Risk Client server programmer’s Manual

by Ryan Silver

[Installation](#_f5yefxdcrnqo)

[Database Setup](#_gabmjakuhbdl)

[File setup](#_pfdjswe6zdka)

[Program Classes and Methods Description](#_ob6wncnwkhrf)

[Class Comm](#_oq8msvig5yrs)

[Method Summary](#_ppyxzyvrqev6)

[Class CubbyHole](#_tjh1juaf50ad)

[Method Summary](#_q73rj13iajtc)

[Class DBConn](#_7pfrmy9s06vn)

[Constructor summary](#_8d2oqap9rdy4)

[Method summary](#_vwmwphp4ez4d)

[Class Game](#_81lignoezjy2)

[Constructor Summary](#_b8hl25g9il7c)

[Method Summary](#_crfnyzsscldr)

[Class playerNode](#_p9tx53vv4ega)

[Constructor Summary](#_2k0nyiqu6qsu)

[Method Summary](#_p4krjr7ew0qb)

[Class playerRAN](#_6h61gv7ewcfr)

[Constructor Summary](#_2rcs0en7u2bt)

[Method Summary](#_fszpgkhpuax4)

[Class playerWeb](#_bl5svx5raijs)

[Constructor Summary](#_fjsj7yqm35xc)

[Method summary](#_tox97adfdfw2)

[Class RDB](#_kaec7df6bk8z)

[Constructor Summary](#_vq5zw3qlbtbn)

[Method Summary](#_23wk5p1c6ta)

[Class riskServer](#_781f3dcnj2ul)

[Constructor Summary](#_ukhtuiyt3su3)

[Class tRiskR](#_ckynxtxl1seg)

[Constructor Summary](#_o46j2rswm4d0)

[Method summary](#_o6s4onwrc2kp)

[Class U](#_4i30k0gs2sgl)

[Method Summary](#_pjea4jon0upr)

[delay(int secs)](#_tnvozit14ytu)

[adds delay for slow play.](#_5znklumu8meh)

# Installation

## Database Setup

The database should be contained in a file called games.db. You can set up exactly what you want to name this file with another file called gameDbLocation.CFG where you can edit paths to the file.

The database should contain a total of 3 tables schemas are as follows:

CREATE TABLE `user` (

`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

`first` VARCHAR(255) NOT NULL,

`last` VARCHAR(255) NOT NULL,

`username` VARCHAR(255) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`email` VARCHAR(255) NOT NULL,

`usertype` INTEGER NOT NULL DEFAULT 0

);

CREATE TABLE ‘games\_history’(

‘game\_number’ int(11) NOT NULL,

‘update\_number’ int(11) NOT NULL,

‘update\_contents’ varchar(300) NOT NULL

);

CREATE TABLE ‘games\_tracker’(

‘user\_id’ int(11) NOT NULL,

‘game\_number’ integer primary key AUTOINCREMENT not null

);

## File setup

To install the software simply move the CSGServer.jar file to any folder. This folder must contain 2 separate files as well. The filenames should be gameDbLocation.CFG if the game is located externally or named differently the path to that database file should be in gameDbLocation.CFG as the first line there should be no extra lines in this file. Lastly start CSGServer.jar and a file will be generated. Called either ras or rbf.bat depending on the running operating system. Give this file executable permissions if you are on linux.

After file setup is complete simply run the CSGServer.jar file and the server will be running

Port forwarding

The Server uses ports 5551-5555 be sure to forward all of these ports tcp and udp

# Program Classes and Methods Description

(more information can be found on all code in CSGServer/dist/javadoc/index.html)

## Class Comm

unused class used to implement external code through file manipulation

### Method Summary

recvMsg(java.lang.String fileName)

waits for a file to appear and reads it

sendMsg(java.lang.String fileName, java.lang.String msg)

used to send a message to another program or thread via a file

## Class CubbyHole

Thread safe communication between threads

### Method Summary

get()

used to get messages from other threads

put(java.lang.String value)

Used to send messages to other threads

## Class DBConn

Used to set up connections to the database

### Constructor summary

DBConn()

reads from a file the name of the database and connects to it.

### Method summary

closeAll()

closes the connection to the database.

createNewGame(java.lang.String username)

sets up a game for the user and associates their username with games\_tracker using games\_tracker it is possible to tell what game belongs to who

main(java.lang.String[] args)

testing method only to verify function of database

updateHistory(java.lang.String map\_update)

records actions of the players into the database

## Class Game

class used for game features like attacking and setting up playernodes

### Constructor Summary

Game(boolean d)

sets up a game

### Method Summary

addArmy(playerNode mod, playerNode inform, boolean init)

used during game.

addBonusArmies(playerNode AttTerr, playerNode DefTerr)

adds bonus armies according to game rules

attack(playerNode AttTerr, playerNode DefTerr)

communicates with seperate threads for attacks

countP0Armies()

counts armies belonging to P0

countP0Territories()

counts territories belonging to P0

countP1Armies()

counts armies belonging to P1

countP1Territories()

counts territories belonging to P1

getGameNumber()

UNUSED

getP0()

returns player0

getP1()

returns player2

initializeTerritories()

resets the territory nodes and starts a game of risk allowing players to place armies

initPlayer(java.lang.String playerType, java.lang.String playerFileName, java.lang.String color)

initialize a random player

initPlayerWeb(java.lang.String playerType, java.lang.String playerFileName, java.lang.String color, CubbyHole recv,CubbyHole send)

initialize a web based player version of initPlayer to use with web application

message(java.lang.String s)

prints a message S to console

movement(playerNode AttTerr, playerNode DefTerr)

currently does not work.

report()

p1 and p0 output strings and status

sendTransGetString(playerNode p, java.lang.String s)

sends a transaction and gets a string back from player

setSpeedSlow()

slows down the game to a human comprehensible level

## Class playerNode

### Constructor Summary

playerNode(boolean d, java.lang.String lang, java.lang.String b)

Used to setup a normal player

playerNode(boolean d, java.lang.String lang, java.lang.String b, CubbyHole to, CubbyHole from)

Used to setup a web player

### Method Summary

getColor()

Gets color of player

receive()

Gets a string reply from player

send(java.lang.String s)

Sends a string to the player

setColor(java.lang.String s)

Sets player color to S

toString()

Returns attributes of player as string

## Class playerRAN

Class used to play random moves in a game of risk

### Constructor Summary

playerRAN(CubbyHole a, CubbyHole b)

Sets up a random player

### Method Summary

run()

When run is ran the player begins to respond to commands through the cubbyholes

printAL(java.util.ArrayList<java.lang.String> t)

## Class playerWeb

variant of player A that uses java sockets to communicate with a player over the web

### Constructor Summary

playerWeb(CubbyHole a, CubbyHole b, java.net.Socket c)

sets up a cubby hole that can be used to communicate over the web

### Method summary

run()

When run is ran the player begins to respond to commands through the cubbyholes

## Class RDB

keeps track of territories

### Constructor Summary

RDB()

used to set up and add all continents to board

### Method Summary

countArmies(java.lang.String cc)

Counts armies of color cc

countTerritories(java.lang.String color)

Counts Territories of color color

getTerritoryNode(java.lang.String s)

gets territory node with shortname S

resetTn()

sets count to 0 and color to grey for all nodes

territoriesFull()

used to determine if the territories are full or not

## Class riskServer

### Constructor Summary

riskServer(java.lang.String[] args, int port)

sets timeout and ports for sockets

## Class tRiskR

the next level up from game.java: uses the game class to actually setup and play games of risk

### Constructor Summary

tRiskR(int gameNumber)

First time setup for a new game

### Method summary

runGame(java.lang.String[] args, CubbyHole rec, CubbyHole send)

Starts a new game of risk

## Class U

used to throw exceptions and errors for disconnecting, and misbehaved players

### Method Summary

### delay(int secs)

### adds delay for slow play.

pr(java.lang.String s)

print something to screen

rAssert(boolean b, java.lang.String err)

used to throw exceptions and errors for disconnecting, and misbehaved players

### 

## 

### 

### 

## 

### 

## 