**Course**: Principals of Software Development – ENSF 409

**Lab 3**

**Instructor**: M. Moshirpour

**Student Names**: Mitchell Sawatzky, Connor Newman

**Date Submitted**: Feb 5, 2016

Exercise B

Point.java

/\*\*

\* ENSF 409 - Lab 3 - Winter 2015

\* Started by: Mahmood Moussavi

\* January 22, 2015

\* Completed by: Mitchell Sawatzky & Connor Newman

\*/

class Point {

private int x, y;

public Point(int x, int y) {

this.x = x;

this.y = y;

}

static public double distance(Point a, Point b){

double diffx = a.x - b.x;

double diffy = a.y - b.y;

return Math.sqrt(diffx \* diffx + diffy \* diffy);

}

public String toString(){

String s = "(" + x + ", " + y + ")";

return s;

}

}

Line.java

/\*\*

\* ENSF 409 - Lab 3 - Winter 2015

\* Started by: Mahmood Moussavi

\* January 22, 2015

\* Completed by: Mitchell Sawatzky & Connor Newman

\*/

class Line {

Point start, end;

private static int classID = 0;

private int objID;

public Line(Point a, Point b) {

start = a;

end = b;

objID = ++ classID;

}

public double distance(){

return Point.distance(start, end);

}

public String toString()

{

String s = "Line " + objID + ": starts at " + start.toString() + ", and ends at " + end.toString();

return s;

}

}

Polygon.java

/\*\*

\* ENSF 409 - Lab 3 - Winter 20115

\* Started by: Mahmood Moussavi

\* January 22, 2015

\* Completed by: Mitchell Sawatzky & Connor Newman

\*/

import java.util.\*;

class Polygon {

private final LinkedHashSet <Line> polygon;

private int objID;

private static int classID;

Iterator <Line> it;

public Polygon(LinkedHashSet<Line> polygon) {

this.polygon = new LinkedHashSet<Line>();

for(Line l: polygon)

this.polygon.add (l);

objID = ++ classID;

it = this.polygon.iterator();

}

public Iterator <Line> getLine() {

it = polygon.iterator();

return it;

}

public static int classID(){

return classID;

}

public String toString() {

String s = "\nThe lines in polygon " + objID + " are:";

for (Line l : polygon)

s += "\n " + l.toString();

return s;

}

}

Terminal Output

The lines in polygon 1 are:

Line 1: starts at (20, 30), and ends at (50, 100)

Line 2: starts at (50, 100), and ends at (105, 30)

Line 3: starts at (105, 30), and ends at (20, 30)

The perimeter of the polygon 1 is 250.18:

The lines in polygon 2 are:

Line 4: starts at (120, 130), and ends at (150, 200)

Line 5: starts at (150, 200), and ends at (200, 130)

Line 6: starts at (200, 130), and ends at (120, 130)

The perimeter of the polygon 2 is 242.18:

The lines in polygon 3 are:

Line 7: starts at (320, 330), and ends at (250, 400)

Line 8: starts at (250, 400), and ends at (400, 330)

Line 9: starts at (400, 330), and ends at (320, 330)

The perimeter of the polygon 3 is 344.52:

Exercise C

**Important note regarding Game.java:**

**The UML diagram, sample program output, and exercise instructions for this project did not include the create\_player method defined in Game.java, nor did it include a HumanPlayer, RandomPlayer, BlockingPlayer, or SmartPlayer class. The exercise instructions did not ask us to define 3 types of AI as well as the Player class. In order for the package to compile, the sections of Game.java referring to the method and classes above were commented out.**

Constants.java

//Constants.java

/\*\*

\* Provides constants to the rest of the Package.

\* @version 1.0

\* @author Originally written by Mahmood Moussavi, modified by Mitchell Sawatzky and Connor Newman

\* @since Originally written in fall 2001, adapted in 2014, modified in 2016

\*/

public interface Constants {

/\*\*

\* The character to use when the board slot is empty.

\*/

static final char SPACE\_CHAR = ' ';

/\*\*

\* The character to use when Player O has entered into the board.

\*/

static final char LETTER\_O = 'O';

/\*\*

\* The character to use when Player X has entered into the board.

\*/

static final char LETTER\_X = 'X';

}

Player.java

//Player.java

import java.util.Scanner;

/\*\*

\* Provides a container to hold a Player's name and preferred mark (X or O), as well as logic to make moves.

\* @author Mitchell Sawatzky and Connor Newman

\* @version 1.0

\* @since Feb 5, 2016

\*/

public class Player implements Constants {

/\*\*

\* The name of the player.

\*/

private String name;

/\*\*

\* The player's mark, either 'X' or 'O'.

\*/

private char mark;

/\*\*

\* The player's opponent.

\*/

private Player opp;

/\*\*

\* The Board to play the game on.

\*/

private Board b;

/\*\*

\* Constructs a Player Object with a given name, mark, and Board.

\* @param name the Player's name

\* @param mark the Player's mark, either 'X' or 'O'

\* @param b the Board to play the game on

\*/

public Player(String name, char mark, Board b) {

this.name = name;

this.mark = mark;

this.b = b;

}

/\*\*

\* Getter function for the Player's name.

\* @return the String name of the player

\*/

public String getName() {

return this.name;

}

/\*\*

\* Sets the opponent of a given Player to another Player.

\* @param opp the Player opponent

\*/

public void setOpponent(Player opp) {

this.opp = opp;

}

/\*\*

\* Initiate a game of tic-tac-toe with the opponent player.

\*/

public void play() {

String winner;

Player p = this;

while (true) {

if (b.isFull()) {

winner = "Nobody";

break;

} else if (b.xWins() == 1) {

winner = this.name;

break;

} else if (b.oWins() == 1) {

winner = this.opp.getName();

break;

}

p.makeMove();

b.display();

p = p.opp;

}

System.out.printf("\nTHE GAME IS OVER: %s is the winner!\n", winner);

}

/\*\*

\* Prompt the user to place their mark on a given board slot retrieved through stdin.

\*/

public void makeMove() {

int row, col;

Player p = this;

while (true) {

while (true) {

System.out.printf("%s, what row should your next %c be placed in? ", p.name, p.mark);

Scanner input = new Scanner(System.in);

row = input.nextInt();

if (row < 0 || row > 2)

System.out.printf("\nInvalid row: %d, please try again.\n", row);

else

break;

}

while (true) {

System.out.printf("%s, what column should your next %c be placed in? ", name, mark);

Scanner input = new Scanner(System.in);

col = input.nextInt();

if (col < 0 || col > 2)

System.out.printf("\nInvalid column: %d, please try again.\n", col);

else

break;

}

if (b.getMark(row, col) == SPACE\_CHAR) {

b.addMark(row, col, mark);

break;

} else {

System.out.printf("\nThe coordinate (%d, %d) has already been used.\n", row, col);

}

}

}

}

Referee.java

//Referee.java

/\*\*

\* Mediates and controls a game of Tic Tac Toe.

\* Begins the game by printing the board, and then asks Player X to choose

\* @author Mitchell Sawatzky and Connor Newman

\* @version 1.0

\* @since Feb 5, 2016

\*/

public class Referee {

/\*\*

\* Player X of the game.

\*/

private Player x;

/\*\*

\* Player O of the game.

\*/

private Player o;

/\*\*

\* The board to play on.

\*/

private Board b;

/\*\*

\* Construct a Referee object from Players and a Board.

\* @param board the Board for the referee to control

\* @param xPlayer the player with the mark 'X'

\* @param oPlayer the player with the mark 'O'

\*/

public Referee(Board board, Player xPlayer, Player oPlayer) {

this.b = board;

this.x = xPlayer;

this.o = oPlayer;

}

/\*\*

\* Initiate a game with Player X as the starting player.

\*/

public void runTheGame() {

x.setOpponent(o);

o.setOpponent(x);

b.display();

x.play();

System.out.println("\033[1mGame ended ...\033[0m");

}

}

Board.java

// Board.java

// ENSF 409 - LAB 3 - Ex. C

// This file was originally written for ENGG 335 in fall 2001, and was

// adapted for ENSF 409 in 2014

//

/\*\*

\* Provides a tic-tac-toe board and logic to fill, empty, and test if a player has won.

\* @author Originally written by Mahmood Moussavi, modified by Mitchell Sawatzky and Connor Newman

\* @version 1.0

\* @since Originally written in fall 2001, adapted in 2014, modified in 2016

\*/

public class Board implements Constants {

/\*\*

\* Two-Dimensional char array to hold the values of each slot on the board

\*/

private char theBoard[][];

/\*\*

\* The total number of slots filled in on the board.

\*/

private int markCount;

/\*\*

\* Constructs a Board object without any spaces filled in.

\*/

public Board() {

markCount = 0;

theBoard = new char[3][];

for (int i = 0; i < 3; i++) {

theBoard[i] = new char[3];

for (int j = 0; j < 3; j++)

theBoard[i][j] = SPACE\_CHAR;

}

}

/\*\*

\* Returns the value of a board slot at a given row and column.

\* @param row the row to retrieve the board slot from

\* @param col the column to retrieve the board slot from

\* @return the Character value of the board slot

\*/

public char getMark(int row, int col) {

return theBoard[row][col];

}

/\*\*

\* Returns whether or not the board has values in all 9 slots.

\* @return True if all 9 slots are full, False otherwise

\*/

public boolean isFull() {

return markCount == 9;

}

/\*\*

\* Checks whether or not the letter X has won on the current board.

\* @return 0 if X has not won, 1 otherwise

\*/

public int xWins() {

return checkWinner(LETTER\_X);

}

/\*\*

\* Checks whether or not the letter O has won on the current board.

\* @return 0 if O has not won, 1 otherwise

\*/

public int oWins() {

return checkWinner(LETTER\_O);

}

/\*\*

\* Prints the board to stdout.

\*/

public void display() {

displayColumnHeaders();

addHyphens();

for (int row = 0; row < 3; row++) {

addSpaces();

System.out.print(" row " + row + ' ');

for (int col = 0; col < 3; col++)

System.out.print("| " + getMark(row, col) + " ");

System.out.println("|");

addSpaces();

addHyphens();

}

}

/\*\*

\* Sets the value of the board slot at a given row and column.

\* @param row the row to set the slot value

\* @param col the column to set the slot value

\* @param mark the Character to set the slot to

\*/

public void addMark(int row, int col, char mark) {

theBoard[row][col] = mark;

markCount++;

}

/\*\*

\* Resets every value on the board to SPACE\_CHAR.

\*/

public void clear() {

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

theBoard[i][j] = SPACE\_CHAR;

markCount = 0;

}

/\*\*

\* Uses tic-tac-toe logic to determine if a specific player has won.

\* @param mark the player to check, either LETTER\_X or LETTER\_O

\* @return 0 if the player has lost, 1 otherwise

\*/

int checkWinner(char mark) {

int row, col;

int result = 0;

for (row = 0; result == 0 && row < 3; row++) {

int row\_result = 1;

for (col = 0; row\_result == 1 && col < 3; col++)

if (theBoard[row][col] != mark)

row\_result = 0;

if (row\_result != 0)

result = 1;

}

for (col = 0; result == 0 && col < 3; col++) {

int col\_result = 1;

for (row = 0; col\_result != 0 && row < 3; row++)

if (theBoard[row][col] != mark)

col\_result = 0;

if (col\_result != 0)

result = 1;

}

if (result == 0) {

int diag1Result = 1;

for (row = 0; diag1Result != 0 && row < 3; row++)

if (theBoard[row][row] != mark)

diag1Result = 0;

if (diag1Result != 0)

result = 1;

}

if (result == 0) {

int diag2Result = 1;

for (row = 0; diag2Result != 0 && row < 3; row++)

if (theBoard[row][3 - 1 - row] != mark)

diag2Result = 0;

if (diag2Result != 0)

result = 1;

}

return result;

}

/\*\*

\* Print the board's column headers to stdout.

\*/

void displayColumnHeaders() {

System.out.print(" ");

for (int j = 0; j < 3; j++)

System.out.print("|col " + j);

System.out.println();

}

/\*\*

\* Adds a line to separate the board's rows.

\*/

void addHyphens() {

System.out.print(" ");

for (int j = 0; j < 3; j++)

System.out.print("+-----");

System.out.println("+");

}

/\*\*

\* Adds spacing inside the board to correctly place the values of the slots.

\*/

void addSpaces() {

System.out.print(" ");

for (int j = 0; j < 3; j++)

System.out.print("| ");

System.out.println("|");

}

}

Game.java

//Game.java

import java.io.\*;

/\*\*

\* @author Started by: M. Moussavi

\* Completed by: Mitchell Sawatzky and Connor Newman

\* Asks the user to select a player type, creates the player, creates the board,

\* assigns a referee to the game, then initiates the game.

\* @version 1.0

\* @since Feb 2016

\*/

public class Game implements Constants {

/\*\*

\* The Board to play the Game on.

\*/

private Board theBoard;

/\*\*

\* The Referee to control the Game.

\*/

private Referee theRef;

/\*\*

\* creates a board for the game.

\*/

public Game( ) {

theBoard = new Board();

}

/\*\*

\* calls the referee method runTheGame

\* @param r refers to the appointed referee for the game

\* @throws IOException when a player inputs something unparsable

\*/

public void appointReferee(Referee r) throws IOException {

theRef = r;

theRef.runTheGame();

}

public static void main(String[] args) throws IOException {

Referee theRef;

Player xPlayer, oPlayer;

BufferedReader stdin;

Game theGame = new Game();

stdin = new BufferedReader(new InputStreamReader(System.in));

System.out.print("\nPlease enter the name of the \'X\' player: ");

String name= stdin.readLine();

while (name == null) {

System.out.print("Please try again: ");

name = stdin.readLine();

}

// xPlayer = create\_player (name, LETTER\_X, theGame.theBoard, stdin);

xPlayer = new Player(name, LETTER\_X, theGame.theBoard);

System.out.print("\nPlease enter the name of the \'O\' player: ");

name = stdin.readLine();

while (name == null) {

System.out.print("Please try again: ");

name = stdin.readLine();

}

// oPlayer = create\_player (name, LETTER\_O, theGame.theBoard, stdin);

oPlayer = new Player(name, LETTER\_O, theGame.theBoard);

theRef = new Referee(theGame.theBoard, xPlayer, oPlayer);

theGame.appointReferee(theRef);

}

/\*\*

\* Creates the specified type of player indicated by the user.

\*

\* @param name player's name

\* @param mark player's mark (X or O)

\* @param board refers to the game board

\* @param stdin refers to an input stream

\* @return a newly created player

\* @throws IOException

\*/

// static public Player create\_player(String name, char mark, Board board,

// BufferedReader stdin)throws IOException {

// // Get the player type.

// final int NUMBER\_OF\_TYPES = 4;

// System.out.print ( "\nWhat type of player is " + name + "?\n");

// System.out.print(" 1: human\n" + " 2: Random Player\n"

// + " 3: Blocking Player\n" + " 4: Smart Player\n");

// System.out.print( "Please enter a number in the range 1-" + NUMBER\_OF\_TYPES + ": ");

// int player\_type = 0;

//

// String input;

// stdin = new BufferedReader(new InputStreamReader(System.in));

// input= stdin.readLine();

// player\_type = Integer.parseInt(input);

// while (player\_type < 1 || player\_type > NUMBER\_OF\_TYPES) {

// System.out.print( "Please try again.\n");

// System.out.print ( "Enter a number in the range 1-" +NUMBER\_OF\_TYPES + ": ");

// input= stdin.readLine();

// player\_type = Integer.parseInt(input);

// }

//

// // Create a specific type of Player

// Player result = null;

// switch(player\_type) {

// case 1:

// result = new HumanPlayer(name, mark, board);

// break;

// case 2:

// result = new RandomPlayer(name, mark, board);

// break;

// case 3:

// result = new BlockingPlayer(name, mark, board);

// break;

// case 4:

// result = new SmartPlayer(name, mark, board);

// break;

// default:

// System.out.print ( "\nDefault case in switch should not be reached.\n"

// + " Program terminated.\n");

// System.exit(0);

// }

// return result;

// }

}

Sample Terminal Output

Mitchell@ttys000 11:44 {0} [tic]$ java Game

Please enter the name of the 'X' player: John

Please enter the name of the 'O' player: Sandy

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | | | |

| | | |

+-----+-----+-----+

| | | |

row 1 | | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | | | |

| | | |

+-----+-----+-----+

John, what row should your next X be placed in? 0

John, what column should your next X be placed in? 0

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | |

| | | |

+-----+-----+-----+

| | | |

row 1 | | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | | | |

| | | |

+-----+-----+-----+

Sandy, what row should your next O be placed in? 0

Sandy, what column should your next O be placed in? 2

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | O |

| | | |

+-----+-----+-----+

| | | |

row 1 | | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | | | |

| | | |

+-----+-----+-----+

John, what row should your next X be placed in? 1

John, what column should your next X be placed in? 0

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | O |

| | | |

+-----+-----+-----+

| | | |

row 1 | X | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | | | |

| | | |

+-----+-----+-----+

Sandy, what row should your next O be placed in? 0

Sandy, what column should your next O be placed in? 2

The coordinate (0, 2) has already been used.

Sandy, what row should your next O be placed in? 2

Sandy, what column should your next O be placed in? 0

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | O |

| | | |

+-----+-----+-----+

| | | |

row 1 | X | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | O | | |

| | | |

+-----+-----+-----+

John, what row should your next X be placed in? 2

John, what column should your next X be placed in? 2

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | O |

| | | |

+-----+-----+-----+

| | | |

row 1 | X | | |

| | | |

+-----+-----+-----+

| | | |

row 2 | O | | X |

| | | |

+-----+-----+-----+

Sandy, what row should your next O be placed in? 1

Sandy, what column should your next O be placed in? 1

|col 0|col 1|col 2

+-----+-----+-----+

| | | |

row 0 | X | | O |

| | | |

+-----+-----+-----+

| | | |

row 1 | X | O | |

| | | |

+-----+-----+-----+

| | | |

row 2 | O | | X |

| | | |

+-----+-----+-----+

THE GAME IS OVER: Sandy is the winner!

Game ended ...