0. Presentación del Informe

Título del Informe:

Tarea Obligatoria Nro. 2: TA-TE-TI.

Autor:

Rodrigo Ciosek.

Materia / Asignatura:

Algoritmos y Estructuras de Datos 1.

Profesor:

Carlos Mascheroni.

Fecha de entrega:

13 de mayo de 2025

1. Introducción

Se presenta el desarrollo del juego tateti, implementado mediante una matriz de 3×3 de tipo char. Dos jugadores interactúan por turnos utilizando los caracteres ‘X’ y ‘O’ para marcar sus jugadas, mientras que las posiciones vacías se representan con el carácter ‘-‘.
El objetivo principal del juego es que uno de los jugadores logre alinear tres caracteres propias en forma horizontal, vertical o diagonal. Se emplean condiciones adicionales para finalizar la partida: el abandono de uno de los jugadores donde ganaría el oponente, o la ocupación completa del tablero sin un ganador, termina en empate.
El programa debe solicitar al jugador la posición deseada para colocar su marca, validando que sea una posición válida y sin ocupar. Cada jugada válida, se muestra en pantalla el tablero actual para mantener el seguimiento visual de la partida.

2. Análisis

2.1 Definición y entendimiento del problema

Crear un juego tateti, en donde:

1. Crear una interfaz gráfica por consola interactiva con el usuario, donde se le solicite una opción, si salir o jugar.
2. En caso que seleccione jugar se mostrara un tablero 3x3 con las posiciones vacías representadas con '-'.
3. Se le solicitara al usuario ingresar dos datos numéricos que representen la fila y la columna o darle la oportunidad de salir.
4. Si es valido se ingresa en el tablero, y se muestra nuevamente con el cambio actual, y el turno ahora será para el jugador 2.
5. Si el jugador activo concreta tres caracteres consecutivos, ya sea horizontal, vertical o diagonal, se le concede la victoria y se termina el juego
6. En caso que se complete el tablero sin haber concretado 3 caracteres consecutivos, se concede el empate.

2.2 Entradas y salidas:

2.2.1 Entradas:

- Función Pedir Jugada
  - Un **valor booleano "turnoPlayer1"**
  - Un **valor numérico** que representa la fila
  - Un **valor numérico** que representa la columna
  - Un **carácter** por si desea abandonar el juego (opcional)

- Función Validar filas y columnas
  - Un **valor entero "fila"** que representa el índice de fila.
  - Un **valor entero "columna"** que representa el índice de columna.

- Función Validar Jugada
  - Un **array de enteros "jugada"** que contiene el índice de fila y columna**.**
- Función Hubo Empate
  - No recibe ninguna entrada**.**
- Función Gano Jugador
  - Un **char "letra"** que representa el carácter activo.

2.2.2 Salidas:

1. Función Pedir Jugada
   - Mensaje del jugador activo
   - Mensaje para **ingresar índice de Fila**
   - Mensaje para **ingresar índice de Columna**
   - Mensaje para **ingresar carácter** en caso de querer abandonar
   - Retorno de un **Array con los valores Fila y Columna**

2. Función Validar filas y columnas
   - Retorna un **booleano** que evalúa si los valores pasados por parámetros están dentro del rango de la matriz del tateti.

3. Función Validar jugada
   - Retorna un **booleano** que evalúa si los valores pasados por parámetros, que representa la ubicación, se encuentra dentro de los rangos de la matriz tateti y valida también se encuentra vacía “-“.

4. Función Hubo Empate
   - Retorna un **booleano** que evalúa si todavía se encuentra espacios en blancos dentro de la matriz tateti.

5. Función Gano Jugador
   Retorna un **booleano** que evalúa si el jugador con su carácter propio a concretado 3 caracteres consecutivos, ya sea en horizontal, vertical o en diagonal.

2.3 Pre y Post condiciones

2.3.1 Pre condiciones

1. Función inicio()
    - Ninguna, se muestra el menú principal.

2. Función juego()
    - Debe existir la **matriz "tateti"** y estar correctamente inicializada.

3. Función mostrarJuego().
    - La **matriz "tateti"** tiene que estar definida.

4. Función pedirJugada(boolean turnoPlayer1).
    - Debe existir un **Scanner** valido y el jugador debe poder ingresar valores por consola.

5 . Función validarFilasColuma(int fila, int columna).

    Los parámetros **"fila"** y **"columna"** deben de ser enteros.

6. Función validarJugada(int[] jugada)
    - **"jugada"** debe ser un **array de dos enteros**.

7. Función huboEmpate()
    - Debe existir **la matriz "tateti"** y estar definida.

8. Función insertarJugada(int fila, int columna, char letra)
    - La posición **"fila", "columna"**, debe estar dentro del rango de la matriz y estar vacía (**"-"**).

9. Función ganoJugador(char letra)
    - Debe existir la **matriz tateti** y **"letra"** debe ser **'X'** o **'O'.**

10. Función darVerdicto(char letra)
    - "letra" debe ser **'X', 'O'** o **'-**'.

2.3.2 Post condiciones

1. Función inicio()
    - Si el jugador selecciona opción 1, ira a jugar, si selecciona opción 2 sale del programa.

2. Función juego()
    - El juego termina con un ganador, empate y se imprime el resultado final, si un jugador decide salirse se cierra el programa.

3. Función mostrarJuego()
    - Imprime la **matriz tateti** en el estado actual.

4. Función pedirJugada(boolean turnoPlayer1)
    - Retorna un array de dos números enteros ingresados previamente por el jugador para colocar su marca en el tablero, tiene la opcion de ingresar ‘s’ para salir del programa.

5. Funcion validarFilasColumna(int fila, int columna)
    - Retorna **true** si están dentro del rango del tablero (0 a 2), **false** si no se cumple.

5. Función validarJugada(int[] jugada)
    - Retorna **true** si la fila **“jugada[0]”** y columna **“jugada[1]”** esta dentro del rango de la matriz **“tateti”** y si se encuentra vacía (**“-“**), **false** si no cumple.

6. Función huboEmpate()
- Retorna **true** si no quedan espacios vacíos (**‘-‘**), lo que implica un empate, **false** si aún hay jugadas posibles.

7. Función insertarJugada(int fila, int columna, char letra)
- Coloca el carácter **“letra”** en la posición especificada de la **matriz “tateti”**.

8. Función ganoJugador(char letra)
- Retorna **true** si el jugador correspondiente a **“letra”** ha ganado (3 en línea o diagonal), **false** si no.

9. Función darVeredicto(char letra)
    - Imprime en pantalla si ganó Player1 **‘X’**, Player2 **‘O’** o si hubo empate **‘-‘**.

3. Diseño

3.2 Lenguaje Natural

El programa comienza mostrando un menú con dos opciones: **jugar** o **salir**. Mientras el usuario no seleccione una opción válida, el programa sigue solicitando la entrada. Si elige **jugar**, se inicia el ciclo principal del juego.

Una vez que inicia la partida, se establece que es el turno del jugador 1 y se entra en un bucle que se repite hasta que el juego termina. En cada turno, se muestra el estado actual del tablero en pantalla y se solicita al jugador su jugada, la fila y columna donde desea colocar su marca. Esta jugada se convierte a un índice de la matriz (restando 1 para ajustarse al índice que comienza en 0) y se valida que esté dentro del rango del tablero y que la celda esté vacía. Si no es válida, se vuelve a pedir la jugada.

Una vez validada la jugada, se coloca la marca correspondiente en la matriz: 'X' para el jugador 1 o 'O' para el jugador 2. Después, el programa verifica si esa jugada hizo ganar al jugador actual, comprobando filas, columnas y diagonales. Si no hay ganador, también se verifica si el tablero está completo, en este caso se declara empate.

Si alguno de estos dos casos se cumple (victoria o empate), el juego termina y se muestra un mensaje en pantalla indicando quién ganó o si hubo empate. Luego, se reinicia el menú principal.

Además, el programa permite que un jugador abandone en cualquier momento escribiendo la letra 's', lo que hace que el programa finalice. Las funciones auxiliares como mostrar el tablero, validar jugadas, insertar marcas y determinar ganadores están organizadas en métodos separados para mejorar la claridad.

4. Implementación

Tateti.java

```java
import java.util.Scanner;

public class Tateti {  1 usage  Rodrigo *

    private static Boolean turnoPlayer1 = true;  5 usages
    private static Scanner sc = new Scanner(System.in);  4 usages

    private static final char[][] tateti = {  25 usages
            {'-', '-', '-'},
            {'-', '-', '-'},
            {'-', '-', '-'},
    };

    /* INTERACCION */
    public static void inicio()  2 usages  Rodrigo
    {
        System.out.println("/////// TATETI ///////\n");
        System.out.println("SELECCIONE UNA OPCION: ");
        System.out.println("1. Jugar ");
        System.out.println("2. Salir ");
        String opcion = "0";
        while (opcion.equals("0")){
            opcion = sc.nextLine();
            switch (opcion) {
                case "1":
                    Juego();
                    break;
                case "2":
                    System.exit( status: 0);
                    break;
                default:
                    System.out.println("Opcion no valida");
                    opcion = "0";
            }
        }
    }
```

```java
private static void Juego(){  1 usage   Rodrigo
    boolean juegoTerminado = false;
    int[] jugada = new int[2];
    char letra = 'X';
    // Mientras el juego siga
    while (!juegoTerminado){
        letra = turnoPlayer1 ? 'X' : 'O';
        //      Mostrar el juego
        mostrarJuego();

        //      Pedir jugada al usuario
        jugada = pedirJugada(turnoPlayer1);

        //      Validar jugada del usuario
        while(!validarJugada(jugada)){
            //      Si no es valida que ingrese devuelta
            System.out.println("Jugada no valida");
            mostrarJuego();
            jugada = pedirJugada(turnoPlayer1);

        }

        //      Ver si es el player1 o no para colocar las letras
        //      Modificar el tateti
        insertarJugada(jugada[0],jugada[1],letra);

        //      Ver si gano el juego o hubo empate
        //      Mostrar el resultado
        if (ganoJugador(letra)){
            juegoTerminado = true;
        } else if(huboEmpate()){
            juegoTerminado = true;
            letra = '-';
        }
        turnoPlayer1 = !turnoPlayer1;
    }
    mostrarJuego();
    darVeredicto(letra);
    inicio();
}
```

```java
private static void mostrarJuego(){  3 usages  Rodrigo
    for (int i = 0; i < tateti.length; i++) {
        for (int j = 0; j < tateti[i].length; j++) {
            if (j == 1){
                System.out.print("| "+tateti[i][j]+" |");
            }
            else {
                System.out.print(" "+tateti[i][j]+" ");
            }
        }
        System.out.println();
    }
}

private static int[] pedirJugada(boolean turnoPlayer1){  2 usages  Rodrigo
    String jugador = turnoPlayer1 ? "Player1" : "Player2";
    int[] jugada = new int[2];
    System.out.println("Turno de "+jugador);
    //Fila
    System.out.println("Ingrese fila: ");
    try {
        jugada[0] = Integer.parseInt(sc.nextLine()) - 1;
    } catch(Exception e){
        jugada[0] = -1;
    }
    //Columna
    System.out.println("Ingrese columna: ");
    try {
        jugada[1] = Integer.parseInt(sc.nextLine()) - 1;
    } catch(Exception e){
        jugada[1] = -1;
    }
    //Por si desea salir
    System.out.println("Ingrese 's' si quiere salir: ");
    String salida = sc.nextLine().toLowerCase();
    if (salida.equals("s")) System.exit( status: 0);

    return jugada;
}
```

```java
/* VALIDACIONES */
private static boolean validarFilasColumna(int fila, int columna){  1 usage  new *
    return fila <= tateti.length && columna <= tateti.length && fila >= 0 && columna >= 0;
}

private static boolean validarJugada(int[] jugada){  1 usage  new *
    return validarFilasColumna(jugada[0],jugada[1]) && tateti[jugada[0]][jugada[1]] == '-';
}

private static boolean huboEmpate(){  1 usage  Rodrigo
    boolean empate = true;
    for (int i = 0; i < tateti.length; i++) {
        for (int j = 0; j < tateti[i].length; j++) {
            if (tateti[i][j] == '-') {
                return false;
            }
        }
    }
    return empate;
}

private static void insertarJugada(int fila, int columna,char letra){  1 usage  Rodrigo
    tateti[fila][columna] = letra;
}
```

```java
    private static boolean ganoJugador(char letra){  1 usage  Rodrigo
        boolean gana = false;
        //Por filas
        for (int i = 0; i < tateti.length; i++) {
            if (tateti[i][0] == letra && tateti[i][1] == letra && tateti[i][2] == letra){
                return true;
            }
        }

        //Por columnas
        for (int i = 0; i < tateti.length; i++) {
            if (tateti[0][i] == letra && tateti[1][i] == letra && tateti[2][i] == letra){
                return true;
            }
        }

        //Por cruce 1
        if (tateti[0][0] == letra && tateti[1][1] == letra && tateti[2][2] == letra){
            return true;
        }

        //Por cruce 2
        int letras = 0;
        for (int i = 0; i < tateti.length; i++) {
            if (tateti[i][tateti.length-(i+1)] == letra) letras++;
            if (letras == 3) return true;
        }
        return gana;
    }

    private static void darVeredicto(char letra){  1 usage  Rodrigo
        if (letra == 'X') System.out.println("Player1 win");
        else if (letra == 'O') System.out.println("Player2 win");
        else System.out.println("Empate");
    }

}
```

Main.java

```java
public class Main {  Rodrigo
    public static void main(String[] args) {  Rodrigo
        Tateti.inicio();
    }
}
```

5. Verificación y Validación

Pruebas

- **Caso básico 1:**
  - Player1: (1,1) -> X
  - Player2: (2,1) -> O
  - Player1: (1,2) -> X
  - Player2: (2,2) -> O
  - Player1: (1,3) -> X
- **Caso básico 2:**
  - Player1: (1,1) -> X
  - Player2: (1,2) -> O
  - Player1: (2,1) -> X
  - Player2: (2,2) -> O
  - Player1: (3,3) -> X
  - Player2: (3,2) -> O
- **Caso básico 3:**
  - Player1: (1,3) -> X
  - Player2: (1,2) -> O
  - Player1: (2,2) -> X
  - Player2: (3,1) -> O
  - Player1: (1,1) -> x
- **Caso básico 4:**
  - Player1: (1,2) -> X
  - Player2: (1,1) -> O
  - Player1: (2,1) -> X
  - Player2: (2,2) -> O
  - Player1: (3,2) -> X
  - Player2: (3,3) -> O

Resultados:

- **Caso básico 1:**
  - Resultado esperado: “Player1 win” (Correcto)
- **Caso básico 2:**
  - Resultado esperado: “Player2 win” (Correcto)
- **Caso básico 3:**
  - Resultado esperado: “Player1 win” (Correcto)
- **Caso básico 4:**
  - Resultado esperado: “Player2 win” (Correcto)

6. Conclusiones