

# RAPORT z snake'a w Java

Temat: **Java Snake**

Adam Badeński

Numer indexu: 262310

grupa wtorek 18:55

## 1. Opis

Zadaniem było napisanie gry snake wykorzystując, watki. Gra miała zawierać niezbędne elementy, ruchomy owoc, owoce , dwóch graczy ai, gracza głównego menu, i histore wyników.

## 2. Wykonanie i plan

Wykonałem powyższe zadanie jednak ograniczyłem sterowanie watkami do kybierania kierunku.

Dodatkowo w poniższym kodzie próbowałem niestety bezskutecznie zaimplementować synchronizację- jest to watek uruchamiany i działający ciągle oraz co 100ms ustawiający odpowiednio kierunek ruchu weza.

```
private void runAI(List<Point> snakeAI, boolean isFirstAI){  
  
    while (true) {  
  
        synchronized (lock){  
  
            if(isFirstAI){  
  
                while (que != 1) {  
  
                    try { lock.wait(); } catch (InterruptedException e) {}  
  
                }  
  
            }  
  
        }  
  
    }  
  
}
```

```

    }

    if(!isFirstAI){

        while (que != 2) {

            try { lock.wait(); } catch (InterruptedException e) {}

        }

    }

}

try {

    Thread.sleep(100);

} catch (InterruptedException ignored) {}

Point head = snakeAI.get(0);

List<Character> possibleDirs = new ArrayList<>(List.of('U','D','L','R'));

// Odrzucamy kierunek odwrotny

if (isFirstAI) {

    if (directionAI1 == 'U') possibleDirs.remove((Character) 'D');

    if (directionAI1 == 'D') possibleDirs.remove((Character) 'U');

    if (directionAI1 == 'L') possibleDirs.remove((Character) 'R');

    if (directionAI1 == 'R') possibleDirs.remove((Character) 'L');

} else {

    if (directionAI2 == 'U') possibleDirs.remove((Character) 'D');

    if (directionAI2 == 'D') possibleDirs.remove((Character) 'U');

    if (directionAI2 == 'L') possibleDirs.remove((Character) 'R');

    if (directionAI2 == 'R') possibleDirs.remove((Character) 'L');

}

// Logika ustawiania nowego kierunku

if (food != null) {

    possibleDirs.sort(((dir1, dir2) -> {

        Point next1 = nextPoint(head, dir1);

        Point next2 = nextPoint(head, dir2);

        int distance1 = Math.abs(next1.x - food.x) + Math.abs(next1.y - food.y);

        int distance2 = Math.abs(next2.x - food.x) + Math.abs(next2.y - food.y);

        return Integer.compare(distance1, distance2);

    }));

}

for (char candidate : possibleDirs) {

    Point nextHead = nextPoint(head, candidate);

    if (!isPointCollision(nextHead, snakeAI, snake, snakeAI1, snakeAI2) && !obstacles.contains(nextHead)) {

        if (!isFirstAI) {

            directionAI1 = candidate;

        } else {

            directionAI2 = candidate;

        }

        break;
    }
}

```

```
}  
}
```

```
if (isFirstAI) {  
  
    if(!game_over){  
  
        moveSnake(snakeAI, directionAI1);  
  
        //que=2;  
  
        if ((isCollision(snakeAI) || isCollisionWithOther(snakeAI, snake) || isCollisionWithOther(snakeAI, snakeAI2) || isCollisionWithObstacles(snakeAI)) {  
  
            message = "Wygrales";  
  
            status = message;  
  
            game_over=true;  
  
            saveScores();  
  
            repaint();  
  
            resetGame();  
  
            continue;  
  
        }  
  
    }  
  
    try {  
  
        Thread.sleep(1000);  
  
    } catch (InterruptedException ignored) {}  
  
    que = 2;  
  
    lock.notifyAll();  
  
    /*if (consumelfNeeded(snakeAI)) {  
  
        ai1Score++;  
  
    }*/  
  
    } else {  
  
        if(!game_over){  
  
            moveSnake(snakeAI, directionAI2);  
  
            //que=0;  
  
            if ((isCollision(snakeAI) || isCollisionWithOther(snakeAI, snake) || isCollisionWithOther(snakeAI, snakeAI1) || isCollisionWithObstacles(snakeAI)) {  
  
                message = "Wygrales";  
  
                status = message;  
  
                game_over=true;  
  
                saveScores();  
  
                repaint();  
  
                resetGame();  
  
                continue;  
  
            }  
  
        }  
  
        try {  
  
            Thread.sleep(1000);  
  
        } catch (InterruptedException ignored) {}  
  
        que = 0;  
  
        lock.notifyAll();  
  
        /*if (consumelfNeeded(snakeAI)) {  
  
            ai2Score++;  

```

3.

## 4 Wnioski

Gra prezentuje wymagane funkcjonalności