# Client Side Code

## Board.java

package edu**.**cp**.**project**.**client**;**

**import** java**.**util**.\*;**

**import** java**.**awt**.\*;**

**import** java**.**awt**.**image**.**BufferedImage**;**

**import** java**.**io**.**File**;**

**import** java**.**io**.**IOException**;**

**import** java**.**net**.**MalformedURLException**;**

**import** javax**.**imageio**.**ImageIO**;**

**import** javax**.**swing**.\*;**

class Board **extends** JFrame **implements** Runnable **{**

private static final long serialVersionUID **=** **-**864005501773990684L**;**

JFrame frame**;**

ArrayList**<**COWClient**>** pacMenList **=** **new** ArrayList**<**COWClient**>();**

ArrayList**<**COWClient**>** ghostList **=** **new** ArrayList**<**COWClient**>();**

BufferedImage pacL **=** **null;**

BufferedImage pacR **=** **null;**

BufferedImage pacU **=** **null;**

BufferedImage pacD **=** **null;**

BufferedImage pacS **=** **null;**

BufferedImage wallimg **=** **null;**

BufferedImage ghostimg **=** **null;**

public Board**(**JFrame frame**,** String clientsPos**)** **{**

**this.**frame **=** frame**;**

**try** **{**

pacL **=** ImageIO**.**read**(new** File**(**"PacL.gif"**));**

pacR **=** ImageIO**.**read**(new** File**(**"PacR.gif"**));**

pacU **=** ImageIO**.**read**(new** File**(**"PacU.gif"**));**

pacD **=** ImageIO**.**read**(new** File**(**"PacD.gif"**));**

pacS **=** ImageIO**.**read**(new** File**(**"PacS.gif"**));**

wallimg **=** ImageIO**.**read**(new** File**(**"wall.jpg"**));**

ghostimg **=** ImageIO**.**read**(new** File**(**"ghost.gif"**));**

**}** **catch** **(**IOException e**)** **{**

**}**

String**[]** str **=** clientsPos**.**split**(**" "**);**

int i **=** 0**;**

**while** **(**str**[**i**].**equals**(**"G"**)** **==** **false)** **{**

COWClient cl **=** **new** COWClient**(**PAC\_SIZE**,** PAC\_SIZE**,** 's'**,** 's'**);**

changeClientsState**(**cl**,** str**,** i**);**

pacMenList**.**add**(**cl**);**

System**.**out**.**println**(**"pacman added"**);**

i **+=** 4**;**

**}**

i**++;**

**while** **(**i **!=** str**.**length**)** **{**

COWClient cl **=** **new** COWClient**(**11 **\*** PAC\_SIZE**,** 11 **\*** PAC\_SIZE**,** 's'**,** 's'**);**

changeClientsState**(**cl**,** str**,** i**);**

ghostList**.**add**(**cl**);**

System**.**out**.**println**(**"ghost added"**);**

i **+=** 4**;**

**}**

frame**.**setDefaultCloseOperation**(**JFrame**.**EXIT\_ON\_CLOSE**);**

drawPanel **=** **new** BoardPanel**();**

frame**.**getContentPane**().**add**(**drawPanel**);**

frame**.**setSize**(**MAX\_X**,** MAX\_Y**);**

frame**.**setVisible**(true);**

frame**.**setResizable**(false);**

frame**.**setLocation**(**400**,** 400**);**

**}**

int**[][]** mazeArray **=** **{** **{** 1**,** 1**,** 1**,** 1**,** 1**,** 1**,** 0**,** 1**,** 1**,** 1**,** 1**,** 1**,** 1 **},**

**{** 1**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 1 **},**

**{** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1 **},**

**{** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1 **},**

**{** 1**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 1 **},**

**{** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1 **},**

**{** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0 **},**

**{** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1 **},**

**{** 1**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 1 **},**

**{** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1 **},**

**{** 1**,** 0**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 1**,** 0**,** 1**,** 0**,** 1 **},**

**{** 1**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 1 **},**

**{** 1**,** 1**,** 1**,** 1**,** 1**,** 1**,** 0**,** 1**,** 1**,** 1**,** 1**,** 1**,** 1 **}** **};**

public void run**()** **{**

String gameOverMessage **=** **new** String**();**

**while** **(true)** **{**

**if** **(**pacMenList**.**size**()** **>** 0 **&&** ghostList**.**size**()** **>** 0**)** **{**

COWClient**.**ClientPosition pc**;** // pc is immutable

pc **=** pacMenList**.**get**(**0**).**getClientPosition**();**

int px **=** pc**.**getX**();**

int py **=** pc**.**getY**();**

**for** **(**COWClient ghost **:** ghostList**)** **{**

COWClient**.**ClientPosition gc**;**

gc **=** ghost**.**getClientPosition**();**

int gx **=** gc**.**getX**();**

int gy **=** gc**.**getY**();**

// This collision logic replaces the logic we had at the demo, and now detects collisions in all cases

**if** **(**Math**.**abs**(**gx **-** px**)** **<** PAC\_SIZE **&&** Math**.**abs**(**gy **-** py**)** **<** PAC\_SIZE**)** **{**

gameState **=** 2**;**

gameOverMessage **=** Client**.**GHOSTWINS**;**

**}**

//This is the old logic, which could have a failure

// if (gx / PAC\_SIZE == px / PAC\_SIZE) {

// if (gy / PAC\_SIZE == py / PAC\_SIZE) {

// gameState = 2;

// gameOverMessage = Client.GHOSTWINS;

// }

**}**

**}**

boolean pelletsExist **=** **false;**

**for** **(**int i **=** 0**;** i **<** 13**;** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** 13**;** **++**j**)** **{**

**if** **(**mazeArray**[**i**][**j**]** **==** 0**)**

pelletsExist **=** **true;**

**}**

**}**

**if** **(!**pelletsExist**)** **{**

gameState **=** 1**;**

gameOverMessage **=** Client**.**PACMANWINS**;**

**}**

**if** **(**gameState **!=** 0**)** **{**

**break;**

**}**

drawPanel**.**repaint**();**

**}**

Client**.**sendToServer**(**gameOverMessage**);**

**}**

class BoardPanel **extends** JPanel **{**

private static final long serialVersionUID **=** 1L**;**

public void paintComponent**(**Graphics comp**)** **{**

Graphics2D g **=** **(**Graphics2D**)** comp**;**

g**.**setColor**(**Color**.**black**);**

g**.**fillRect**(**0**,** 0**,** MAX\_X**,** MAX\_Y**);**

**for** **(**int i **=** 0**;** i **<** MAZE\_SIZE**;** i**++)** **{** // creating the board layout

**for** **(**int j **=** 0**;** j **<** MAZE\_SIZE**;** j**++)** **{**

**if** **(**mazeArray**[**i**][**j**]** **==** 1**)** **{**

g**.**drawImage**(**wallimg**,** j **\*** PAC\_SIZE**,** i **\*** PAC\_SIZE**,**

PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**}** **else** **if** **(**mazeArray**[**i**][**j**]** **==** 0**)** **{**

g**.**setColor**(**Color**.**MAGENTA**);**

g**.**fillOval**(**j **\*** PAC\_SIZE **+** 13**,** i **\*** PAC\_SIZE **+** 13**,** 3**,** 3**);**

**}**

**}**

**}** // finish board layout

**for** **(**COWClient pacMen **:** pacMenList**)** **{**

**try** **{**

eatPelletAndDrawClient**(**pacMen**,** g**);**

**}** **catch** **(**MalformedURLException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**for** **(**COWClient ghost **:** ghostList**)** **{**

drawClient**(**ghost**,** g**,** "G"**);**

**}**

**}**

void eatPelletAndDrawClient**(**COWClient pacMen**,** Graphics2D g**)**

**throws** MalformedURLException **{**

COWClient**.**ClientPosition tmpc**;**

tmpc **=** pacMen**.**getClientPosition**();**

int tmpx **=** tmpc**.**getX**();**

int tmpy **=** tmpc**.**getY**();**

**if** **((**tmpx **%** PAC\_SIZE **==** 0**)** **&&** **(**tmpy **%** PAC\_SIZE **==** 0**))** **{**

int i **=** tmpx **/** PAC\_SIZE**;**

int j **=** tmpy **/** PAC\_SIZE**;**

**if** **(**mazeArray**[**j**][**i**]** **==** 0**)** **{**

mazeArray**[**j**][**i**]** **=** **-**1**;** // eats pellet

**}**

**}**

drawClient**(**pacMen**,** g**,** "P"**);**

**}**

void drawClient**(**COWClient cl**,** Graphics2D g**,** String type**)** **{**

COWClient**.**ClientPosition tmpc**;**

tmpc **=** cl**.**getClientPosition**();**

int tmpx **=** tmpc**.**getX**();**

int tmpy **=** tmpc**.**getY**();**

char curDir **=** tmpc**.**getCurrDir**();**

char newDir **=** tmpc**.**getNewDir**();**

**if** **((**tmpx **%** PAC\_SIZE **==** 0**)** **&&** **(**tmpy **%** PAC\_SIZE **==** 0**))** **{**

int i **=** tmpx **/** PAC\_SIZE**;**

int j **=** tmpy **/** PAC\_SIZE**;**

**if** **(**newDir **==** 'u'**)** **{**

**if** **(**mazeArray**[(**j **+** 12**)** **%** 13**][**i**]** **==** 1**)** **{**

**if** **(**curDir **==** 'l' **&&** mazeArray**[**j**][(**i **+** 12**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'r'

**&&** mazeArray**[**j**][(**i **+** 1**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'd'

**&&** mazeArray**[(**j **+** 1**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **{**

curDir **=** 's'**;**

**}**

**}** **else** **{**

curDir **=** newDir**;**

**}**

**}** **else** **if** **(**newDir **==** 'd'**)** **{**

**if** **(**mazeArray**[(**j **+** 1**)** **%** 13**][**i**]** **==** 1**)** **{**

**if** **(**curDir **==** 'l' **&&** mazeArray**[**j**][(**i **+** 12**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'r'

**&&** mazeArray**[**j**][(**i **+** 1**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'u'

**&&** mazeArray**[(**j **+** 12**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **{**

curDir **=** 's'**;**

**}**

**}** **else** **{**

curDir **=** newDir**;**

**}**

**}** **else** **if** **(**newDir **==** 'l'**)** **{**

**if** **(**mazeArray**[**j**][(**i **+** 12**)** **%** 13**]** **==** 1**)** **{**

**if** **(**curDir **==** 'u' **&&** mazeArray**[(**j **+** 12**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'r'

**&&** mazeArray**[**j**][(**i **+** 1**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'd'

**&&** mazeArray**[(**j **+** 1**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **{**

curDir **=** 's'**;**

**}**

**}** **else** **{**

curDir **=** newDir**;**

**}**

**}** **else** **if** **(**newDir **==** 'r'**)** **{**

**if** **(**mazeArray**[**j**][(**i **+** 1**)** **%** 13**]** **==** 1**)** **{**

**if** **(**curDir **==** 'u' **&&** mazeArray**[(**j **+** 12**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'l'

**&&** mazeArray**[**j**][(**i **+** 12**)** **%** 13**]** **!=** 1**)** **{**

**}** **else** **if** **(**curDir **==** 'd'

**&&** mazeArray**[(**j **+** 1**)** **%** 13**][**i**]** **!=** 1**)** **{**

**}** **else** **{**

curDir **=** 's'**;**

**}**

**}** **else** **{**

curDir **=** newDir**;**

**}**

**}**

**}**

**if** **(**curDir **==** 's'**)** // movement not started. pacman in same

// position

**{**

**if** **(**type**.**equals**(**"P"**))**

g**.**drawImage**(**pacS**,** tmpx**,** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**else**

g**.**drawImage**(**ghostimg**,** tmpx**,** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**}** **else** **if** **(**curDir **==** 'u'**)** **{** // move in the UP direction

**if** **(**tmpy **==** 0 **-** PAC\_SIZE **/** 2**)** **{**

tmpy **=** 12 **\*** PAC\_SIZE **+** PAC\_SIZE **/** 2**;**

**}**

tmpy **-=** SPEED**;**

**if** **(**type**.**equals**(**"P"**))**

g**.**drawImage**(**pacU**,** tmpx**,** **(--**tmpy**),** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**else**

g**.**drawImage**(**ghostimg**,** tmpx**,** **(--**tmpy**),** PAC\_SIZE**,** PAC\_SIZE**,**

**this);**

**}** **else** **if** **(**curDir **==** 'd'**)** **{** // move in the DOWN direction

**if** **(**tmpy **==** 12 **\*** PAC\_SIZE **+** PAC\_SIZE **/** 2**)** **{**

tmpy **=** 0 **-** PAC\_SIZE **/** 2**;**

**}**

tmpy **+=** SPEED**;**

**if** **(**type**.**equals**(**"P"**))**

g**.**drawImage**(**pacD**,** tmpx**,** **(++**tmpy**),** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**else**

g**.**drawImage**(**ghostimg**,** tmpx**,** **(++**tmpy**),** PAC\_SIZE**,** PAC\_SIZE**,**

**this);**

**}** **else** **if** **(**curDir **==** 'l'**)** **{** //move in the LEFT direction

**if** **(**tmpx **==** 0 **-** PAC\_SIZE **/** 2**)** **{**

tmpx **=** 12 **\*** PAC\_SIZE **+** PAC\_SIZE **/** 2**;**

**}**

tmpx **-=** SPEED**;**

**if** **(**type**.**equals**(**"P"**))**

g**.**drawImage**(**pacL**,** **(--**tmpx**),** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**else**

g**.**drawImage**(**ghostimg**,** **(--**tmpx**),** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,**

**this);**

**}** **else** **if** **(**curDir **==** 'r'**)** **{**

**if** **(**tmpx **==** 12 **\*** PAC\_SIZE **+** PAC\_SIZE **/** 2**)** **{**

tmpx **=** **-**PAC\_SIZE **/** 2**;**

**}**

tmpx **+=** SPEED**;**

**if** **(**type**.**equals**(**"P"**))**

g**.**drawImage**(**pacR**,** **(++**tmpx**),** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,** **this);**

**else**

g**.**drawImage**(**ghostimg**,** **(++**tmpx**),** tmpy**,** PAC\_SIZE**,** PAC\_SIZE**,**

**this);**

**}**

synchronized**(this)** **{**

**if** **(**cl**.**getClientPosition**().**equals**(**tmpc**))** **{**

cl**.**setClientPosition**(**tmpx**,** tmpy**,** curDir**,** newDir**);**

**}**

**}**

**try** **{**

Thread**.**sleep**(**SLEEPTIME**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

public void changeClientsState**(**String clientPos**)** **{**

String**[]** str **=** clientPos**.**trim**().**split**(**" "**);**

int i **=** 0**;**

int cnt **=** 0**;**

**while** **(**str**[**i**].**equals**(**"G"**)** **==** **false)** **{**

changeClientsState**(**pacMenList**.**get**(**cnt**++),** str**,** i**);**

i **+=** 4**;**

**}**

i**++;**

cnt **=** 0**;**

**while** **(**i **<** str**.**length**)** **{**

changeClientsState**(**ghostList**.**get**(**cnt**++),** str**,** i**);**

i **+=** 4**;**

**}**

**}**

public void changeClientsState**(**COWClient cl**,** String**[]** str**,** int idx**)** **{**

**if** **(**str**[**idx**].**equals**(**"\*"**)** **==** **false)** **{**

cl**.**setClientPosition**(**Integer**.**parseInt**(**str**[**idx**]),**

Integer**.**parseInt**(**str**[**idx **+** 1**]),**

str**[**idx **+** 2**].**charAt**(**0**),**

str**[**idx **+** 3**].**charAt**(**0**));**

**}**

**}**

class COWClient **{**

// Immutable class

class ClientPosition **{**

final int x**;**

final int y**;**

final char currDir**;**

final char newDir**;**

ClientPosition**(**int x**,** int y**,** char currDir**,** char newDir**)** **{**

**this.**x **=** x**;**

**this.**y **=** y**;**

**this.**currDir **=** currDir**;**

**this.**newDir **=** newDir**;**

**}**

int getX**()** **{**

**return** x**;**

**}**

int getY() {

return y;

}

public char getCurrDir() {

return currDir;

}

char getNewDir() {

return newDir;

}

}

protected ClientPosition pos;

volatile char newDir = 's';

public COWClient(int x, int y, char c, char n) {

pos = new ClientPosition(x, y, c, n);

}

void setClientPosition(int x, int y, char c, char n){

pos = new ClientPosition(x, y, c, n);

}

ClientPosition getClientPosition() {

return pos;

}

}

BoardPanel drawPanel;

static final int MAZE\_SIZE = 13;

static final int PAC\_SIZE = 30;

static final int MAX\_X = 400; // widest the playing screen can be

static final int MAX\_Y = 420; // tallest the playing screen can be

static final int SPEED = 4;

public static final int SLEEPTIME = 12;

public COWClient.ClientPosition getMyClientPosition(int myClientIdx, String clientType) {

if (clientType.equals("P")) {

return pacMenList.get(myClientIdx).pos;

} else {

return ghostList.get(myClientIdx).pos;

}

}

// 0 - game is going on, 1 - pacman won, 2 - ghost won

// volatile to avoid data race

// it is updated in the run() function of this thread and

// in the main thread in the function listenToServer

public volatile int gameState = 0;

}

## Client.java

package edu**.**cp**.**project**.**client**;**

**import** java**.**awt**.**event**.**KeyEvent**;**

**import** java**.**awt**.**event**.**KeyListener**;**

**import** java**.**io**.\*;**

**import** java**.**net**.\*;**

**import** java**.**util**.**Arrays**;**

**import** javax**.**swing**.**JFrame**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**SwingUtilities**;**

public class Client **extends** JFrame **implements** KeyListener**{**

Client**(){**

frame**.**addKeyListener**(this);**

frame**.**setLocation**(**400**,**100**);**

REGMSG **=** **null;**

clientType **=** **null;**

**}**

void run**()**

**{**

**try{**

requestSocket **=** **new** DatagramSocket**();**

serverAddr **=** InetAddress**.**getByName**(**serverIpAddr**);**

registerWithServer**();**

getIdFromServer**();**

receiveInitialBoardStateAndInitializeBoard**();**

listenToServer**();**

**}**

**catch(**UnknownHostException unknownHost**){**

System**.**err**.**println**(**"You are trying to connect to an unknown host!"**);**

**}**

**catch(**IOException ioException**){**

ioException**.**printStackTrace**();**

**}**

**finally{**

requestSocket**.**close**();**

**}**

**}**

private void listenToServer**()** **throws** IOException **{**

**do{**

Arrays**.**fill**(**rcvPacket**.**getData**(),** **(**byte**)** 0**);**

requestSocket**.**receive**(**rcvPacket**);**

message **=** **new** String**(**rcvPacket**.**getData**()).**trim**();**

System**.**out**.**println**(**"server>" **+** message**);**

**if** **(**PACMANWINS**.**equals**(**message**))** **{**

JOptionPane**.**showMessageDialog**(**frame**,** "GAME OVER Pac-Man wins!!"**);**

board**.**gameState **=** 1**;**

**break;**

**}** **else** **if(**GHOSTWINS**.**equals**(**message**))** **{**

JOptionPane**.**showMessageDialog**(**frame**,** "GAME OVER Ghosts wins!!"**);**

board**.**gameState **=** 2**;**

**break;**

**}** **else** **{**

// create new thread so that we keep listening to server as

// board is being updated

**new** Thread**()** **{**

public void run**()** **{**

board**.**changeClientsState**(**message**);**

**}**

**}.**start**();**

**}**

**}** **while(true);**

**}**

void registerWithServer**()** **throws** IOException **{**

byte**[]** sendData **=** REGMSG**.**getBytes**();**

sendPacket **=** **new** DatagramPacket**(**sendData**,**

sendData**.**length**,**

serverAddr**,**

port**);**

requestSocket**.**send**(**sendPacket**);**

**}**

void getIdFromServer**()** **throws** IOException **{**

requestSocket**.**receive**(**rcvPacket**);**

message **=** **new** String**(**rcvPacket**.**getData**()).**trim**();**

myClientIdx **=** Integer**.**parseInt**(**message**);**

**}**

private void receiveInitialBoardStateAndInitializeBoard**()** **throws** IOException **{**

requestSocket**.**receive**(**rcvPacket**);**

message **=** **new** String**(**rcvPacket**.**getData**()).**trim**();**

System**.**out**.**println**(**"server >" **+** message**);**

board**=new** Board**(**frame**,**message**);**

SwingUtilities**.**invokeLater**(new** Runnable**()** **{**

public void run**()** **{**

initializeBoardUI**();**

**}**

**});**

**}**

static void sendToServer**(**String msg**)**

**{**

DatagramPacket sendPacket**;**

System**.**out**.**println**(**"client>"**+**msg**);**

byte**[]** sendData **=** msg**.**getBytes**();**

sendPacket **=** **new** DatagramPacket**(**sendData**,**

sendData**.**length**,**

serverAddr**,**

port**);**

**try** **{**

requestSocket**.**send**(**sendPacket**);**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

private void initializeBoardUI**()** **{**

**new** Thread**(**board**).**start**();**

**}**

public static void main**(**String args**[])**

**{**

Client client **=** **new** Client**();**

client**.**run**();**

**}**

@Override

public void keyTyped**(**KeyEvent e**)** **{**

**}**

@Override

public void keyPressed**(**KeyEvent e**)** **{**

int ckey**;**

ckey **=** e**.**getKeyCode**();**

**if** **(**ckey**==**38**){**

//UP

**if** **(**lastKeyPressed **!=** 1**)** **{**

prependClientInfoAndSendToServer**(**"u"**);**

lastKeyPressed **=** 1**;**

**}**

**}**

**if** **(**ckey**==**40**){**

//Down

**if** **(**lastKeyPressed **!=** 2**)** **{**

prependClientInfoAndSendToServer**(**"d"**);**

lastKeyPressed **=** 2**;**

**}**

**}**

**if** **(**ckey**==**37**){**

//Left

**if** **(**lastKeyPressed **!=** 3**)** **{**

prependClientInfoAndSendToServer**(**"l"**);**

lastKeyPressed **=** 3**;**

**}**

**}**

**if** **(**ckey**==**39**){**

//Right

**if** **(**lastKeyPressed **!=** 4**)** **{**

prependClientInfoAndSendToServer**(**"r"**);**

lastKeyPressed **=** 4**;**

**}**

**}**

**}**

public void prependClientInfoAndSendToServer**(**String msg**)** **{**

StringBuilder sb **=** **new** StringBuilder**();**

Board**.**COWClient**.**ClientPosition cl **=** board**.**getMyClientPosition**(**myClientIdx**,** clientType**);**

sb**.**append**(**cl**.**getX**())**

**.**append**(**" "**)**

**.**append**(**cl**.**getY**())**

**.**append**(**" "**)**

**.**append**(**cl**.**getCurrDir**())**

**.**append**(**" "**)**

**.**append**(**msg**);**

msg **=** sb**.**toString**();**

sendToServer**(**msg**);**

**}**

@Override

public void keyReleased**(**KeyEvent e**)** **{**

**}**

private static final long serialVersionUID **=** 1L**;**

static DatagramSocket requestSocket**;**

DatagramPacket sendPacket**;**

DatagramPacket rcvPacket **=** **new** DatagramPacket**(new** byte**[**100**],** 100**);**

String message**;**

static int port **=** 6790**;**

String serverIpAddr **=** "127.0.0.1"**;**

protected static String REGMSG**;**

protected static String clientType**;**

static JFrame frame**=new** JFrame**();**

static InetAddress serverAddr**;**

Board board**;**

int myClientIdx **=** 0**;**

static final String PACMANWINS **=** "PW"**;**

static final String GHOSTWINS **=** "GW"**;**

int lastKeyPressed **=** 0**;**

**}**

## GhostClient.java

package edu**.**cp**.**project**.**client**;**

@SuppressWarnings**(**"serial"**)**

public class GhostClient **extends** Client **{**

GhostClient**()** **{**

**super();**

REGMSG **=** "REG ggg G"**;**

serverIpAddr **=** "127.0.0.1"**;**

clientType **=** "G"**;**

**}**

public static void main**(**String**[]** args**)** **{**

**new** GhostClient**().**run**();**

**}**

**}**

## PacManClient.java

package edu**.**cp**.**project**.**client**;**

@SuppressWarnings**(**"serial"**)**

public class PacmanClient **extends** Client **{**

PacmanClient**()** **{**

**super();**

REGMSG **=** "REG ppp P"**;**

serverIpAddr **=** "127.0.0.1"**;**

clientType **=** "P"**;**

**}**

public static void main**(**String**[]** args**)** **{**

**new** PacmanClient**().**run**();**

**}**

**}**

# Server Side code

## Server.java

package edu**.**cp**.**project**.**server**;**

// PacMen Server

**import** java**.**io**.**IOException**;**

**import** java**.**net**.\*;**

**import** java**.**util**.\*;**

public final class Server **implements** Runnable **{**

private static final int PACKETSIZE **=** 100**;**

public static void main**(**String**[]** args**)** **{**

**new** Thread**(new** Server**()).**start**();**

**}**

public void run**()** **{**

**try** **{**

System**.**out**.**println**(**"Hi there"**);**

serverSocket **=** **new** DatagramSocket**(**6790**);**

System**.**out**.**println**(**"Server socket started"**);**

DatagramPacket packet **=** **new** DatagramPacket**(new** byte**[**PACKETSIZE**],** PACKETSIZE**)** **;**

registerClients**(**packet**);**

board **=** **new** ClientsPositions**(**pacMenClients**,** ghostClients**);**

// send initial configuration to clients

broadcast**(**board**.**toString**());**

listenAndRespondToClients**(**packet**);**

**}** **catch** **(**IOException e**)** **{**

System**.**err**.**println**(**"Could not listen on port: 4444."**);**

**}** **finally** **{**

serverSocket**.**close**();**

**}**

**}**

private void registerClients**(**DatagramPacket packet**)** **{**

**for** **(**int i **=** 0**;** i **<** NUMCLIENT**;** i**++)** **{**

**try** **{**

Arrays**.**fill**(**packet**.**getData**(),** **(**byte**)** 0**);**

serverSocket**.**receive**(**packet**);**

ClientInfo clientInfo **=** **new** ClientInfo**(**packet**);**

map**.**put**(**clientInfo**.**addr**.**toString**()** **+** clientInfo**.**port**,** clientInfo**);**

**if** **(**clientInfo**.**clientType **==** RegisterClientMessage**.**

CLIENT\_TYPE**.**GHOST**)** **{**

ghostClients**.**add**(**clientInfo**);**

sendToClient**(**clientInfo**,** Integer**.**toString**(**ghostClients**.**size**()** **-** 1**));**

**}** **else** **{**

pacMenClients**.**add**(**clientInfo**);**

sendToClient**(**clientInfo**,** Integer**.**toString**(**pacMenClients**.**size**()** **-** 1**));**

**}**

**}** **catch** **(**Exception e**)** **{**

System**.**err**.**println**(**e**.**getMessage**());**

System**.**exit**(**1**);**

**}**

**}**

**}**

private void sendToClient**(**ClientInfo clientInfo**,** String msg**)** **{**

System**.**out**.**println**(**msg **+** " sending to client"**);**

DatagramPacket sendPacket**;**

byte**[]** sendData **=** msg**.**getBytes**();**

sendPacket **=** **new** DatagramPacket**(**sendData**,**

sendData**.**length**,**

clientInfo**.**addr**,**

clientInfo**.**port**);**

**try** **{**

serverSocket**.**send**(**sendPacket**);**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

private void listenAndRespondToClients**(**DatagramPacket packet**)**

**throws** IOException **{**

**while** **(true)** **{**

Arrays**.**fill**(**packet**.**getData**(),** **(**byte**)** 0**);**

serverSocket**.**receive**(**packet**);**

String msg **=** **new** String**(**packet**.**getData**()).**trim**();**

System**.**out**.**println**(**"client> " **+** msg**);**

**if** **(**PACMANWINS**.**equals**(**msg**))** **{**

broadcast**(**PACMANWINS**);**

**}** **else** **if** **(**GHOSTWINS**.**equals**(**msg**))** **{**

broadcast**(**GHOSTWINS**);**

**}** **else** **{**

updateClientsPositionAndBroadcast**(**map**.**get**(**packet**.**getAddress**().**toString**()**

**+** packet**.**getPort**()),**

msg**);**

**}**

**}**

**}**

private void updateClientsPositionAndBroadcast**(**ClientInfo

clientInfo**,**

String clientMsg**)** **{**

String boardState **=** board**.**updateClientPosition**(**clientInfo**,** clientMsg**);**

**if** **(**boardState **!=** **null)** **{**

broadcast**(**boardState**);**

**}**

**}**

private void broadcast**(**String boardState**)** **{**

**for** **(**ClientInfo clientInfo **:** pacMenClients**)** **{**

sendToClient**(**clientInfo**,** boardState**);**

**}**

**for** **(**ClientInfo clientInfo **:** ghostClients**)** **{**

sendToClient**(**clientInfo**,** boardState**);**

**}**

**}**

static List**<**ClientInfo**>** pacMenClients **=** **new** ArrayList**<**ClientInfo**>();**

static List**<**ClientInfo**>** ghostClients **=** **new** ArrayList**<**ClientInfo**>();**

static DatagramSocket serverSocket **=** **null;**

static ClientsPositions board**;**

Socket clientSocket**;**

HashMap**<**String**,** ClientInfo**>** map **=** **new** HashMap**<**String**,** ClientInfo**>();**

private static final int NUMCLIENT **=** 2**;**

private static final String PACMANWINS **=** "PW"**;**

private static final String GHOSTWINS **=** "GW"**;**

**}**

## RegisterClientMessage.java

package edu**.**cp**.**project**.**server**;**

**import** java**.**util**.**regex**.**Pattern**;**

class RegisterClientMessage **{**

RegisterClientMessage**(**String msg**)** **{**

**if** **(**Pattern**.**matches**(**REG\_MSG**,** msg**)** **==** **false)** **{**

**throw** **new** RuntimeException**(**"Not reg msg from client: " **+** msg**);**

**}**

**else** **{**

String**[]** msgSplit **=** msg**.**split**(**" "**);**

clientName **=** msgSplit**[**1**];**

**if** **(**msgSplit**[**2**].**equals**(**"G"**))** **{**

clientType **=** CLIENT\_TYPE**.**GHOST**;**

**}** **else** **{**

clientType **=** CLIENT\_TYPE**.**PACMEN**;**

**}**

**}**

**}**

public String getClientName**()** **{**

**return** clientName**;**

**}**

CLIENT\_TYPE getClientType**()** **{**

**return** clientType**;**

**}**

String clientName **=** **null;**

CLIENT\_TYPE clientType **=** **null;**

public static final String REG\_MSG **=** "REG [a-zA-Z]\* [GP]"**;**

public enum CLIENT\_TYPE **{**PACMEN**,** GHOST**};**

**}**

## ClientInfo.java

package edu**.**cp**.**project**.**server**;**

**import** java**.**net**.**DatagramPacket**;**

**import** java**.**net**.**InetAddress**;**

class ClientInfo **{**

ClientInfo**(**DatagramPacket pkt**)** **{**

addr **=** pkt**.**getAddress**();**

port **=** pkt**.**getPort**();**

String clientMsg **=** **new** String**(**pkt**.**getData**());**

System**.**out**.**println**(**"Client msg: " **+** clientMsg**);**

RegisterClientMessage registerClientMsg **=**

**new** RegisterClientMessage**(**clientMsg**.**trim**());**

clientName **=** registerClientMsg**.**clientName**;**

clientType **=** registerClientMsg**.**clientType**;**

**}**

public String toString**()** **{**

**return** clientName **+** " " **+** clientType**;**

**}**

String clientName**;**

RegisterClientMessage**.**CLIENT\_TYPE clientType**;**

InetAddress addr**;**

int port**;**

**}**

## ClientsPositions.java

package edu**.**cp**.**project**.**server**;**

**import** java**.**util**.\*;**

**import** java**.**util**.**List**;**

class ClientsPositions **{**

private static final int PAC\_SIZE **=** 30**;**

ArrayList**<**ClientPos**>** pacMenList **=** **new** ArrayList**<**ClientPos**>();**

ArrayList**<**ClientPos**>** ghostList **=** **new** ArrayList**<**ClientPos**>();**

public ClientsPositions**(**List**<**ClientInfo**>** pacMenClients**,**

List**<**ClientInfo**>** ghostClients**)** **{**

**for** **(**ClientInfo pacMenClient **:** pacMenClients**)** **{**

ClientPos cl **=** **new** ClientPos**(**PAC\_SIZE**,** PAC\_SIZE**,** "s"**,** "s"**);**

clientInfoToClient**.**put**(**pacMenClient**,** cl**);**

pacMenList**.**add**(**cl**);**

**}**

**for** **(**ClientInfo ghostClient **:** ghostClients**)** **{**

ClientPos cl **=** **new** ClientPos**(**11 **\*** PAC\_SIZE**,** 11 **\*** PAC\_SIZE**,** "s"**,** "s"**);**

clientInfoToClient**.**put**(**ghostClient**,** cl**);**

ghostList**.**add**(**cl**);**

**}**

**}**

public String toString**()** **{**

StringBuilder sb **=** **new** StringBuilder**();**

**for** **(**ClientPos pacMen **:** pacMenList**)** **{**

sb**.**append**(**pacMen**.**toString**());**

**}**

sb**.**append**(**"G"**).**append**(**" "**);**

**for** **(**ClientPos ghost **:** ghostList**)** **{**

sb**.**append**(**ghost**.**toString**());**

**}**

**return** sb**.**toString**().**trim**();**

**}**

public String updateClientPosition**(**ClientInfo clientInfo**,** String clientMsg**)** **{**

ClientPos cl **=** clientInfoToClient**.**get**(**clientInfo**);**

String**[]** str **=** clientMsg**.**split**(**" "**);**

int idx **=** 0**;**

cl**.**x **=** Integer**.**parseInt**(**str**[**idx**]);**

cl**.**y **=** Integer**.**parseInt**(**str**[**idx **+** 1**]);**

cl**.**currd **=** str**[**idx **+** 2**];**

cl**.**newd **=** str**[**idx **+** 3**];**

cl**.**flag **=** **true;**

**return** toString**();**

**}**

class ClientPos **{**

boolean flag **=** **false;**

int x **=** 0**;**

int y **=** 0**;**

String currd**;**

String newd**;**

String str **=** "\* \* \* \* "**;**

ClientPos**(**int x**,** int y**,** String cd**,** String nd**)** **{**

**this.**x **=** x**;**

**this.**y **=** y**;**

currd **=** cd**;**

newd **=** nd**;**

**}**

public String toString**()** **{**

StringBuilder sb **=** **new** StringBuilder**();**

**if** **(**flag**)** **{**

sb**.**append**(**x**)**

**.**append**(**" "**)**

**.**append**(**y**)**

**.**append**(**" "**)**

**.**append**(**currd**)**

**.**append**(**" "**)**

**.**append**(**newd**)**

**.**append**(**" "**);**

flag **=** **false;**

**return** sb**.**toString**();**

**}**

**else** **{**

**return** str**;**

**}**

**}**

**}**

Map**<**ClientInfo**,** ClientPos**>** clientInfoToClient **=** **new** HashMap**<**ClientInfo**,** ClientPos**>();**

**}**