
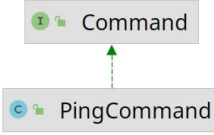
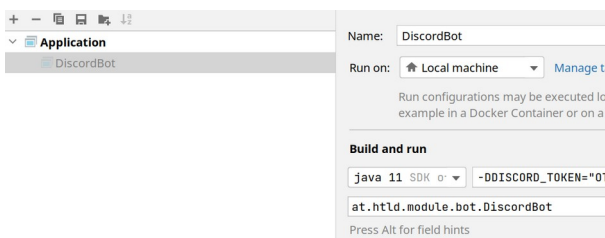
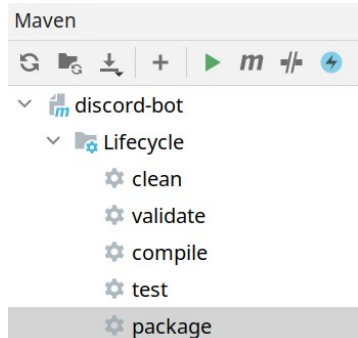


Fach/Klasse	SWP1	4 JG						
Themenbereich	Objektorientierte Programmierung / OO Modellierung mit UML							
bsp03b Discord Bot	<div></div> <p>Erstelle einen Telegram Bot, welcher folgende Kommandos ausführen kann:</p> <table><thead><tr><th><i>Kommando</i></th><th><i>Aktion</i></th></tr></thead><tbody><tr><td>?echo ECHO_STRING</td><td>Der Bot liefert den ECHO_STRING zurück</td></tr><tr><td>?reverse FORWARD_STRING</td><td>Liefer die Eingabe FORWARD_STRING in der umgekehrten Zeichenfolge</td></tr></tbody></table>	<i>Kommando</i>	<i>Aktion</i>	?echo ECHO_STRING	Der Bot liefert den ECHO_STRING zurück	?reverse FORWARD_STRING	Liefer die Eingabe FORWARD_STRING in der umgekehrten Zeichenfolge	
<i>Kommando</i>	<i>Aktion</i>							
?echo ECHO_STRING	Der Bot liefert den ECHO_STRING zurück							
?reverse FORWARD_STRING	Liefer die Eingabe FORWARD_STRING in der umgekehrten Zeichenfolge							
Modellierung I	<ul style="list-style-type: none">● Modelliere das Usecasediagramm für die Kommandos des Bots● Speicher das UseCase-Diagramm auch als Image unter „model/Bot_UseCaseDiagramm.png“							
Modellierung II	<ul style="list-style-type: none">● Erstelle ein Klassendiagramm für einen Bot mit den zwei Kommandos echo und reverse● Zum Starten wird die Klasse <code>at.htld.module.bot.DiscordBot</code> verwendet.● Speicher das Klassen-Diagramm auch als Image unter „model/Bot_ClassDiagramm.png“							
Codierung	<ul style="list-style-type: none">● Erstelle den Bot mit den angeführten Kommandos.							
Modellierung III	<ul style="list-style-type: none">● Skizziere ein Aktivitätendiagramm zu den einzelnen Kommandos des Bots● Speichere das Diagramm als png-File „Bot_Activity_Diagram.png“ ab.							
Modellierung IV	<ul style="list-style-type: none">● Modelliere mit einem Sequenzdiagramm den Start des Bots und den Ablauf bei Erhalt des Kommandos „reverse“.● Speichere das Sequenzdiagramm mit dem Namen „Bot_SequenceDiagram.png“ als ab.							
Abgabe / Bewertung	<p>Gib das Beispiel termingerecht in Teams ab!</p> <ul style="list-style-type: none">● Use-Case-Diagramm● Klassendiagramm mit Packages● Lauffähiges Programm (ohne BotToken und BotUsername)● Activity-Diagramm● Sequence-Diagramm● Einhaltung Namenskonventionen + Codierrichtlinien - siehe Style-Guide							

Fach/Klasse	SWP1 Hilfestellung Discord Bot	4 JG
Bot erstellen	<p>1) Applikation und Bot erstellen - siehe https://www.baeldung.com/spring-discord4j-bot</p> <p>2) Bot zu einem Server hinzufügen</p> <ul style="list-style-type: none"> • https://discord.com/developers/applications • Menü OAuth2 Bot auswählen • OAuth2 Url Generator – bot auswählen • Url kopieren und im Browser aufrufen – anschließend den Server auswählen, auf welchem der Bot registriert werden soll 	
Discord4j	<p>Siehe Coding</p> <ul style="list-style-type: none"> • at.htld.module.bot.DiscordBot und Implementierung  <pre> classDiagram class Command class PingCommand PingCommand -- > Command </pre> <ul style="list-style-type: none"> • Aufruf des Bots mit den Bot-Token – der Parameter „DISCORD_TOKEN“ muss beim Programmstart übergeben werden: 	
Bot als Docker Image / Container	<ul style="list-style-type: none"> • Über-Jar File erstellen (siehe pom.xml File Plugin maven-assembly-plugin) • Jar-File erstellen Maven – package – Goal  <ul style="list-style-type: none"> • jar-File <code>target/discord-bot-0.0.1-SNAPSHOT-jar-with-dependencies.jar</code> und <code>src/main/resources/docker/Dockerfile</code> auf VM mit scp-Command kopieren: <pre>scp discord-bot-0.0.1-SNAPSHOT-jar-with-dependencies.jar xxxx@10.115.2.xxx:/home/xxxx/.</pre> <ul style="list-style-type: none"> • Login auf VM – Docker Image erstellen <pre>docker build . -t discord-bot:0.0.1</pre> <ul style="list-style-type: none"> • Docker Container erstellen <pre>docker run -e "DISCORD_TOKEN=fill_in_my_token" --name=discord-bot -d discord-bot:0.0.1</pre>	