Hnefatafl is a family of 2-player Viking board games, where one player tries to guide a King to a refugee square, while attackers try to prevent this. Right now, we're implementing the popular 11x11 variant.

To develop and test this game, please follow the steps below:

1. **Configure Development Environment**
2. Download and install the **JDK 1.8.0\_221**

A screenshot of a cell phone

Description automatically generated

1. Download and install the database **MYSQL 8.0.18**

A screenshot of a cell phone

Description automatically generated

1. Download and install the **Navicat Premium 12.1.27**

A screenshot of a cell phone

Description automatically generated

1. Download and install the **IntelliJ IDE 2019.2.1**

A screenshot of a cell phone

Description automatically generated

1. Download the apache-tomcat-9.0.24 package

A screenshot of a computer screen

Description automatically generated

1. **Download the source code from Github**

<https://github.com/JacindaQiong/cs414-fa19-001-Party-A/tree/master/PartyA>

Clone our project from Github:

*$ git init*

*$ git remote add origin https://github.com/JacindaQiong/cs414-fa19-001-Party-A*

*$ git clone https://github.com/JacindaQiong/cs414-fa19-001-Party-A*

A screenshot of a cell phone

Description automatically generated

1. **Configure and Deploy the project**
2. **Import the project** that we just cloned from gitHub into IDEA:

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. **Change database connection information(db.properties)**

A screenshot of a cell phone

Description automatically generated

1. Configure the project:

Select the JDK we just installed:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Configure the tomcat server:

Choose the tomcat we just downloaded:

A screenshot of a computer screen

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. **Test the whole process**

**STEP 1: Download the JDK**

The first thing we need to do is prepare our PC to develop the game using **Java**.  
Install a software package called the **Java Development Kit** (**JDK**), which allows us to develop in

Java.

**STEP 2: Set Up a Development Environment**If you downloaded the JDK with the NetBeans IDE, start NetBeans, and begin programming.

You can also program using a simple text editor, and compile and run from the command line. Many text editors now come with the ability to run and compile Java files, but you may need to tell the program where javac.exe and java.exe reside on your computer. Once, your IDE or text editor is set up, you can begin programming.

**STEP 3*:* Application**

1. Use IntelliJ IDEA software to clone code from Github https://github.com/JacindaQiong/cs414- fa19-001-Party-A Project name: PartyA Database name: SQL
2. Opening the Navicat of MySQL, add the game\_invitaion.sql, game\_user.sql and game\_match.sql into that application

3.Compile the login.java application with the java command and run it with the java command. To run the tests:

1. Eclipse provides a couple of ways to run individual test methods, one from within the editor itself and another from the JUnit view.
2. For example, this is a simple code to run the testcases: **publicclass**SomeTest

{ @Test

**publicvoid**testMethod1() {...} @Test  
**publicvoid**testMethod2() {...} } 4.Add a testing logic (3 A’s)

page1image44746624page1image44751424

**Arrange:** consists of a few lines of code that are used to declare and initialize the objects we need in our test.  
**Act:** is usually a few lines of code where we perform the actions, whether it is some calculation or modify the state of our objects.

**Assert:** usually consists of a single line of code where we verify that the outcome of the **Act** part was made successfully.

Testing improves the quality of the code and it makes the development process more Agile.