**Concurrent Programming**

**Assignment SP2 2017**

**Written By**

**Duoqi Zheng**

# Introduction

This document defines the basic information taken to design the implementation of the multiplayer game. Below, you can find the constraints and game rules which were defined and implemented in the snake game.

# Constraints

1. The game server will hold and update the game state.
2. Player only sends actions to the server
3. Only one server exists.
4. Player should be able to login to the server.
5. Game is called multiplayer snakes
6. Move are up, down, left and right.
7. Moves will be handled by the server then the game board will be updated.
8. Once fruit has been eaten, the player scores will be updated, and a new fruit will be placed
9. The fruit will be placed at a random point on the map
10. No Networking
11. Players and Server to be threads on the Java Virtual Machine(JVM)
12. Server must be able to handle actions from one human controlled player
13. Server must be able to simulate four AI players
14. AI players make random moves on the map

# Game rules

1. Walls cannot be passed through.
2. Players cannot pass through own self.
3. Players, including AI players will die upon they have collision with wall, themselves or another player.
4. Players, including AI players will continues in last direction if no inputs are passed to the server.
5. The Human controlled player has a scoreboard which records his movements.
6. Once a player makes a collision with the wall or another player the player is removed from the current game.
7. Snakes will grow to an infinite size.
8. Snakes length will be increased by eating foods.
9. Fruit will be placed on board for players to eat and gain points after a certain random time.
10. Fruit will last on screen until eaten
11. **Only one player (AI player) can eat the fruit.**
12. The timer will stay at constant speed.

# Classes

**Package: Client**

Client

The Client class get the inputs and push it to the server

**Package: Server**

Server

The server class run 4 AI players runnable and listen requests from client. It runs a new human player runnable when a new client connects with it.

PlayerHuman(Runnable)

The playerHuman handles login request from the client and has a thread which runs on it. That thread handles instructions from client, updates the snake on gameboard.

PlayerAI

The AI player does random move and it won’t interact with the client

BlockingQueue

The blockingqueue class is a fixed capacity queue. It contains all the redraw works need to be done. It surrounded with guard while condition so that it won’t have race condition.

ServerThread (consumer)

The server thread class plays its role as a worker. It takes a redraw work from blocking queue, and them execute it by using an executor.

Snake

The Snake class holds all the credentials of a snake, including the snake direction, the head position etc.

**Package: Tools**

GameBoard(producer)

The GameBoard class holds a 100x100 nodes gameboard which has 1 human player and 4 AI players run on it. It updates the gameboard and pop the redraw command as a runnable to the blocking queue.

The update snake method is protected by monitor so that it wont suffer race condition.

MapDBUtil

The mapDB class holds all player’s identifications.

Node

The nodes class represents a small block on the gameboard. This class in change of making colour changes, food state management.

The methods which might be accessed by multiple threads are surrounded by the monitor so that it will not suffer race condition.

NodeList

The node list class holds a 100x100 nodes gameboard. It is implemented with a concurrent map in case of race condition. It is also doing the gameboard updates, snake redraw and remove.

# LTSA Draft

PLAYER1 = (keypressed1->PLAYER1|die1->LOGIN1),

LOGIN1 = (login1->PLAYER1),

DIED1 = (revive1->PLAYER1|logout1->LOGIN1).

PLAYER2 = (keypressed2->PLAYER2|die2->DIED2),

LOGIN2 = (login2 -> PLAYER2),

DIED2 = (revive2->PLAYER2|logout2->LOGIN2).

BUFFERC1 = (keypressed1->keypressedRead1->BUFFERC1).

BUFFERC2 = (keypressed2->keypressedRead2->BUFFERC2).

SERVER1 = (keypressedRead1->SERVER1).

SERVER2 = (keypressedRead2->SERVER2).

GAMEBOARD = (keypressedRead1->GAMEBOARD|keypressedRead2->GAMEBOARD).

||SERVERCONTROLL = (PLAYER1||PLAYER2||GAMEBOARD||BUFFERC1||BUFFERC2||SERVER1||SERVER2).

Player1 represent the human controlled player, player2 represent AI players.

Each player is assigned with a dedicated server and buffer.

The player class handles keypress and passes to the server class. The server class will pass the data from player class to the gameboard. The gameboard is shared by 4 server threads and only one thread can access gameboard class at the same time.

The gameboard will repaint base on the data which is passed in from the server.

Each server class is allocated with a dedicated buffer, and each buffer knows the player and server which it is connected to.

The LTSA draft is in a same directory with the document.