

# Jungle

---

COMP3211 Software Engineering Group Project "Jungle" - 2022 Fall

Implemented by Group11

Contributors: Yuxing PEI, Suizhi MA, Jiaming HAN, Yunfei LIU

GitHub: [Jungle](#) - Link to the upstream

## Description

---

Jungle / Dou Shou Qi (斗兽棋), is a modern Chinese board game with an obscure history. This project is a console/command line based implementation of Jungle with exquisite GUI. For more knowledge about Jungle, please visit [Jungle \(board game\)](#).

This document serves as a manual, describing details about the followings.

- As a player
  - How to START / PLAY this game
  - How to INSTALL this game
  - How to UPDATE previously installed game
- As a developer
  - How to VIEW / EDIT the source code
  - How to COMPILE / BUILD the project
  - How to TEST the source code
  - How to start the game in DEBUG mode
  - How to IMPROVE our game
- How to interpret common ERROR messages
- And most importantly, how to provide COMMENTS to the other developer

## Prerequisites

---

### JDK 17

Java 17 LTS is the latest long-term support release for the Java SE platform.

For this project, you must fulfill this requirement.

Otherwise, you will get error messages at starting like this:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError: group11/comp3211/JungleAp
plication has been compiled by a more recent version of the Java Runtime (class file version
61.0), this version of the Java Runtime only recognizes class file versions up to 52.0
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:760)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:455)
    at java.net.URLClassLoader.access$100(URLClassLoader.java:73)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:367)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:495)
```

If you are seeing this error message, please upgrade your jdk version to at least 17.

## \*nix Platform

In this game, we have adopted conventions defined in \*nix / POSIX standard.

If you can only use Windows, there are two alternative options for you.

1. Set up a Linux environment using virtual machine (WSL, VirtualBox, VMWare...)
2. Use Docker (will be described in details later)

We strongly recommend you to use \*nix system, because the compatability issues may not be considered as complete as possible. Hence, for Windows users, option 1 will work in a better manner.

## Terminal

This game is terminal based. You must have at least one terminal application / simulator to start playing.

Minimum requirement: 30 rows \* 100 columns.

If your terminal does not satisfy this requirement, there are two options solving this problem.

1. Resize the terminal window
2. Resize the font size in terminal

For macOS, the default [Terminal.app](#) works fine.

For Ubuntu, the default [Terminal.app](#) works fine.

## For developers

---

# IDE

You may view and edit the source code in any IDE, the developers use IntelliJ for its strong compatibility for Maven and package / plugins importing. But you may NOT start the application by clicking the 'start' button in IDEs, that won't work, and you will see our kindly reminder "Please read [README](#) carefully".

## Shell script

In the project directory, you may find a shell script [jungle](#). Most developer options are performed by this script. (This is probably one main reason for only supporting \*nix platforms)

### Usage (goto the project base directory)

```
./jungle argument?
```

while argument is an option in the followings.

#### 'help'

```
./jungle help  
# display README.md
```

```
./jungle help pdf  
# display README.pdf
```

```
./jungle help page  
# display README in web page (using default browser)
```

#### 'version'

```
./jungle version  
# display the current project version
```

#### 'build'

```
./jungle build  
# build and run the project (build and then start the game)
```

The project will start building process using Apache Maven, if you don't have maven installed on your local machine, we will automatically use our provided [maven](#) for building.

A reminder message will show up if this is the case. Noted if this happens, you may need to wait for 1-2 minutes, because maven will fetch some plugins from maven central, and this process usually depends on

you network

connection. So be patient if you don't have maven preinstalled in you local machine. After the first build, this process will be much faster.

Note that this command will clean all the previously built targets, so technically this is 'rebuild and run'

## 'test'

```
./jungle test  
# build and start unit tests in the whole project
```

Note that this command will not start the game, just finish the unit tests, the result will be displayed.

## 'clean'

```
./jungle clean  
# clean all generated targets, including classes, jars, basically anything in target/  
directory
```

## 'install' && 'uninstall'

```
sudo ./jungle install  
# install ./jungle to /usr/local/bin and set $JUNGLE_HOME
```

```
sudo ./jungle uninstall  
# remove /usr/local/bin/jungle
```

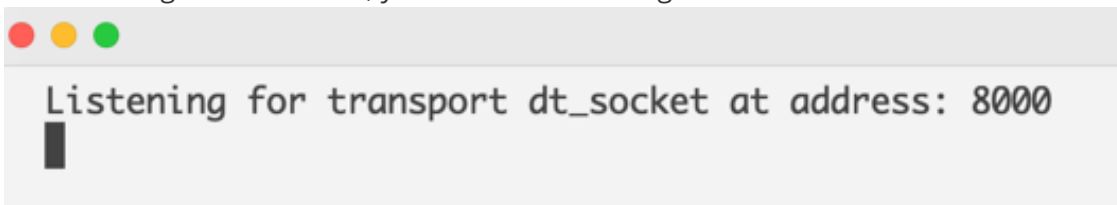
Note these two commands are NO longer recommended in current version. They require root, and are implemented by naive undergraduates instead of formal package managers. Their functions, are replaced by popular package managers like Homebrew and apt-get...

If you are interested in installing Jungle, we have cleaner methods for you.

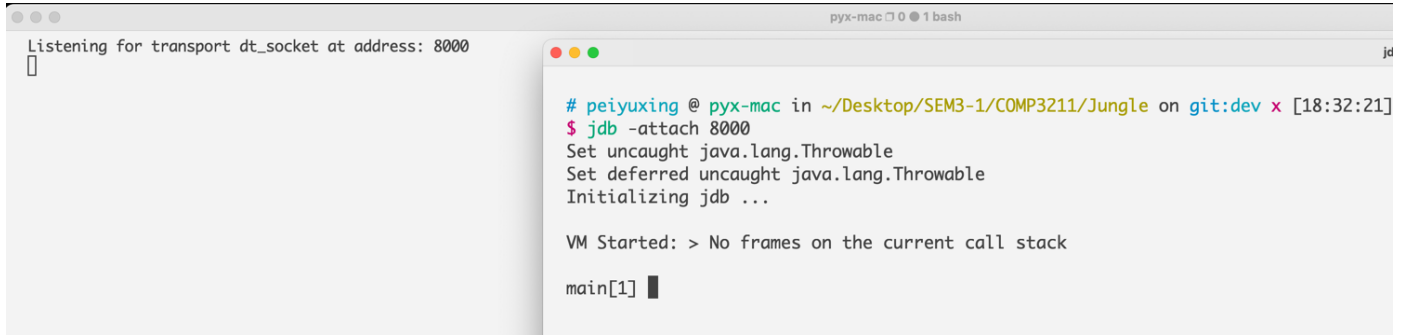
## 'debug'

```
./jungle debug  
# start the project in debug mode
```

After running this command, you will see something like this:



Then, you should open another terminal window to start [jdb](#)



The image shows two terminal windows. The left window is titled 'pyx-mac' and shows 'Listening for transport dt\_socket at address: 8000'. The right window is titled 'pyx-mac' and shows the command 'jdb -attach 8000' being executed, followed by several status messages and the prompt 'main[1]'.

```
Listening for transport dt_socket at address: 8000
# peiyuxing @ pyx-mac in ~/Desktop/SEM3-1/COMP3211/Jungle on git:dev x [18:32:21]
$ jdb -attach 8000
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
Initializing jdb ...

VM Started: > No frames on the current call stack

main[1]
```

Seeing something like that means you have successfully attached Jungle to jdb, and you may start debugging using jdb commands.

## No argument

```
./jungle
# start the game directly
```

Before running this command, you must have built the project for at least ONE times. Check if target/ directory exists. If not, you must build the project first.

## Use jungle-app-maker

In the project directory, you may find a shell script [jungle-app maker](#).

This is a "magic" shell script that can combine the executable jar along with startup environment settings together into a single executable file.

## Usage

```
cat jungle-app-maker target/Jungle-1.0-SNAPSHOT.jar > jungle-app && chmod +x jungle-app
# generate jungle-app
```

After doing this, you will find an executable file [jungle-app](#).

```
./jungle-app
# Start the game!
```

The jungle-app-maker makes developers much easier for publishing their own software!

# External Libraries and Tools

## Maven

[Maven](#)'s primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern.

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices

Maven is used in the phase of implementation and building, for the sake of maximizing compatibility and making the project lifecycle more comprehensive.

## Lombok

Project [Lombok](#) is a java library that automatically plugs into your editor and build tools, spicing up your java.

Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.

Lombok is used in the phase of coding only, for the sake of reducing redundant code (Getters and Setters, equals and hashCode). And it is customizable.

## Docker

[Docker](#) takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development – desktop and cloud. Docker's comprehensive end-to-end platform includes UIs, CLIs, APIs and security that are engineered to work together across the entire application delivery lifecycle.

In this project, docker is mainly used for solving compatibility for Windows users.

## For Players

---

The players are free to follow the above steps, but we have provided more user-friendly methods for those who don't want to deal with compiling and building.

# Homebrew

Homebrew is available on macOS and Linux (x86).

## Preparation

[what is Homebrew](#)

If you don't have Homebrew installed before, paste the command in your terminal.

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

## Installation

Paste the commands in the terminal.

The game will be automatically installed.

```
brew tap xyliax/homebrew-tap
brew install xyliax/tap/jungle
```

To start the game, use this command.

```
jungle-app
```

To update the game, use this command.

```
brew reinstall xyliax/tap/jungle
```

This command will reinstall jungle-app from GitHub repository.

## Get the executable

We have provided releases on GitHub.

## Release Binaries

[jungle-app](#)

```
wget https://github.com/xyliax/Jungle/releases/download/test-developers/jungle-app
chmod +x jungle-app
```

And you are free to start the game.

```
./jungle-app
```

# Docker

This is one possible solution for Windows users who have started Docker service.

Using docker, players can start Jungle at ANY platforms!

On Windows, open PowerShell to use docker commands.

## Preparation

[Install Docker](#)

## Fetch the compressed Jungle image

[jungle-image.tar.gz](#)

## Load Jungle image

Load the image using the following command. (For Mac and Linux)

```
docker load < jungle-image.tar.gz
```

For windows, please use the following commands instead.

```
docker load -i jungle-image.tar.gz
```

You will see "Loaded image: jungle-docker:TEST".

## Start the game

```
docker run -it jungle-docker:TEST /bin/bash
```

You will see a bash prompt, which implies that you have started the interactive shell.

Simply use the following command to start the game.

```
./jungle-app
```