Урок 4

Крестики-нолики в процедурном стиле

Пишем крестики-нолики в процедурном стиле

.

Приступаем к написанию программы

1. **Создание игрового поля.** Игровое поле можно представить в виде символьного двумерного массива размером 3x3. В каждой ячейке массива могут находиться символы 'X', 'O', '•', которые означают фишки игроков и пустое поле. Для указания размера поля и количества фишек для победы добавим две константы *SIZE* и *DOTS_TO_WIN*.

```
public static char[][] map;
public static final int SIZE = 3;
public static final int DOTS_TO_WIN = 3;
```

2. **Ячейки поля.** Чтобы не забыть, какой символ за что отвечает, их можно записать в именные константы.

```
public static final char DOT_EMPTY = '•';
public static final char DOT_X = 'X';
public static final char DOT_O = 'O';
```

3. **Инициализация поля.** При запуске программы необходимо инициализировать поле и заполнить все его ячейки символом, обозначающим пустое поле.

```
public static void initMap() {
    map = new char[SIZE][SIZE];
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            map[i][j] = DOT_EMPTY;
        }
    }
}
```

Заполнение происходит за счёт того, что двойной цикл «пробегает» по всем ячейкам массива и в каждую ячейку записывает символ *DOT_EMPTY*.

4. **Вывод поля в консоль.** Проинициализированное поле готово. Теперь необходимо вывести его в консоль.

```
public static void printMap() {
    for (int i = 0; i <= SIZE; i++) {
        System.out.print(i + " ");
    }
    System.out.println();
    for (int i = 0; i < SIZE; i++) {
        System.out.print((i + 1) + " ");
        for (int j = 0; j < SIZE; j++) {
            System.out.print(map[i][j] + " ");
        }
        System.out.println();
    }
    System.out.println();
}
```

.

Первый цикл i отвечает за распечатку шапки игрового поля. Далее идёт двойной цикл i, j, внутренний j цикл отвечает за печать одной строки, после того как одна строка отпечаталась в консоль (закончился цикл j), происходит перевод каретки на следующую строку методом System.out.println(). После этого счетчик i увеличивается и производится печать следующих строк. После распечатки всего поля ставится дополнительная пустая строка для разделения выводов полей.

5. **Тест программы.** Можно запускать первый тест программы.

```
public static void main(String[] args) {
    initMap();
    printMap();
}
```

Произойдёт инициализация и вывод поля в консоль. Программа завершит свою работу.

6. **Ход человека.** Теперь можно сделать метод, отвечающий за ход человека-игрока.

```
public static Scanner sc = new Scanner(System.in);
public static void humanTurn() {
    int x, y;
    do {
        System.out.println("Введите координаты в формате X Y");
        x = sc.nextInt() - 1;
        y = sc.nextInt() - 1;
    } while (!isCellValid(x, y));
    map[y][x] = DOT_X;
}
```

Итак, что же нужно для хода игрока. Во-первых, создать Scanner для считывания данных из консоли. Во-вторых, перед ходом игрока вывести сообщение о запросе координат. В-третьих, запросить координаты и уменьшить их на 1, так как для игрока удобно вводить координаты в формате 1-3, при том что в массиве это индексы 0-2. После ввода проверить, можно ли поставить фишку в указанную ячейку. Если ячейка свободна и существует, то поставить крестик.

7. **Проверка ячеек.** Метод isCellValid() проверяет возможность установки фишки в указанную ячейку.

```
public static boolean isCellValid(int x, int y) {
    if (x < 0 || x >= SIZE || y < 0 || y >= SIZE) return false;
    if (map[y][x] == DOT_EMPTY) return true;
    return false;
}
```

Если введены неверные координаты (либо указывают на занятое поле, либо находятся за его пределами), возвращаем false, в противном случае – true.

8. **Ход компьютера.** Для реализации хода компьютера достаточно чуть изменить метод хода игрока, добавив туда генератор случайных чисел.

```
public static Random rand = new Random();
public static void aiTurn() {
    int x, y;
        do {
```

.

```
            x = rand.nextInt(SIZE);
            y = rand.nextInt(SIZE);
        } while (!isCellValid(x, y));
    System.out.println("Компьютер походил в точку " + (x + 1) + " " + (y + 1));
    map[y][x] = DOT_O;
}
```

Метод rand.nextInt(n) возвращает случайное число в диапазоне от 0 до n – 1 включительно.

9. **Проверка победы.** Необходимо проверить все выигрышные комбинации. Если хотя бы одна будет найдена, вернуть true. В приведённом ниже варианте производится проверка всех возможных комбинаций с помощью 8 условий, для поля 3x3 такой подход может быть оправдан. Если размер поля или длину выигрышной серии можно будет увеличить, лучше найти другой вариант проверки победы, например, через циклы(для поля 4x4 придется написать уже не 8, а 24 условия).

```
public static boolean checkWin(char symb) {
    if(map[0][0] == symb && map[0][1] == symb && map[0][2] == symb) return true;
    if(map[1][0] == symb && map[1][1] == symb && map[1][2] == symb) return true;
    if(map[2][0] == symb && map[2][1] == symb && map[2][2] == symb) return true;
    if(map[0][0] == symb && map[1][0] == symb && map[2][0] == symb) return true;
    if(map[0][1] == symb && map[1][1] == symb && map[2][1] == symb) return true;
    if(map[0][2] == symb && map[1][2] == symb && map[2][2] == symb) return true;
    if(map[0][0] == symb && map[1][1] == symb && map[2][2] == symb) return true;
    if(map[2][0] == symb && map[1][1] == symb && map[0][2] == symb) return true;
    return false;
}
```

10. **Основной игровой цикл.** Игровой цикл состоит из последовательного вызова ходов человек-компьютер-человек-компьютер-…до тех пор, пока не заполнится игровое поле или не выиграет один из участников.

```
public static void main(String[] args) {
    initMap();
    printMap();
    while (true) {
        humanTurn();
        printMap();
        if (checkWin(DOT_X)) {
            System.out.println("Победил человек");
            break;
        }
        if (isMapFull()) {
            System.out.println("Ничья");
            break;
        }
        aiTurn();
        printMap();
        if (checkWin(DOT_O)) {
            System.out.println("Победил Искуственный Интеллект");
            break;
        }
        if (isMapFull()) {
            System.out.println("Ничья");
            break;
        }
```

.

```
    }
    System.out.println("Игра закончена");
}
```

Полный листинг программы

```
import java.util.Random;
import java.util.Scanner;
public class MainClass {
    public static int SIZE = 3;
    public static int DOTS_TO_WIN = 3;
    public static final char DOT_EMPTY = '•';
    public static final char DOT_X = 'X';
    public static final char DOT_O = 'O';
    public static char[][] map;
    public static Scanner sc = new Scanner(System.in);
    public static Random rand = new Random();
    public static void main(String[] args) {
        initMap();
        printMap();
        while (true) {
            humanTurn();
            printMap();
            if (checkWin(DOT_X)) {
                System.out.println("Победил человек");
                break;
            }
            if (isMapFull()) {
                System.out.println("Ничья");
                break;
            }
            aiTurn();
            printMap();
            if (checkWin(DOT_O)) {
                System.out.println("Победил Искуственный Интеллект");
                break;
            }
            if (isMapFull()) {
                System.out.println("Ничья");
                break;
            }
        }
        System.out.println("Игра закончена");
    }
    public static boolean checkWin(char symb) {
        if(map[0][0] == symb && map[0][1] == symb && map[0][2] == symb) return
true;
        if(map[1][0] == symb && map[1][1] == symb && map[1][2] == symb) return
true;
        if (map[2][0] == symb && map[2][1] == symb && map[2][2] == symb) return
true;
        if (map[0][0] == symb && map[1][0] == symb && map[2][0] == symb) return
true;
        if (map[0][1] == symb && map[1][1] == symb && map[2][1] == symb) return
true;
        if (map[0][2] == symb && map[1][2] == symb && map[2][2] == symb) return
```

.

```
true;
        if (map[0][0] == symb && map[1][1] == symb && map[2][2] == symb) return
true;
        if (map[2][0] == symb && map[1][1] == symb && map[0][2] == symb) return
true;
        return false;
    }
    public static boolean isMapFull() {
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                if (map[i][j] == DOT_EMPTY) return false;
            }
        }
        return true;
    }
    public static void aiTurn() {
        int x, y;
        do {
            x = rand.nextInt(SIZE);
            y = rand.nextInt(SIZE);
        } while (!isCellValid(x, y));
        System.out.println("Компьютер походил в точку " + (x + 1) + " " + (y +
1));
        map[y][x] = DOT_O;
    }
    public static void humanTurn() {
        int x, y;
        do {
            System.out.println("Введите координаты в формате X Y");
            x = sc.nextInt() - 1;
            y = sc.nextInt() - 1;
        } while (!isCellValid(x, y)); // while(isCellValid(x, y) == false)
        map[y][x] = DOT_X;
    }
    public static boolean isCellValid(int x, int y) {
        if (x < 0 || x >= SIZE || y < 0 || y >= SIZE) return false;
        if (map[y][x] == DOT_EMPTY) return true;
        return false;
    }
    public static void initMap() {
        map = new char[SIZE][SIZE];
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) {
                map[i][j] = DOT_EMPTY;
            }
        }
    }
    public static void printMap() {
        for (int i = 0; i <= SIZE; i++) {
            System.out.print(i + " ");
        }
        System.out.println();
        for (int i = 0; i < SIZE; i++) {
            System.out.print((i + 1) + " ");
            for (int j = 0; j < SIZE; j++) {
                System.out.print(map[i][j] + " ");
            }
            System.out.println();
        }
```

.

```
            System.out.println();
        }
}
```

Домашнее задание

1. Полностью разобраться с кодом, попробовать переписать с нуля, стараясь не подглядывать в методичку.
2. Переделать проверку победы, чтобы она не была реализована просто набором условий, например, с использованием циклов.
3. * Попробовать переписать логику проверки победы, чтобы она работала для поля 5х5 и количества фишек 4. Очень желательно не делать это просто набором условий для каждой из возможных ситуаций;
4. *** Доработать искусственный интеллект, чтобы он мог блокировать ходы игрока.

Дополнительные материалы

1. Вебинар Простая консольная игра на Java "Битвы с монстрами": https://geekbrains.ru/events/2185
2. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала. Том 1. Основы // Пер. с англ. – М.: Вильямс, 2014. – 864 с.
3. Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.
4. Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 1376 с.
5. Г. Шилдт Java 8: Руководство для начинающих. // 6-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 720 с.

Используемая литература

1. Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.
2. Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 1376 с.
3. Г. Шилдт Java 8: Руководство для начинающих. // 6-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 720 с.