benutzer eine Nachricht in einem persönlichen Postfach (JMS Queue) hinterlassen werden. Der Name des Postfachs ist die IP Adresse des Benutzers (Eindeutigkeit).  
    
  Nachricht an das Postfach senden:   
  MAIL <ip\_des\_benutzers> <nachricht>  
    
  Eignes Postfach abfragen:   
  MAILBOX
* Der Chatraum wird mit dem Schlüsselwort EXIT verlassen. Der Benutzer verlässt den Chatraum, die anderen Teilnehmer sind davon nicht betroffen.

# Designüberlegung

## Erster Ansatz

Mit den Erkenntnissen des KnockKnock-Servers aus den vorherigen Einheiten haben wir bereits einen guten Einblick bekommen, wie Nachrichtenübermittlung ohne Middleware funktioniert. Mithilfe von Apache ActiveMQ wollen wir es nun auch völlig unterschiedlichen Systemen möglich machen, miteinander zu kommunizieren. Die MOM (‚Message Oriented Middleware‘) handelt als Broker ‚in der Mitte‘ die Kommunikation ab, sodass die Clients nur mehr Befehle zum Senden & Empfangen geben müssen.

## Konkrete Idee

Bestehend auf dem Wissen der vorigen Beispiele, gab es die Idee, dass die 2 Mechanismen Sender & Receiver in einer kontrollierten ‚Endlosschleife‘ Nachrichten entgegennehmen und den Parametern entsprechend verschicken. Auf einem System müssen, um beide Funktionen zu erfüllen, sprich die eines vollwertigen Chats mit Senden & Empfangen von Nachrichten zu gewährleisten, also beide Instanzen ausgeführt werden.

# Detaillierte Arbeitsaufteilung (Aufwandsabschätzung, Endzeitaufteilung)

# Arbeitsdurchführung (Resultate/Niederlagen)

# Testbericht

# Quellenangaben