Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ОТЧЁТ №3**

**Дисциплина: Многоагентное моделирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. В. Сидоренко

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Миков

Краснодар

2024

**Цель работы**

Написать программу по заданному ниже ТЗ, используя любой язык программирования.

**Большой теннис**

**Контекст**

Агент играет 1 на 1 против болванчика. Алгоритм болванчика достаточно прост: при подаче мяча в радиус действия (окружность радиуса r)

он всегда отбивает его в случайную точку половины корта агента. Агент может отбить мяч, который попадает в его радиус действия

(полукруг перед ним радиуса 2r, агент всегда смотрит в сторону противника). При отбивании агент, как и болванчик, могут перемещаться

не более, чем на l от той точки, в которой они находятся. Цель агента - победить болванчика. Для этого он пользуется следующей

стратегией: он делит половину корта болванчика на квадраты. В момент отбития он выбирает один из квадратов, в который он отправит

мяч (алгоритм выбора квадрата реализующий разрабатывает самостоятельно). Ввиду неидеальности исполнения, мяч может с 5% вероятностью

попасть в один из соседних клеток или в аут, если выбранный квадрат находится у края корта (в квадрате мяч попадает в случайную точку).

**Задача**

Придумать и описать алгоритм выбора квадрата. Провести моделирование с изменяемыми параметрами (радиус r, максимальное расстояние

передвижения l и количество квадратов n). Построить графики вероятности победы агента в матче (для построения графика, одна из

переменных может быть константой, а две другие проецироваться на оси координат).

**Ремарки**

Пример алгоритма выбора квадрата: агент рассчитывает на две или более итерации свои удары так, чтобы болванчик не успел добраться до

точки попадания следующего удара. Если результаты моделирования не изменяются от параметров, указанных в задаче, можно попробовать

изменить соотношение радиусов действия агента и болванчика или их максимальное преодолимое расстояние. Агент теряет очки (их получает

болванчик) если его мяч улетел в аут или он не смог отбить подачу болванчика. Болванчик теряет очки, если не отбивает удары агента

(в аут болванчик попасть не может).

**Правила большого тениса**

Подаёт всегда агент, вне зависимости от того выиграл или проиграл подачу. Подача происходит с задней линии (область А) в области E

или F (при подаче считается, что агент не может промахнуться и всегда попадает в выбранный квадрат). При подаче агент стоит по центру

задней линии, а болванчик на пересечении областей D, E, F. Аутом считается вся область за задней линией и полосы по краям, т.е. все

области, кроме A, B, C, D, E, F.

**Счёт в гейме**

Счёт ведётся следующим образом: 0-15-30-40-гейм. Если возникает ситуация, при которой у обоих игроков по 40 очков (ровно), то далее

игра происходит до тех пор, пока разница не будет равняться двум выигранным мячам. Игрок, который выиграл одно очко после счёта

"ровно", получает оценку "больше".

**Сет**

Очки 0-15-30... и т.д. складываются в геймы, геймы в сеты, сеты в матчи. Самый распространённый формат проведения соревнований

заключается в том, что для выигрыша одного сета нужно выиграть 6 геймов, но с разницей в 2 очка. Т.е. если счёт на табло значится

6-5, то проводится ещё один гейм, чтобы получилось 7-5. Но если счёт становится равный 6-6, то играется "тай-брейк". Это особый гейм,

в котором за каждый выигранный мяч присваивается одно очко, счёт ведётся до семи. Опять же вступает правило: разница в счёте должна

равняться минимум двум очкам (7/5, 7/4 и т.д.). Но если счёт становится 7-6, то игра продолжается до тех пор, пока разница не станет

равной двум.

**Счёт в сетах**

Игрок, который выиграл сет, получает в свой актив одно очко. Большинство соревнований имеет такой формат, что игрокам нужно сыграть

минимум 2 сета и выиграть их. Но на некоторых турнирах (например, Уимблдон) теннисистам нужно провести 3 сета. Если счёт на 3-сетовых

матчах становится равным 1-1, то проводится третий сет для определения победителя. В рамках задания используется второй вариант.

**Описание решения**

Решение представлено следующим алгоритмом:

**Описание алгоритма выбора квадрата:**

1. **Начальная стратегия**:
   * Агент выбирает квадрат для подачи случайным образом в зонах E или F, в зависимости от типа подачи (первая подача или обычная).
   * При обычной подаче агент может выбрать случайный квадрат из зон D, E, или F.
2. **Тактики**:
   * **Рандомная тактика**: Агент случайно выбирает один из доступных квадратов в зависимости от типа подачи.
   * **Тактика дальнего квадрата**: Агент выбирает самую дальнюю точку от болванчика для подачи, чтобы минимизировать шансы болванчика на отбивание.
3. **Обработка промахов**:
   * С вероятностью 5% мяч может попасть в соседний квадрат. Причем, если квадрат у края корта, мяч может оказаться в ауте.

**Смоделированные параметры:**

* **r** - Радиус действия игрока, как агента, так и болванчика.
* **l** - Максимальное расстояние перемещения агента и болванчика между атаками.
* **n** - Количество квадратов на корте.

**Код программы**

public class Ball {  
  
 private Integer x; // при инициализации мяча у него нет координат  
 private Integer y;  
  
 public Ball() {  
 this.x = null;  
 this.y = null;  
 }  
  
 public void newLocation(int newX, int newY) {  
 this.x = newX;  
 this.y = newY;  
 }  
  
 public Integer getX() {  
 return x;  
 }  
  
 public Integer getY() {  
 return y;  
 }  
  
}

import java.util.HashMap;  
import java.util.Map;  
import java.util.stream.IntStream;  
  
public class Cort {  
 private int[][] matrix;  
 private Map<String, Zone> allZones;  
  
 public Cort(int numberOfSquares) {  
 int width = (int) Math.*sqrt*(numberOfSquares / 3);  
 this.matrix = new int[width][width \* 3];  
  
 // Creating the matrix   
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = 0;  
 }  
 }  
  
 this.allZones = divisionIntoZones(width \* 3, width);  
 }  
  
 private Map<String, Zone> divisionIntoZones(int length, int width) {  
 Zone a = new Zone(length, width, "A");  
 Zone b = new Zone(length, width, "B");  
 Zone c = new Zone(length, width, "C");  
 Zone d = new Zone(length, width, "D");  
 Zone e = new Zone(length, width, "E");  
 Zone f = new Zone(length, width, "F");  
  
 Map<String, Zone> zones = new HashMap<>();  
 zones.put("A", a);  
 zones.put("B", b);  
 zones.put("C", c);  
 zones.put("D", d);  
 zones.put("E", e);  
 zones.put("F", f);  
  
 return zones;  
 }  
  
 public void startPositions(Zone a, Zone d, Player player, Dummy dummy) {  
 int playerX = Math.*round*((a.getMinX() + a.getMaxX()) / 2);  
 int playerY = Math.*round*((a.getMinY() + a.getMaxY()) / 2);  
 player.setX(playerX);  
 player.setY(playerY);  
  
 int dummyX = d.getMinX();  
 int dummyY = Math.*round*((d.getMinY() + d.getMaxY()) / 2);  
 dummy.setX(dummyX);  
 dummy.setY(dummyY);  
 }  
  
 public void printCortState(Player player, Dummy dummy, Ball ball) {  
 System.*out*.println("Текущее состояние корта");  
  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 if (i == player.getY() && j == player.getX()) {  
 System.*out*.print("P ");  
 } else if (i == dummy.getY() && j == dummy.getX()) {  
 System.*out*.print("D ");  
 } else if (i == ball.getY() && j == ball.getX()) {  
 System.*out*.print("B ");  
 } else {  
 System.*out*.print("O ");  
 }  
 }  
 System.*out*.println();  
 }  
 }  
  
 public Map<String, Zone> getAllZones() {  
 return allZones;  
 }  
}

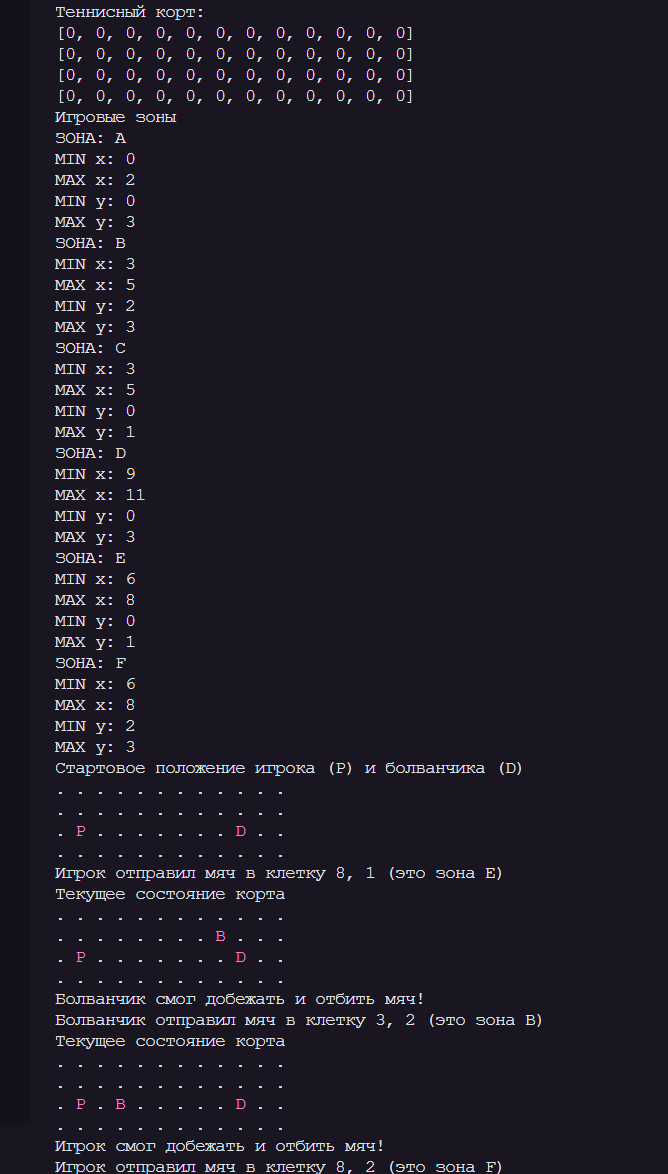
import java.util.HashMap;  
import java.util.Map;  
import java.util.Random;  
  
public class Dummy {  
 private int r;  
 private int l;  
 private Integer x; // координаты могут быть null  
 private Integer y;  
 private int score;  
 private int game;  
 private int set;  
 private int match;  
  
 public Dummy(int r, int l) {  
 this.r = r;  
 this.l = l;  
 this.x = null;  
 this.y = null;  
 this.score = 0;  
 this.game = 0;  
 this.set = 0;  
 this.match = 0;  
 }  
  
 public void pitch(Map<String, Zone> zones, Ball ball) {  
 Random rand = new Random();  
 int randZone = rand.nextInt(3) + 1;  
 Zone zone;  
 if (randZone == 1) {  
 zone = zones.get("A");  
 } else if (randZone == 2) {  
 zone = zones.get("B");  
 } else {  
 zone = zones.get("C");  
 }  
  
 int squareX = zone.getMinX() + rand.nextInt(zone.getMaxX() - zone.getMinX() + 1);  
 int squareY = zone.getMinY() + rand.nextInt(zone.getMaxY() - zone.getMinY() + 1);  
 ball.newLocation(squareX, squareY);  
 }  
  
 public boolean move(Ball ball) {  
 int ballX = ball.getX();  
 int ballY = ball.getY();  
 int numberOfMovements = this.l;  
  
 while ((this.y + this.r) < ballY || (this.y - this.r) > ballY) {  
 if (numberOfMovements > 0) {  
 if ((this.y + this.r) < ballY) {  
 this.y += 1;  
 } else if ((this.y - this.r) > ballY) {  
 this.y -= 1;  
 }  
 numberOfMovements--;  
 } else {  
 return false;  
 }  
 }  
  
 while ((this.x + this.r) < ballX || (this.x - this.r) > ballX) {  
 if (numberOfMovements > 0) {  
 if ((this.x + this.r) < ballX) {  
 this.x += 1;  
 } else if ((this.x - this.r) > ballX) {  
 this.x -= 1;  
 }  
 numberOfMovements--;  
 } else {  
 return false;  
 }  
 }  
  
 return true;  
 }  
  
 public int getR() {  
 return r;  
 }  
  
 public void setR(int r) {  
 this.r = r;  
 }  
  
 public int getL() {  
 return l;  
 }  
  
 public void setL(int l) {  
 this.l = l;  
 }  
  
 public Integer getX() {  
 return x;  
 }  
  
 public void setX(Integer x) {  
 this.x = x;  
 }  
  
 public Integer getY() {  
 return y;  
 }  
  
 public void setY(Integer y) {  
 this.y = y;  
 }  
  
 public int getScore() {  
 return score;  
 }  
  
 public void setScore(int score) {  
 this.score = score;  
 }  
  
 public int getGame() {  
 return game;  
 }  
  
 public void setGame(int game) {  
 this.game = game;  
 }  
  
 public int getSet() {  
 return set;  
 }  
  
 public void setSet(int set) {  
 this.set = set;  
 }  
  
 public int getMatch() {  
 return match;  
 }  
  
 public void setMatch(int match) {  
 this.match = match;  
 }  
  
}

import java.util.Random;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 private int n;  
 private int r;  
 private int l;  
  
 private String[] playerPitch;  
 private String playerTactic;  
  
 private Cort cort;  
 private Player player;  
 private Dummy dummy;  
 private Ball ball;  
  
 public Main(int n, int r, int l, String[] playerPitch, String playerTactic) {  
 this.n = n;  
 this.r = r;  
 this.l = l;  
 this.playerPitch = playerPitch;  
 this.playerTactic = playerTactic;  
 this.cort = new Cort(n);  
 this.player = new Player(r, l);  
 this.dummy = new Dummy(r, l);  
 this.ball = new Ball();  
 }  
  
 public boolean game() {  
 int playerScore = 0;  
 int dummyScore = 0;  
 cort.startPositions(cort.getAllZones().get("A"), cort.getAllZones().get("D"), player, dummy);  
  
 while (true) {  
  
 player.zoneSelection(cort.getAllZones(), ball, playerPitch[0], playerTactic, dummy);  
  
  
 while (true) {  
  
 boolean flagDummy = dummy.move(ball);  
  
 if (!flagDummy) {  
 playerScore++;  
  
 cort.startPositions(cort.getAllZones().get("A"), cort.getAllZones().get("D"), player, dummy);  
 break;   
 }  
  
  
 dummy.pitch(cort.getAllZones(), ball);  
 boolean flagPlayer = player.move(ball);  
  
 if (!flagPlayer) {  
 dummyScore++;  
  
 cort.startPositions(cort.getAllZones().get("A"), cort.getAllZones().get("D"), player, dummy);  
 break;   
 }  
  
  
 boolean miss = player.zoneSelection(cort.getAllZones(), ball, playerPitch[1], playerTactic, dummy); // subsequent serves have a chance to miss  
 if (miss) {  
 dummyScore++;  
  
 cort.startPositions(cort.getAllZones().get("A"), cort.getAllZones().get("D"), player, dummy);  
 break;  
 } else {  
  
 }  
   
 }  
  
   
 if ((playerScore >= 10 || dummyScore >= 10) && Math.*abs*(playerScore - dummyScore) >= 2) {  
 break;   
 }  
   
 }  
  
 if (playerScore > dummyScore) {  
   
 return true;  
 } else {  
   
 return false;  
 }  
 }  
  
 public static void main(String[] args) {  
 int[] nValues = {48, 192, 432, 768};   
 int[] rValues = {1, 2, 3, 4, 5};   
 int[] lValues = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};   
 String[] playerPitch = {"first", "default"};  
 String[] playerTactics = {"random", "far square"};  
  
 for (int n : nValues) {  
   
 int[][] playerWins = new int[rValues.length][lValues.length];  
  
 for (int r = 0; r < rValues.length; r++) {  
 for (int l = 0; l < lValues.length; l++) {  
 int playerTotalWin = 0;  
 int dummyTotalWin = 0;  
  
 for (int i = 0; i < 100; i++) {  
 int playerGameWin = 0;  
 int dummyGameWin = 0;  
 int playerSetWin = 0;  
 int dummySetWin = 0;  
  
 while (playerSetWin < 2 && dummySetWin < 2) {  
 playerGameWin = 0;  
 dummyGameWin = 0;  
  
 while (true) {  
 Main game = new Main(n, rValues[r], lValues[l], playerPitch, playerTactics[0]);  
 if (game.game()) {  
 playerGameWin++;  
 } else {  
 dummyGameWin++;  
 }  
  
 if ((playerGameWin >= 6 || dummyGameWin >= 6) && Math.*abs*(playerGameWin - dummyGameWin) >= 2) {  
 break;  
 }  
 }  
  
   
 if (playerGameWin > dummyGameWin) {  
 playerSetWin++;  
 } else {  
 dummySetWin++;  
 }  
 }  
  
 if (playerSetWin > dummySetWin) {  
 System.*out*.println("Игрок победил!");  
 playerTotalWin++;  
 } else {  
 System.*out*.println("Болванчик победил!");  
 dummyTotalWin++;  
 }  
 }  
  
  
 playerWins[r][l] = playerTotalWin;  
 }  
 }  
  
  
 System.*out*.println("Результаты для n = " + n);  
 for (int r = 0; r < rValues.length; r++) {  
 for (int l = 0; l < lValues.length; l++) {  
 System.*out*.println("r = " + rValues[r] + ", l = " + lValues[l] + ": " + playerWins[r][l]);  
 }  
 }  
  
  
 }  
  
 System.*out*.println("Закончили построение графиков для всех n значений.");  
 }  
}

import java.util.Map;  
import java.util.Random;  
  
public class Player {  
 private int r; // radius of view  
 private int l; // maximum distance of movement between attacks  
 private Integer x; // x-coordinate of the player  
 private Integer y; // y-coordinate of the player  
 private int score; // score of the player  
 private int game; // game number  
 private int set; // set number  
 private int match; // match number  
  
 public Player(int r, int l) {  
 this.r = 2 \* r;  
 this.l = l;  
 this.x = null; // no coordinates at initialization  
 this.y = null;  
 this.score = 0;  
 this.game = 0;  
 this.set = 0;  
 this.match = 0;  
 }  
  
 public boolean pitch(Zone zone, Ball ball, String message) {  
 Random random = new Random();  
 int squareX = random.nextInt(zone.getMaxX() - zone.getMinX() + 1) + zone.getMinX();  
 int squareY = random.nextInt(zone.getMaxY() - zone.getMinY() + 1) + zone.getMinY();  
 if (message.equals("Не промах")) {  
 ball.newLocation(squareX, squareY);  
 // print("Player sent the ball to square (" + ball.getX() + ", " + ball.getY() + ") in zone " + zone.getName());  
 return false;  
 } else {  
 int squareOfMissX = random.nextInt(3) - 1 + squareX;  
 int squareOfMissY = random.nextInt(3) - 1 + squareY;  
 while (squareOfMissX == squareX && squareOfMissY == squareY) {  
 squareOfMissX = random.nextInt(3) - 1 + squareX;  
 squareOfMissY = random.nextInt(3) - 1 + squareY;  
 }  
 // check if the ball is out  
 if ((squareOfMissY < zone.getMinY() && (zone.getName().equals("E") || zone.getName().equals("D"))) ||  
 (squareOfMissX > zone.getMaxX() && zone.getName().equals("D")) ||  
 (squareOfMissY > zone.getMaxY() && (zone.getName().equals("D") || zone.getName().equals("F")))) {  
 // print("Player missed and the ball is out (" + squareOfMissX + ", " + squareOfMissY + ")");  
 return true;  
 } else {  
 ball.newLocation(squareOfMissX, squareOfMissY);  
 // print("Player missed and the ball is in square (" + ball.getX() + ", " + ball.getY() + ")");  
 return false;  
 }  
 }  
 }  
  
 public boolean tacticPitch(int maxX, int maxY, Ball ball, String message, Map<String, Zone> dictOfZones) {  
 if (message.equals("Не промах")) {  
 ball.newLocation(maxX, maxY);  
 // print("Player sent the ball to square (" + ball.getX() + ", " + ball.getY() + ")");  
 return false;  
 } else {  
 Random random = new Random();  
 int squareOfMissX = random.nextInt(3) - 1 + maxX;  
 int squareOfMissY = random.nextInt(3) - 1 + maxY;  
 while (squareOfMissX == maxX && squareOfMissY == maxY) {  
 squareOfMissX = random.nextInt(3) - 1 + maxX;  
 squareOfMissY = random.nextInt(3) - 1 + maxY;  
 }  
 // check if the ball is out  
 Zone zoneD = dictOfZones.get("D");  
 Zone zoneE = dictOfZones.get("E");  
 if ((squareOfMissY < zoneE.getMinY()) || (squareOfMissX > zoneD.getMaxX()) || (squareOfMissY > zoneD.getMaxY())) {  
 // print("Player missed and the ball is out (" + squareOfMissX + ", " + squareOfMissY + ")");  
 return true;  
 } else {  
 ball.newLocation(squareOfMissX, squareOfMissY);  
 // print("Player missed and the ball is in square (" + ball.getX() + ", " + ball.getY() + ")");  
 return false;  
 }  
 }  
 }  
  
 public boolean zoneSelection(Map<String, Zone> dictOfZones, Ball ball, String flag, String tactic, Dummy dummy) {  
 Random random = new Random();  
 if (tactic.equals("random")) {  
 String message = "Не промах";  
 if (flag.equals("first")) {  
 int randZone = random.nextInt(2) + 1;  
 Zone zone = dictOfZones.get(randZone == 1 ? "E" : "F");  
 pitch(zone, ball, message);  
 } else if (flag.equals("default")) {  
 int randZone = random.nextInt(3) + 1;  
 Zone zone;  
 if (randZone == 1) {  
 zone = dictOfZones.get("D");  
 } else if (randZone == 2) {  
 zone = dictOfZones.get("E");  
 } else {  
 zone = dictOfZones.get("F");  
 }  
// check if the player missed  
 int probabilityOfMiss = random.nextInt(100);  
 if (probabilityOfMiss <= 5) {  
 message = "Промах";  
 }  
 pitch(zone, ball, message);  
 } else {  
 System.*out*.println("Invalid type of serve: " + flag);  
 throw new IllegalArgumentException();  
 }  
 } else if (tactic.equals("far square")) {  
 String message = "Не промах";  
 int dummyX = dummy.getX();  
 int dummyY = dummy.getY();  
 int maxX = 0;  
 int maxY = 0;  
  
 Zone zoneD = dictOfZones.get("D");  
 Zone zoneE = dictOfZones.get("E");  
 if (flag.equals("first")) {  
  
 if (Math.*abs*(dummyX - zoneE.getMinX()) >= Math.*abs*(dummyX - zoneE.getMaxX())) {  
 maxX = zoneE.getMinX();  
 } else {  
 maxX = zoneE.getMaxX();  
 }  
  
 if (Math.*abs*(dummyY - zoneD.getMinY()) >= Math.*abs*(dummyY - zoneD.getMaxY())) {  
 maxY = zoneD.getMinY();  
 } else {  
 maxY = zoneD.getMaxY();  
 }  
 tacticPitch(maxX, maxY, ball, message, dictOfZones);  
 } else if (flag.equals("default")) {  
  
 if (Math.*abs*(dummyX - zoneE.getMinX()) >= Math.*abs*(dummyX - zoneD.getMaxX())) {  
 maxX = zoneE.getMinX();  
 } else {  
 maxX = zoneD.getMaxX();  
 }  
  
 if (Math.*abs*(dummyY - zoneD.getMinY()) >= Math.*abs*(dummyY - zoneD.getMaxY())) {  
 maxY = zoneD.getMinY();  
 } else {  
 maxY = zoneD.getMaxY();  
 }  
  
 int probabilityOfMiss = random.nextInt(100);  
 if (probabilityOfMiss <= 5) {  
 message = "Промах";  
 }  
 tacticPitch(maxX, maxY, ball, message, dictOfZones);  
 }  
 }  
 return false;  
 }  
  
 public boolean move(Ball ball) {  
 int ballX = ball.getX();  
 int ballY = ball.getY();  
 int numberOfMovements = this.l;  
 if (ballX < this.x) {  
 while (this.x > ballX) {  
 if (numberOfMovements > 0) {  
 this.x -= 1;  
 numberOfMovements -= 1;  
 } else {  
 System.*out*.println("Игрок не смог дотянуться до мяча!");  
 return false;  
 }  
 }  
 while ((this.y + this.r) < ballY || (this.y - this.r) > ballY) {  
 if (numberOfMovements > 0) {  
 if ((this.y + this.r) < ballY) {  
 this.y += 1;  
 } else if ((this.y - this.r) > ballY) {  
 this.y -= 1;  
 }  
 numberOfMovements -= 1;  
 } else {  
 System.*out*.println("Игрок не смог дотянуться до мяча!");  
 return false;  
 }  
 }  
 } else if (ballX > this.x) {  
 while ((this.y + this.r) < ballY || (this.y - this.r) > ballY) {  
 if (numberOfMovements > 0) {  
 if ((this.y + this.r) < ballY) {  
 this.y += 1;  
 } else if ((this.y - this.r) > ballY) {  
 this.y -= 1;  
 }  
 numberOfMovements -= 1;  
 } else {  
 System.*out*.println("Игрок не смог дотянуться до мяча!");  
 return false;  
 }  
 }  
 while ((this.x + this.r) < ballX || (this.x - this.r) > ballX) {  
 if (numberOfMovements > 0) {  
 if ((this.x + this.r) < ballX) {  
 this.x += 1;  
 } else if ((this.x - this.r) > ballX) {  
 this.x -= 1;  
 }  
 numberOfMovements -= 1;  
 } else {  
 System.*out*.println("Игрок не смог дотянуться до мяча!");  
 return false;  
 }  
 }  
 }  
 System.*out*.println("Игрок добрался до мяча!");  
 return true;  
 }  
  
 public void setX(Integer x) {  
 this.x = x;  
 }  
  
 public void setY(Integer y) {  
 this.y = y;  
 }  
  
 public Integer getX() {  
 return x;  
 }  
  
 public Integer getY() {  
 return y;  
 }  
}

public class Zone {  
 private String name;  
 private int minX;  
 private int maxX;  
 private int minY;  
 private int maxY;  
  
 public Zone(int length, int width, String name) {  
 this.name = name;  
  
 switch (name) {  
 case "A":  
 this.minX = 0;  
 this.maxX = (length / 4) - 1;  
 this.minY = 0;  
 this.maxY = width - 1;  
 break;  
  
 case "B":  
 this.minX = length / 4;  
 this.maxX = (length / 2) - 1;  
 this.minY = width / 2;  
 this.maxY = width - 1;  
 break;  
  
 case "C":  
 this.minX = length / 4;  
 this.maxX = (length / 2) - 1;  
 this.minY = 0;  
 this.maxY = (width / 2) - 1;  
 break;  
  
 case "D":  
 this.minX = (length / 4) \* 3;  
 this.maxX = length - 1;  
 this.minY = 0;  
 this.maxY = width - 1;  
 break;  
  
 case "E":  
 this.minX = length / 2;  
 this.maxX = ((length / 4) \* 3) - 1;  
 this.minY = 0;  
 this.maxY = (width / 2) - 1;  
 break;  
  
 case "F":  
 this.minX = length / 2;  
 this.maxX = ((length / 4) \* 3) - 1;  
 this.minY = width / 2;  
 this.maxY = width - 1;  
 break;  
  
 default:  
 System.*out*.println("Получено некорректное имя зоны: " + name);  
 throw new IllegalArgumentException("Недопустимое имя зоны: " + name);  
 }  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getMinX() {  
 return minX;  
 }  
  
 public int getMaxX() {  
 return maxX;  
 }  
  
 public int getMinY() {  
 return minY;  
 }  
  
 public int getMaxY() {  
 return maxY;  
 }  
}

**Примеры вывода**

****

