

## Final Project Report

### - War Drill Recording and Management System –

## 1. Purpose

This project is based on the game, Total War: Warhammer II. The windows application will allow users to log in and manage their own troops, arms compositions, and war drill records.

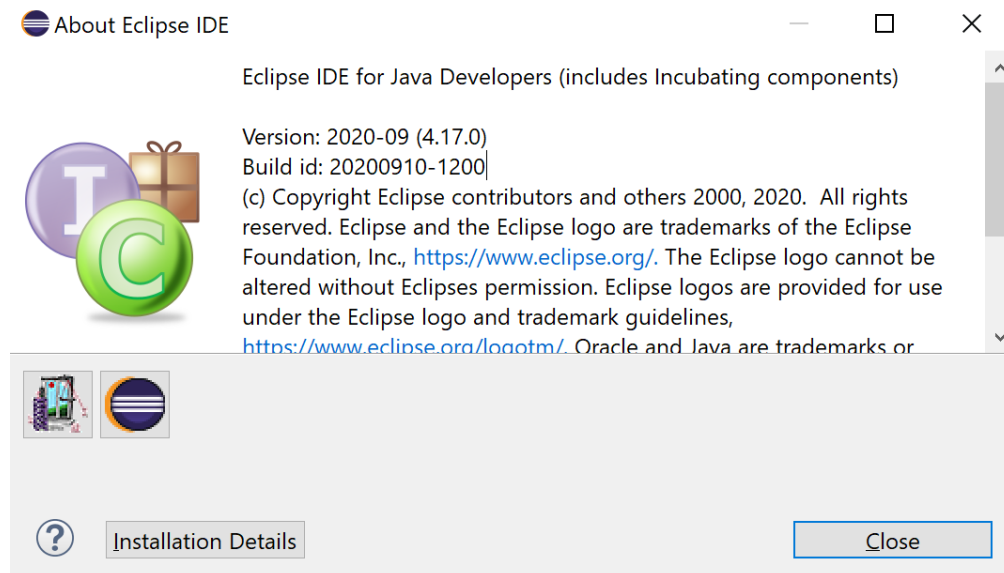
*Why does it interest us?* I was a great fan of Total War: Warhammer II, a war game based on a fantasy world. In this game, a player will play as a commander, exploring the continent, forming up his/her own troops, and selecting proper strategies to beat the enemies. However, since the core game mechanics are too complex, it is fairly unfriendly to beginners. For that reason, this application will be built for those players. They can record and manage the arms compositions they are about to try or have tried and will be able to find out the most suitable ones of their troops.

## 2. Technical Supports and Requirements

This application used MySQL for database design and Java for programming. There are some preparations need to be done before running this application.

### 2.1 Software

This application was developed by *Eclipse IDE for Java Developers*, and the version is *2020-09 (4.17.0)*.



## 2.2 Library

A standard JRE System Library will serve, and it should contain *java.sql*.

```
> java.se - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
> java.security.jgss - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
> java.security.sasl - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
> java.sql - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
> java.sql.rowset - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
> java.transaction.xa - C:\Program Files\Java\jdk-13.0.1\lib\jrt-fs.jar
```



## 2.3 Plugins

The front-end is based on *Swing* and *WindowBuilder*. Follow the links below:

<https://www.eclipse.org/windowbuilder/download.php>

Open the link and you will find a web page containing the *Update Sites* table below. Copy the link in the red rectangular and paste it into Eclipse *Install New Software* window (in the navbar, click help and then click install new software). If you need more detailed information or instructions on this installation, click the link in the red rectangular and you will get what you want.

### Update Sites

	Download and Install		
Version	Update Site	Zippped Update Site	Marketplace
Latest (1.9.4)	<a href="#">link</a>	<a href="#">link</a>	 Install
Last Good Build	<a href="#">link</a>	<a href="#">link</a>	 Install
1.9.4 (Permanent)	<a href="#">link</a>	<a href="#">link</a>	
1.9.3 (Permanent)	<a href="#">link</a>	<a href="#">link</a>	
1.9.2 (Permanent)	<a href="#">link</a>	<a href="#">link</a>	
1.9.1 (Permanent)	<a href="#">link</a>	<a href="#">link</a>	
1.9.0 (Permanent)	<a href="#">link</a>	<a href="#">link</a>	
Archives	<a href="#">link</a>		

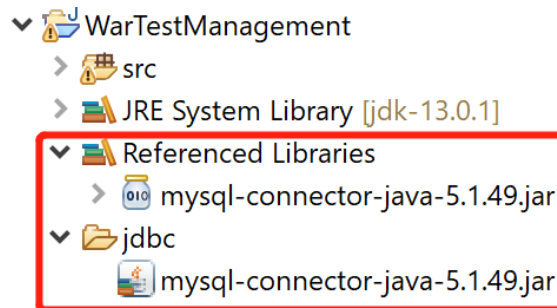
When Eclipse identify that installation link, it prompts you to select plugins. It is suggested to select all of them. When the installation finishes, restart Eclipse and check the installation details. Make sure you have those plugins in the red rectangular below installed.

> Mylyn WikiText	3.0.38.202008172112	org.eclipse.mylyn.wikitext.feature.feature.group	Eclipse Mylyn
> Mylyn WikiText Editors	3.0.38.202008172112	org.eclipse.mylyn.wikitext.editors.feature.feature.g...	Eclipse Mylyn
> Oomph Setup	1.18.0.v20200901-1012	org.eclipse.oomph.setup.feature.group	Eclipse Oomph Project
> POM Editor using LemMinX language server (includes Incubating compo	1.16.2.20200902-0711	org.eclipse.m2e.lemminx.feature.feature.group	Eclipse.org - m2e
> Swing Designer	1.9.4.202009181620	org.eclipse.wb.swing.feature.feature.group	Eclipse WindowBuilder
> Swing Designer Core Documentation	1.9.4.202009181620	org.eclipse.wb.swing.doc.user.feature.feature.group	Eclipse WindowBuilder
> SWT Designer	1.9.4.202009181620	org.eclipse.wb.rcp.feature.feature.group	Eclipse WindowBuilder
> SWT Designer Core	1.9.4.202009181620	org.eclipse.wb.swt.feature.feature.group	Eclipse WindowBuilder
> SWT Designer Core Documentation	1.9.4.202009181620	org.eclipse.wb.rcp.doc.user.feature.feature.group	Eclipse WindowBuilder
> SWT Designer SWT_AWT Support	1.9.4.202009181620	org.eclipse.wb.rcp.SWT_AWT_support.feature.group	Eclipse WindowBuilder
> SWT Designer SWT Support (requires Eclipse WTP/WST)	1.9.4.202009181620	org.eclipse.wb.swt.feature.feature.group	Eclipse WindowBuilder
> Tip of the Day UI Feature	0.2.1100.v20200724-0938	org.eclipse.tips.feature.feature.group	Eclipse.org
> Wild Web Developer XML tools	0.10.2.202007231627	org.eclipse.wildwebdeveloper.xml.feature.feature.g...	Eclipse Wild Web Developer
> WindowBuilder Core	1.9.4.202009181620	org.eclipse.wb.core.feature.feature.group	Eclipse WindowBuilder
> WindowBuilder Core Documentation	1.9.4.202009181620	org.eclipse.wb.doc.user.feature.feature.group	Eclipse WindowBuilder
> WindowBuilder Core UI	1.9.4.202009181620	org.eclipse.wb.core.ui.feature.feature.group	Eclipse WindowBuilder
> WindowBuilder GroupLayout Support	1.9.4.202009181620	org.eclipse.wb.layout.group.feature.feature.group	Eclipse WindowBuilder
> WindowBuilder Java Core	1.9.4.202009181620	org.eclipse.wb.core.java.feature.feature.group	Eclipse WindowBuilder
> WindowBuilder XML Core (requires Eclipse WTP/WST)	1.9.4.202009181620	org.eclipse.wb.core.xml.feature.feature.group	Eclipse WindowBuilder

Editors and tools for working with HTML, CSS, XHTML, in combination, and more.

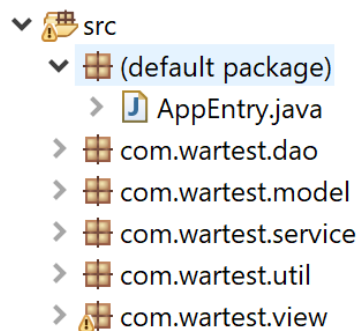
## 2.4 JDBC

Create a folder called jdbc and add mysql-connector-java into it. Remember to add path.



## 2.5 Run Application

Create a Java Project. Copy those packages and .java files into the src folder. Open the file com/wartest/util/DbUtil.java and **change the username and the password** that serve your own database. There is only one java file in the default package: AppEntry.java, which is the only entrance of the whole project. Run this file and the application starts.

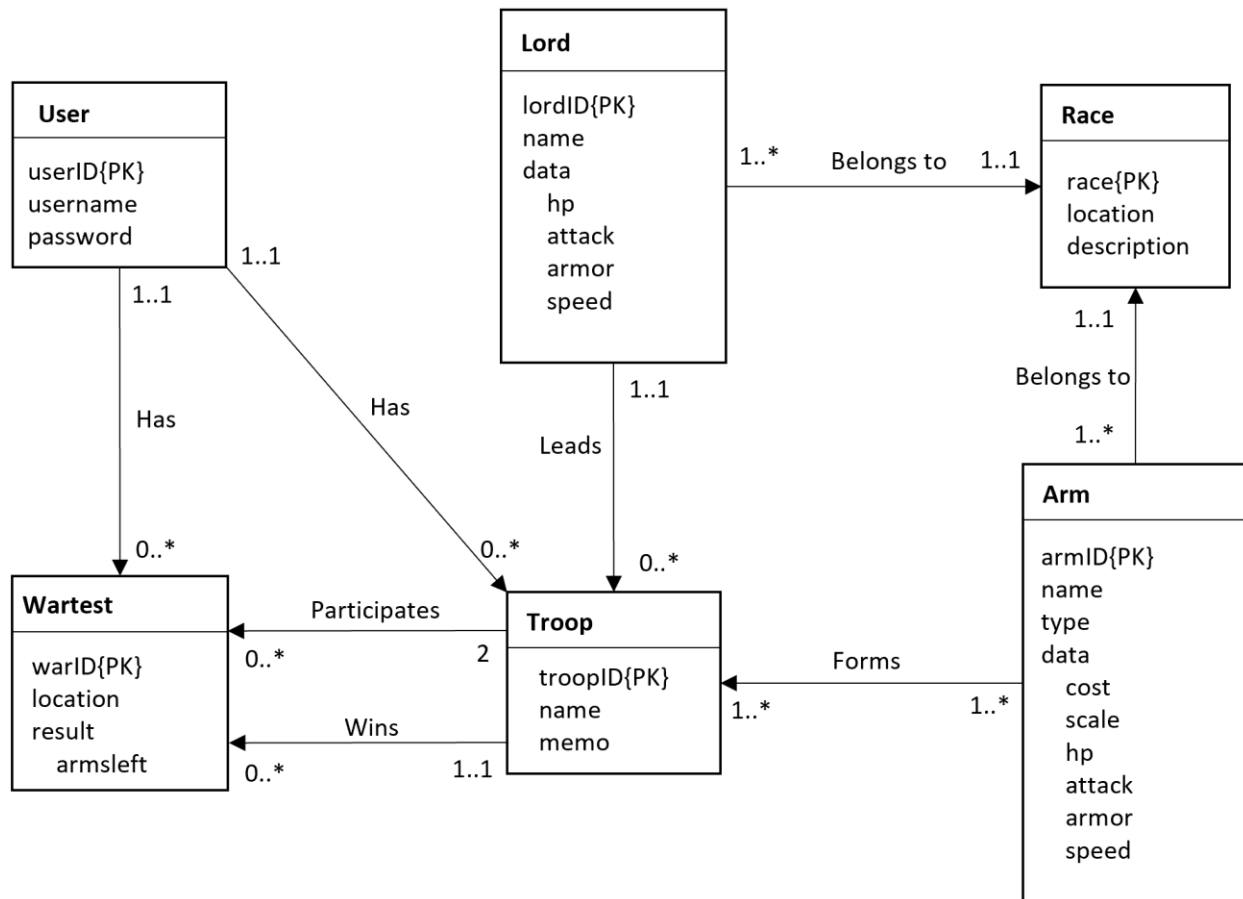


When you can see the Log In form below, it means the application works well so far.

The screenshot shows a window titled 'User Login'. Inside, the title 'Total War II War Drill System' is displayed. Below it, there are two input fields: 'Username' and 'Password'. At the bottom, there are three buttons: 'Log In', 'Sign Up', and 'Reset'.

### 3. Database Design

The database UML diagram is shown below.

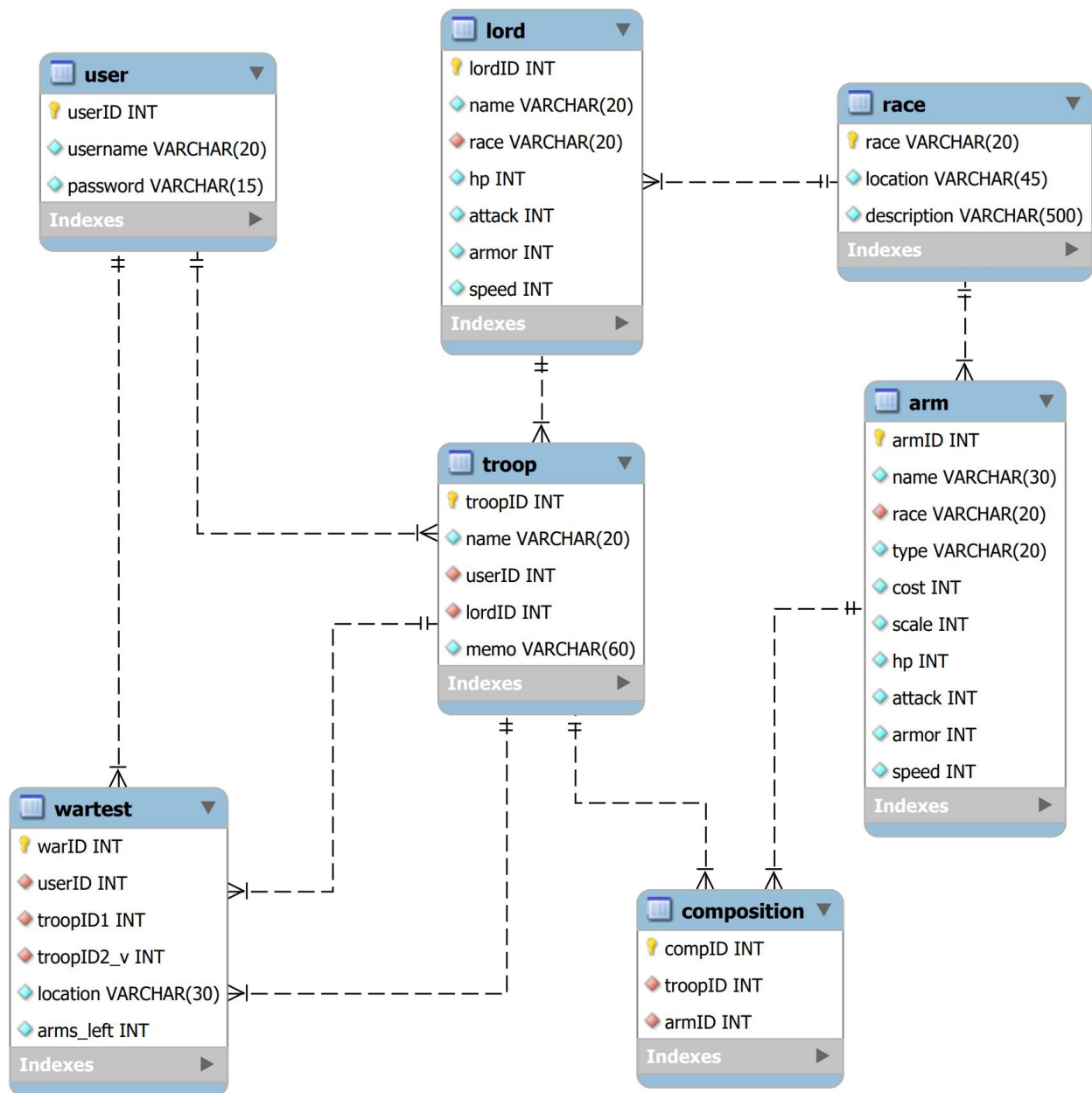


The UML Diagram

The database for this application contains 6 main entities.

- **Users.** One user will have views only on his/her own troops and war-tests. Each user has a user ID, a username, and a password.
- **Troops.** Users can form up their own troops. A troop has a lord and at least one arm. Normally, the lord and the arms in one troop belong to one race. Each troop has a troop ID, a troop name (another unique attribute), a troop memo used to remind its owner the usage or composition of this troop, and a user ID indicating which user this troop belongs to.
- **Lords.** According to the rules of Warhammer II, a troop must have a lord—the commander. Each lord has a lord ID, a race it belongs to, and a set of battle data.
- **Arms.** A phalanx. They are the main components of a troop. Each arm has an arm ID, a race it belongs to, a type indicating its usage, and a set of battle data.
- **Races.** There are several races in the fantasy world of Warhammer II. Every single lord or phalanx belongs to exactly one race. Each race has a race ID, the name of it, and the original location.
- **Wartests.** The record of war drills. A tuple of war-test will record which two troops were engaged, where was the combat, and what the result was. Each tuple of war-test has a war-test ID, two *different* engaged troops, the location, the victor troop, how many arms it left, and a user ID indicating which user this troop belongs to.

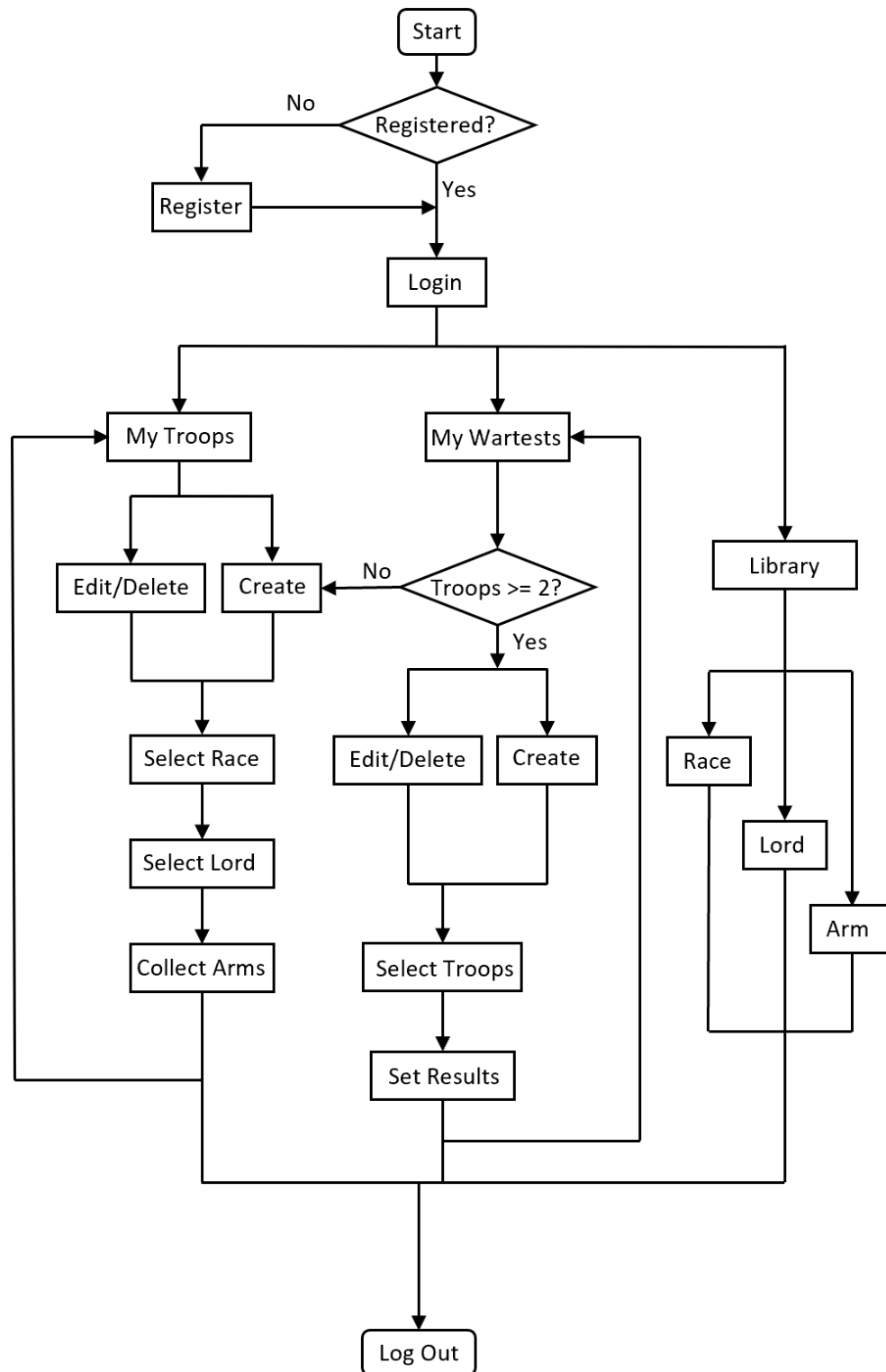
The ER diagram generated by MySQL workbench is shown below. The table *composition* is the join table. It is used to implement the *many-to-many relationship* between troop and arm.



The ER Diagram Generated by MySQL Workbench

## 4. User Flow

The flowchart is shown below.



Below are step-by-step operations a user may execute.

- Log In. If a user has no account, then register for an account. This application requires a mandatory log-in. Once a user logs in, he/she has the access to the library, which contains detailed information on races, lords, and arms.
- The main interface “War Drill Home” contains two menus on the top-left corner: main menus and library.
- In the library, a user can read all the detailed information on races, lords, and arms.
- Inside the main menu, there are two sub-menus: My Troops and My Wartest. They contain the user information on troops formed and war-tests they recorded, and CRUD operations on those data. Click the third item Log Out to log out.
- My Troops, the main interface for this application. It contains two items: Add Troop and Manage Troop. Click Manage Troop, then a user may see the troops that have been formed. One troop row shows simplified troop and lord information.
- Clicking one of the troop records on the table will lead to detailed information shown below the table—as a troop edit interface—where the user can modify this troop by changing the race, the lord, or the arms. After choosing race, click filter and the lord and arm scroll-down menu will be filled with lords and arms belong to the chosen race, respectively. Click DELETE at the bottom of this window to delete this record.
- If the user has no troop, go back and click Add Troop and also enter another edit interface. This interface contains necessary information on the lord and arms of the troop. The user may build their new troops here.
- When adding arms into the troop, choose one arm on the scroll-down menu, click ADD ARM and the selected arm will show up on the table below the scroll-down menus. The user may add the same arm multiple times. There are two small buttons on the right of this table: CLEAR to clear all selected arms; DELETE to delete the arms that the mouse selects.
- When completing the modification/creation, click SUBMIT(for both editing and adding). The information will be updated.
- My Wartest contains two items, too: Add Wartest and Manage Wartest. The operations and functionalities are similar to My Troop, except that all the data in this interface are base on the user’s own troop. It should be pointed out that only when a user has at least two troops can he/she create or edit the wartest results. Empty troops or two identical engaged troops won’t lead to further actions.
- For both editing and adding operation on wartest, after choosing two engaged troops, click CONFIRM and the victor scroll-down menu will be filled with these two troops. Select the victor and how many arms the victor left. Click SUBMIT to update information. On the management window, click DELETE to delete the selected records.

## 5. Lessons Learned

### 5.1 Technical Expertise

The knowledge of using Java to connect with the relational database has been consolidated. After this project, the basic connection works like getting connection, closing connection, and catching and handling relevant exceptions can be implemented cleaner, faster, and smoother. The high volume of practice on manipulating data in the database through application code has improved the SQL writing skills.

For database design, this is also a wonderful practice. The abilities to get aware of a real-life issue, transform the abstract description into a database conceptual design, and further transform it into a logical design by using MySQL DDL have all been strengthened.

### 5.2 Insights

1) Practice makes perfect. One of the core requirements of gaining a skill is a large amount of practice, during which one will be able to get used to the specific working flow, accumulate experience with handling errors, strengthen the ability of accessing information, and consolidate the relevant concepts that support the work.

2) Keep your brain sharp. An inactive brain is very likely to seriously slow down the work progress, and even greatly increase the code error rate. Therefore, if you need to take a noon nap, take one; if you need to work out, work out. Relaxing your brain is not a waste of time; letting a bad-status brain ruin your work, however, is.

3) Constantly optimize. A runnable application is never enough. It should be fairly elegant and readable, respond to the user's manipulations quickly, and has a considerable fault tolerance. There are some operations from the user side that a developer might never have thought about. Therefore, let your friends try your application and be humble to attend to their suggestions or complaints.

4) Do necessary redistribution. For example, when working on the front-end, the logic implements of all kinds of buttons, tables, and scroll-down menus could be quite cumbersome and tedious. If all of the business codes locate in the frontend, it will significantly reduce the readability and increase the difficulty of debugging. In this case, redistributing the business codes into an independent service layer is a very good choice.

### 5.3 Alternative Approaches

Since the user login is mandatory in this application—any user can only see their *own* troops and war-tests—tracking the current user (or current user ID) is significant. Unlike the Passport.js of JavaScript or the authentication & authorization methods in React.js, the identification in this project is very primitive. At the very first, the plan was to create a super-global integer variable to track the current user ID passed by the login session. When saying super-global, it means every part of the project would have the access permission to this variable. But unfortunately, the efforts all failed. And then, the second instinct was to create a temporary table in the database to track the current user ID, because no variable in the application side is more global than the data in the database. But there was an obvious flaw: when there are multiple users logging into the system, it could be a total mess at the database side. So, the second plan was abandoned, too.

Here comes a more manual but meanwhile simpler and more secure approach—passing the current user getting from the login session to every other session that needs it. Indeed, there are several more parameters that need to pass, but the fact is it can do the job well and reduce the load at the database side. This approach is effective and efficient.

### 5.4 Bugs and Flaws

All the desired functionalities and logics work well, and all the bugs and logic flaws were wiped out, according to the tests.



## 6. Future Work

### 6.1 Potential Usage

This project can be aimed at novice players of Warhammer II. Most of them are unfamiliar with game mechanics. They are not familiar with the combination of arms of various races and the advantages and disadvantages of lords, not to mention adjusting their troops according to different battlefield environments. This application can help novice players to record battles independent of the game plot and freely try the results of various forces fighting each other. Through these results, players will be able to become more familiar with the content of the game and explore a set of combat styles that suit them.

### 6.2 Future Improvements

- 1) The information of races, lords, arms, troops, and war-tests are all plain text and numerical data. There's no image or links illustrated to the users. In the future, those rich and colorful elements can be added both to the database and the front-end.
- 2) There is a *Library* section that contains the completed information on all the races, lords, and arms. However, in the *My Troop* and *My Wartest* sections, due to the limited space, the details and numerical data of lords and arms are hidden. In the future, there can be a link for each lord and arm selection that leads users to the library to get the full information and help them collect what they want the most.
- 3) This project is implemented as a desktop application, which has many limitations such as the user have to download the whole project and have to get a Java Running Environment together with all of those plugins, and the plain-text front end can be pretty boring. In the future, it is planned to transform this project into a web application using Spring Boot and React. The user interfaces will be more interesting, and the operations will be more convenient.

# Appendix

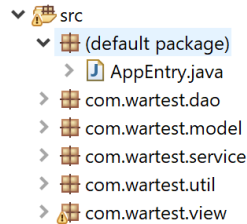
## 1. Data Source

The names of the races, the lords, and the arms are all real names from the game. The website is provided below:

[https://totalwarwarhammer.gamepedia.com/Total\\_War:\\_Warhammer\\_Wiki](https://totalwarwarhammer.gamepedia.com/Total_War:_Warhammer_Wiki)

Since the battle data is simplified in this application, the data, like hp, attack, scale and so on, is made-up.

## 2. Source Files Explanation



1) AppEntry.java, the only entrance of the whole project.

2) util. It contains two files. DbUtil.java is used for the connection works; StringUtil.java is used to integrate the methods related to String operations.

3) model. It contains all the objects corresponding to the entities of the database. Those objects are used for simplifying the process of front-back-database interactions. Passing objects among files are simpler and neater than passing scattered information.

4) dao. The data accessing layer. The files in this package are used for direct interactions with the database. For example, TroopDao.java contains all PreparedStatements and their execution methods for Troop entity.

5) service. The business logic (service) layer. The files in this package are used to implement all the business logic related to the user operations, like click SUBMIT button to submit an update, or generate all troops that formed by the user who currently logged in on the front-end windows.

6) view. The front-end layer. It contains all JFrames and JInterFrames. Those files form up the whole UI. They implement all the dynamic logic and user operation responses by calling methods from the service layer.