**Gaiza: Discord Utility Bot**

Technical Manual

*Spring 2020*

**Slippery Rock University**

Cain Kaltenbaugh [cxk1047@sru.edu](mailto:cxk1047@sru.edu)

Table of Contents

1. **Overview**
2. **System Requirements**
3. **Assumptions**
4. **Installation of the Java Development Kit**
5. **Setting PATH in the Java Development Kit**
6. **Installation of the Eclipse IDE**
7. **Setting up the workspace for the bot with Maven**
8. **Configuring the bot’s token**
9. **Understanding implementation of Gaiza packages and class files**
10. **Exporting as an executable jar file**
11. **Overview**

This project utilizes the JavaCord multithreaded libraries to provide a framework for the Discord bot Gaiza to utilize. The bot’s purpose is to expand the functionality within discord and to extend this use to the users. The technical manual will provide the user with in-depth details that rely on the project dependencies, project outline, and functionality of the project setup. The user will be able to implement new commands and utilize different parts of the bot.

1. **System Requirements**
2. Intel core processor or AMD equivalent
3. Microsoft Windows 7 or later
4. Discord
5. Java Development Environment

i . Java Development Kit 1.8.\_

ii. Eclipse Oxygen IDE or equivalent

1. Ability to make a .bat file
2. **Assumptions**
3. Assume the necessary privileges to modify the environment variables involved (Windows).
4. Assume that the user of this manual has access to a reliable internet access point in order to retrieve the resources necessary for the IDE.
5. Assume the user understands how to operate an IDE utilizing the Java Programming Language.
6. Assume the user understands how Discord works and how to navigate inviting and managing bots.
7. Discord is already downloaded on the system from [here](https://discordapp.com/).

**4. Installation of the JDK**

1. Download the latest version of the JDK (You may find this on oracle.com)
2. Run the JDK installer and follow the steps for installation
   1. Select the “Open” option when running the download, the installer should set up automatically
   2. It is assumed (for the project to navigate the necessary files) the “Save”

**5. Setting PATH in the Java Development Kit**

1. With a batch file, temporarily setting PATH variable designated
   1. Open & Run a text editor (i.e., notepad++)
   2. Write and initiate the following line for the PATH variable:
      1. PATH = C:\<path pointing to the JDK>\jdk.1.8.\_\bin
   3. Save the file with the ‘bat’ extension “<filename>.bat”
   4. After saving the file with valid location, traverse to the directory holding the file. This will automatically assign the PATH variable.
      1. \*Note: This will be the temporary variable path. When the operating system is rebooted, the PATH needs to be set again.

\*Note: To permanently set the PATH variable when it is satisfactory, follow these guidelines:

Windows Button > Control Panel > System > Advanced System Settings > Environment Variables > Path > Edit > Overwrite the field with the listing ‘Variable Value’

**6. Installation of the Eclipse IDE**

1. Download Eclipse (Latest Version) -

i. Pull and Extract the Zip file

ii. Path recommendation: “C:\eclipse”

iii. Run Eclipse Oxygen by selecting the designated parent path, (eclipse.exe)

iii. Identify location for Eclipse

**7. Setting up the workspace for the bot with Maven**

1. Create a new project
2. Select Maven project
   1. Maven-archetype-quickstart 1.1
3. Give a group name and artifact Id then select finish
4. Once the project is created, open up Pom.xml in a text editor
5. Enter the following as a dependency

<dependency>

* 1. <groupId>org.javacord</groupId>
  2. <artifactId>javacord</artifactId>
  3. <version>3.0.5</version>
  4. <type>pom</type>

</dependency>

1. An alternative way to setup the project can be found [here](https://github.com/Javacord/Javacord)
   1. The project can be setup with either Maven or Gradle, but for the current implementation, Maven was shown.

**8. Configuring the bot’s token**

Every bot requires its own token, or secret, to run. This token should be hidden from other users as it can be used to take control of the bot and use different code bases with it. To access the bot token the following steps must be taken.

**For integrity purposes, the bot secret to Gaiza will not be included in this project.**

1. Navigate to [here](https://discordapp.com/developers/applications) on the discord platform.
2. Click on new application in the top right and create a bot with the credentials necessary
3. Once in here, you will need to navigate to the section labeled bot and add a bot user.
4. In the section labeled General Information, the client secret can be found. Reveal the secret and copy this somewhere for use later.
5. Now go to your project. Inside the main, to start up the bot, enter this into the main:
   1. DiscordApi **api** = **new** DiscordApiBuilder().setToken(token).login().join();
6. In place of token inside of “setToken(token)”, enter the secret for the bot. This will allow the bot to go online.
7. The next step is to run the project. This will turn the bot online and able to be used in a server.
   1. An executable jar file can be made in place of this. This will be detailed later in the manual

**9. Understanding implementation of Gaiza packages and class files**

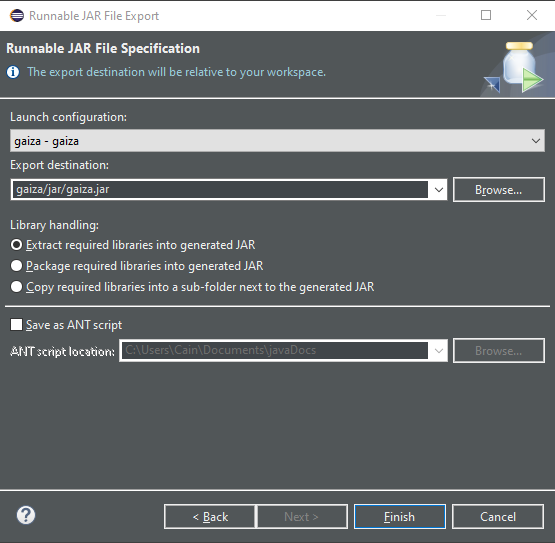
The packages detailing commands, admin commands, and listeners will be left out because they are outlined in the user manual.

* + Com.github.cainKaltenbaugh.gaiza
    - gaiza.java
      * This package is where the main is held. It starts the bot’s processes by acquiring the token from the specified location and starting up all the commands for the bot. Activity is also updated in the bot here and will allow for a message to be displayed underneath the bot’s name on the Discord platform. Initialization of the commands will output information to the console.
  + jsonDatabase
    - InitDatabase.java
      * Handles creation of the databases for every server the bot resides in. When the instructor is called, the server list is created and then checked to see if there are any new servers in the system. Server information will be loaded into a linked list. Any new servers will have a new blank initialization file created for them. If a new server joins while the bot is running, it will create a new file and initialize the file for it as well. If a server leaves this will also be tracked, but a server can join back with the original details of the file it had. Server changes are also documented here. So, if there needs to be a change to the database, then then it will check the files against the active linked list to see if any information has changed. If it is changed, it will save over the old files.
    - DatabaseLL.java
      * The linked list is created and managed within here. The linked list will hold various properties of each server within its scope and can be used by outside functions to change what is in it. The list can be statically allocated and has a way to be instantiated in outside getter methods that will let the code modify values. So if a server has a new prefix, there is a command that allows the server prefix to be changed outside of this package.
    - GlobalUserInformation.java
      * User information will be stored separately of server information. This is because each user can be in multiple servers with the bot currently in it. Here will be the file management system for the user accounts in each server. Currently, it is set up to create files based on each user’s ID and will be used to track information through each server in future implementations.
  + management
    - BotInfo.java
      * BotInfo manages information regarding the bot. Information included currently is what the bot name is, the image icon of the bot, the bot activity to be displayed, the amount of servers the bot is in, and the amount of users the bot has access to. These can all be called statically from outside of the package.
    - Keywords.java
      * Keywords refers to the prefixes the bot uses. To call commands, different prefixes need to be used depending on the server. Any time a new server joins it will default to the $ prefix. Each server’s admins can change this independently of other servers. The get method allows each server to get its own respective key and the set method allows each server to set its own respective key.
    - Token.java
      * This allows the bot token to be acquired from an outside location and returned to the caller. The token/secret is kept in a secure location so that anyone looking at the code cannot access it. This is recommended practice for every bot creator to protect it from outside modification.

**10. Exporting as an executable jar file**

To make the project executable from outside of Eclipse, the project needs to be exported as a runnable jar file.

1. Select the project
2. Click file > export and find the option under Java labeled “Runnable JAR file”.
3. After this, select the launch configuration to be the current project and select a destination to export the file to.



1. After finalizing these choices, click finish.
2. The project will be runnable from the newly created jar file. Any updates made to the project will need to have the jar file updated before the new changes will work.