**TASK 360T**

| **REVISION HISTORY** | | | |
| --- | --- | --- | --- |
| **Version** | **Description** | **Changed By** | **Date** |
| 1.0 | Initial Document | Ramazan Erecir | 08.01.2020 |
|  |  |  |  |
|  |  |  |  |

# ABSTRACT

This document provides solution for the given task (See Task Section), explanations about the class responsibilities in the project (See Class Responsibility Section), steps to build project source code (See Build Project Section) and steps to run test cases (See Test Section).

# TASK

Having a Player class - an instance of this class with that can communicate with other Player(s) (other instances of this class)

The use case for this task is as bellow:

1. create 2 players

2. one of the players should send a message to second player (let's call this player "initiator")

3. when a player receives a message should send back a new message that contains the received message concatenated with the message counter that this player sent.

4. finalize the program (gracefully) after the initiator sent 10 messages and received back 10 messages (stop condition)

5. both players should run in the same java process (strong requirement)

6. document for every class the responsibilities it has.

7. additional challenge (nice to have) opposite to 5: have every player in a separate JAVA process (different PID).

Please use pure Java as much as possible (no additional frameworks like spring, etc.)

Please deliver one single maven project with the source code only (no jars). Please send the maven project as archive attached to e-mail (eventual links for download will be ignored due to security policy).

Please provide a shell script to start the program.

Everything what is not clearly specified is to be decided by developer. Everything what is specified is a hard requirement.

Please focus on design and not on technology, the technology should be the simplest possible that is achieving the target.

The focus of the exercise is to deliver the cleanest and clearest design that you can achieve (and the system has to be functional).

# CLASS RESPONSIBILITY

**PACKAGE: task360T.model**

**AuthModel**: This class authorizes the player instance to the system. All player instances are authorized by default.

**PACKAGE: task360T.model.communication**

**ListenerClient**: This class creates a client socket to given host name and port number and listens this host:port to receive messages. Received messages are processed if only message is instance of PlayerMessage, other messages are discarded. Only one instance of ListenerClient is created to listen and receive messages.

**SessionManager**: This class organizes the listener client, subscription client and authorized player sessions. For each session a unique id is stored to identify session instance. It also calls close methods of client sessions.

**SubscriptionClient**: This class connects to server socket to send messages. An instance of class is created whenever a client is started to listen server socket. Messages are send through socket connection in this class.

**SubscriptionServer**: This class creates a server socket to broadcast messages. It creates SubscriptionClient instances whenever a client socket connection is opened. Only one instance of this class is created.

**TelegramSerializer**: This class prepares the telegram messages by converting telegram message to xml and vice versa during sending and receiving messages.

**PACKAGE: task360T.model.communication.entity**

**Telegram**: This class is the basic type of message which is sent and receiving during communication. The message is converted to telegram instance while sending and message is created from telegram when it is received.

**TelegramError**: This class is a telegram message which is sent as a response message in case any problem occurred while processing received messages.

**TelegramMessage**: This is an interface class which should be implemented by the message class that is requested to be sent and received during communication.

**VoidTelegram**: This class is an empty telegram message class.

**PACKAGE: task360T.model.entity**

**Credentials**: This class is the credential information of Player class. This is used during authentication.

**Messenger**: This is an interface class which should be implemented by the classes that is requested to send and receive messages.

**Player**: This class sends and receives messages. Player instance should be authenticated to send and receive messages with its credentials.

**PlayerMessage**: The instance of this class is used while sending/receiving messages by Player class instances.

**PACKAGE: task360T.model.service**

**AppData**: This class includes application configuration parameters.

**ApplicationService**: This class includes the main function. Application configuration is loaded, and test cases are started according to given arguments.

**Constants**: This class contains the constant variable names and configuration parameter names.

**MessageManager**: This class authenticates the Player instances to be able to send and receive messages. It also distributes the received and processed messages by ListenerClient to the authenticated active Player instances. Only one instance of this class is created.

**RecoveryService:** This class initiated the Subscription Server and Listener Client server. If any problem occurred, it tries to recover the problem and recreate the socket sessions. It calls the session close methods when the program ends. Only one instance of this class is created.

**PACKAGE: task360T.model.service.enumtype**

**ServiceStatus**: This is an enumeration class to state Application status.

**PACKAGE: task360T.model.service.test**

**TestService**: This is an interface class which should be implemented by Test classes. It includes the isCompleted method to be able to query Test is completed.

**TestService5**: This class is implemented to complete “Task at step 5: both players should run in the same java process (strong requirement)”.

**TestService7**: This class is implemented to complete “Task at step 7: additional challenge (nice to have) opposite to 5: have every player in a separate JAVA process (different PID)”.

# BUILD PROJECT

Maven and Java 8 is required to build source code. Maven pom.xml is provided within the project. Project source code is located under project directory. Please follow the below steps to build the project.

1. Go to project\Task360T directory.
2. Run ‘mvn clean install’ to build project
3. Jar file will be created under project\Task360T\target directory.

# TEST

Please copy the Task360T-1.0-jar-with-dependencies.jar file from project\Task360T\target to test directory. Java 8 is required to run test cases.

Two arguments can be given while running the jar file.

1. First parameter is configuration path where log4j.properties (logger configuration file) and task360T.cfg (application configuration file) are located.
2. Second parameter is used for determining test case. 0 for test case 5. 1 for test case 7 with initiator role. 2 for test case 7 with receiver role.

Test scripts and the configuration files for each test case are provided under test directory. Test scripts are provided as bat and sh file. Any of them can be used according to the execution environment.

There are two scripts to run testcases which are named as below. Once jar file is copied please run below scripts to complete test cases.

1. test5.sh / test5.bat which runs the test case “Task at step 5: both players should run in the same java process (strong requirement)”.

Test 5 configuration path is located as test\test5. Configuration file is configured as below for task 5.

|  |
| --- |
| #Communication  tcpserver.subscription.port.number=9999  tcpserver.listen.port.number=9999  tcpserver.listen.host.name=localhost |

Subscription Server socket is created on port 9999 to broadcast messages.

Listener Client socket is created to listen port on localhost:9999 port to receive messages.

1. test7.sh / test7.bat which runs the test case “Task at step 7: additional challenge (nice to have) opposite to 5: have every player in a separate JAVA process (different PID)”.
2. Test 7 with initiator role test configuration path is located as test\test7\initiator. Configuration file is configured as below for task 7 with initiator role.

|  |
| --- |
| #Communication  tcpserver.subscription.port.number=9999  tcpserver.listen.port.number=9998  tcpserver.listen.host.name=localhost |

Subscription Server socket is created on port 9999 to broadcast messages. This will be listened by receiver’s Listener Client.

Listener Client socket is created to listen port on localhost:9998 port to receive messages. This will listen receiver’s Subscription Server.

1. Test 7 with receiver role test configuration path is located as test\test7\ receiver. Configuration file is configured as below for task 7 with receiver role.

|  |
| --- |
| #Communication  tcpserver.subscription.port.number=9998  tcpserver.listen.port.number=9999  tcpserver.listen.host.name=localhost |

Subscription Server socket is created on port 9998 to broadcast messages. This will be listened by initiator’s Listener Client.

Listener Client socket is created to listen port on localhost:9999 port to receive messages. This will listen initiator’s Subscription Server.

The log file will be created in the current folder where test script is run with the name task360T.log.

**Warning:** In the test cases application is configured to use ports 9999 and 9998. Please reconfigure ports in test case configuration files under test\test5 and test\test7 directories in case these ports are occupied in your environment.

**Note:** Due to security reasons test scripts are created as txt file. Please correct the files as below before running the scripts. You can either use sh or bat files.

test5\_sh.txt > test5.sh  
test7\_sh.txt > test7.sh  
test5\_bat.txt > test5.bat  
test7\_bat.txt > test7.bat

# RELEASED FILES

**Document:** Documentation of the task is provided.

**Project:** Project includes the project sources codes and Maven pom file.

**Test:** Test includes the test scripts and configuration files for the corresponding test scripts.