Dokumentation

	Matrikelnummer
Lara Sauer	1687998
Joel Oswald	2957340
Phillip Biedermann	8302847
Johanna Schuster	5725613
Lucas- Maurice Stein	

Programmierung
ON22B
Prof. Dr. rer.nat. Alexander Auch

Projektidee

Unser **Spielprinzip**

Das Spiel, dass wir Programmieren heißt TicTacToe, es wird auf einer 4x4 Spielfläche gespielt, nicht wie üblich auf 3x3. Das Prinzip bei unserem Spiel ist genau dasselbe, wie bei einem normalen TicTacToe-Spiel. Jedoch gewinnt mwan bei unserer Variante, wenn man vier in einer Reihe hat. Dadurch kann ein Spiel länger dauern und ist ein wenig anspruchsvoller, als die 3x3 Variante.

Es treten immer zwei Spieler gegeneinander an. Diese beiden Spieler kommen abwechselnd zum Zug und dürfen ihr Symbol auf eines der sechzehn Felder setzen. Der erste Spieler spielt mit dem Zeichen x und er zweite mit o. Um das jeweilige Zeichen in das gewünschte Feld zu setzen, muss man die Nummer für dieses Feldes angeben. Das wird so lange gemacht, bis der erste Spieler vier Zeichen in einer Reihe, Spalte oder Diagonale gesetzt hat. Wenn beide der Spieler optimal gespielt haben und es keine freien Felder mehr gibt, ohne dass jemand der beiden Teilnehmer eine vierer-Reihe hat, gilt das Spiel als Unentschieden.

Für falsche Eingaben, soll immer eine kurze Meldung ausgegeben werden. Die jeweiligen Anweisungen zu jedem Spielzug werden auf der Konsole in deutsch ausgegeben.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Umsetzung

Erste **Gedanken**

Bei der Umsetzung mussten wir uns zuerst umorientieren, da wir unsere erste Idee ein Snake-Spiel zu programmieren verworfen haben, weil es zu schwer war es umzusetzen. Bevor wir mit begonnen haben, den Code zu schreiben haben wir uns erst einmal intensiv mit dem Spiel auseinandergesetzt. Zunächst benötigen wir eine Methode, die festlegt, wann ein Spieler gewinnt und wann es unetschieden steht. Dann brauchen wir eine Ausgabe für das Spielfeld und noch aller Spielanweisungen. Außerdem dürfen besetzte Felder nicht erneut ausgewählt werden.

Unser **Code**

Zunächst haben wir eine Klasse TicTacToe erstellt. Danach haben wir mit der Umsetzung der Methode checkWinner angefangen.

```
static String checkWinner()
{
    for (int a = 0; a < 16; a++)
    {
        String line = "0";

        switch (a)
        {
        case 0:
            line = board[0] + board[1] + board[2] + board[3];
            break;
        case 1:
            line = board[4] + board[5] + board[6] + board[7];
            break;
        case 2:
            line = board[8] + board[9] + board[10] + board[11];
            break;
        case 3:
            line = board[12] + board[13] + board[14] + board[15];
            break;
        case 4:
            line = board[0] + board[4] + board[8] + board[12];
            break;
        case 5:
            line = board[1] + board[5] + board[9] + board[13];
            break;
        case 6:
            line = board[2] + board[6] + board[10] + board[14];
            break;
        case 7:
            line = board[3] + board[7] + board[11] + board[15];
            break;
        case 8:
            line = board[0] + board[5] + board[10] + board[15];
            break;
        case 9:
            line = board[3] + board[6] + board[9] + board[12];
            break;
```

Hier haben wir eine for-Schleife benutzt und in dieser eine switch, um zunächst alle einzelnen Möglichkeiten zu gewinnen festzulegen. Für unser Spielfeld nutzen wir boards von 0 bis 15. Gewinnmöglichkeiten sind Reihen, Spalten und Diagonalen was bei einem 4x4 Feld 9 verschiedene sind.

```
line = line.replaceAll("\\s+", "");

if (line.equals("XXXX"))
{
    return "X";
}
else if (line.equals("0000"))
{
    return "O";
}
```

Danach wird noch geprüft, wer der Gewinner ist. Hier hatten wir die Schwierigkeit, dass wir beim Ausgeben des Spielfeldes Leerzeichen bei einzelnen Symbolen hinzufügen mussten (siehe S.6f.) und so durch die if-Schleife nicht mehr mit einbezogen wurden. Deshalb haben wir die erste Zeile im Bild eingefügt.

```
for (int a = 0; a < 16; a++)
{
    if (Arrays.asList(board).contains(String.valueOf(a + 1)))
    {
        break;
    }
    else if (a == 16)
    {
        return "Unentschieden";
```

Für ein Unentschieden haben wir ebenfalls mit einer if-Schleife gearbeitet. Diese gibt ein Unentschieden aus, wenn das Spielfeld voll ist, also in jedes Feld ein Symbol eingetragen wurde, und kein Gewinner feststeht.

```java
static void printBoard()
{
    System.out.println("|-----|-----|-----|-----|");
    System.out.println("|  " + board[0] + "  |  " + board[1] + "  |  " + board[2] + "  |  " + board[3] + "  | ");

    System.out.println("|----------------------|");
    System.out.println("|  " + board[4] + "  |  " + board[5] + "  |  " + board[6] + "  |  " + board[7] + "  | ");

    System.out.println("|----------------------|");
    System.out.println("|  " + board[8] + "  |  " + board[9] + " |  " + board[10] + " |  " + board[11] + " | ");

    System.out.println("|----------------------|");
    System.out.println("|  " + board[12] + " |  " + board[13] + " |  " + board[14] + " |  " + board[15] + " | ");
    System.out.println("|-----|-----|-----|-----|");
}
```

Um das Spielfeld auszugeben haben wir die Methode printBoard genutzt. Mithilfe von Bindestrichen wird die Tabelle erstellt. Die einzelnen boards wurden dazwischen eingefügt. Folgenden Vorgedanke haben wir hierfür genutzt:

```
|---|---|---|---|
| 1 | 2 | 3 | 4 |
|---------------|
| 4 | 5 | 6 | 7 |
|---------------|
| 7 | 8 | 9 | 10|
|---------------|
| 11| 12| 13| 14|
|---|---|---|---|
```

In der main Methode haben wir zunächst einen Scanner eingefügt und die Nummerierung der Felder von 1 bis 16 eingestellt. Außerdem wird der erste Zug an X vergeben und es gibt noch keinen Gewinner. Diese Methode gibt den Spielsablauf wieder. In der for-Schleife werden die einzelnen Züge dargestellt.

```java
public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    board = new String[16];
    turn = "X";
    String winner = null;

    for (int a = 0; a < 16; a++)
    {
        board[a] = String.valueOf(a + 1);
    }
```

```
while (winner == null)
{
    int numInput;

    try
    {
        numInput = in.nextInt();
        if (!(numInput > 0 && numInput <= 16))
        {
            System.out.println("Ups, dass kann nicht sein; entscheide weiße:");
            continue;
        }
    }
```

Hier ist der Fall dargestellt, dass eine zu kleine oder eine zu große Zahl angegeben wird. Mithilfe von try und einer for-Schleife wird dann eine kurze Anweiszung ausgegeben und der Spieler kann eine neue Eingabe machen.

```
if (numInput > 9 && turn == "X")
{
    turn = "X ";
}
if (numInput > 9 && turn == "O")
{
    turn = "O ";
}
if (board[numInput - 1].equals(String.valueOf(numInput)))
{
    board[numInput - 1] = turn;

    if (turn.equals("X") || turn.equals("X "))
    {
        turn = "O";
    }
    else
    {
        turn = "X";
    }
```

Hier tritt der Fall ein, bei dem wir das Problem mit den Leerzeichen gelöst haben. Ab dem zehnten Kästchen wird dann zusätzlich noch ein Leerzeichn mit ausgegeben, sodass sich das Spielfeld nicht verschiebt. Denn davor war es so, dass eine zweistellige Zahl durch das Symbol ersetzt wird, das nur eine Stelle hat.

Nach jeder Eingabe wird das Spielfeld neu aktualisiert ausgegeben und es wird geprüft, ob es einen Gewinner gibt.
Wenn das Feld schon belegt ist, wird ebenfalls ein kurzer Hinweis ausgegeben und einen neue Eingabe ermöglicht.
Wenn das Spiel dann entweder unentschieden steht oder es einen Gewinner gibt wird eine dementsprechende Meldung ausgegeben.

Probleme

Das erste Problem, das wir hatten, wurde bereits angesprochen. Wir haben tatsächlich längere Zeit gebraucht, um auf die Lösungsidee zu kommen, die das Problem mit den Leerzeichen behebt.
Immer wieder während des Arbeitsprozesses hat auf es auf einmal nicht mehr funktioniert, dass bei einem Unentschieden das Spiel aufhört. Das konnten wir immer wieder lösen. Eigentlich immer waren der Auslöser dafür kleine Änderungen am Programm, um das erste Problem zu lösen.
Zwischenzeitlich hat es auch nicht funktioniert, dass das Spiel endet wenn es einen Gewinner gibt. Das konnten eir letztendlich auch beheben.

Aktivitäten

Code:
Phillip Biedermann (50%), Johanna Schuster(50%), Joel Oswald(0%),
Lara Sauer(0%), Lucas Stein(0%)
Dokumentation:
Joel Oswald (50%), Lara Sauer (50%), Phillip Biedermann (0%),
Johanna Schuster(0%), Lucas Stein(0%)
Präsentation:
Philip Biedermann(25%), Johanna Schuster(25%), Joel Oswald(25%),
Lara Sauer(25%), Lucas Stein(0%)