Review information

- Jag har gjort en review på Muhammed Kan‘s TicTacoToe
- Repot finns här: https://github.com/Hamood420/Java21-Muhammed-Kan-Tic-Tac-Toe
- Repot gjorde jag en fork på och gjorde förslag på några ändringar
- Forken: https://github.com/tallner/Java21-Muhammed-Kan-Tic-Tac-Toe.git

Sammanfattning

Sammanfattningsvis fungerar koden förutom om man som användare råkar skriva in fel tecken vid inmatning i GUI.
Det finns en del saker man kan fundera över, tex om programmet ska växa och man ska lägga till fler funktioner.
Komplexiteten koden speglar uppgiften väl och det finns inget som är överdrivet komplext utfört.
Nedan ger jag en del input om vad jag har hittat som man kan fundera över.

Design

- Nuvarande struktur är en fil med all kod
  - Kan bli svårt att läsa om koden växer
  - Kan bli svårt att lägga till mer funktionalitet om programmet växer
  - Om man behåller denna struktur hade jag brutit ut UI i en egen metod som anropats från Main och jag hade även brutit ut en del kod från main metoden
- Förslag är att antingen
  - dela upp koden I olika klasser, tex i en MVC struktur
    - Model till player
    - View för vad som ska visas på UI
    - Controller för att skyffla data mellan Model och View
    - En UI class för att rita upp spelplan och menyer
  - eller att göra ett par metoder till, tex
    - UI
    - Hantera användarens input
    - Göra en metod för själva spelmotorn

Kommentarer

- Inga kommentarer i kod
- Skulle kunna finns enklare kommentarer på en övergripande nivå, tex vad är syftet med metoden, utan att gå in på detaljer

Coding conventions

- Följer coding conventions såvitt jag kan se
- Namngivning av metoder och variabler följer samma mönster

Funktionalitet

- All grundläggande funktionalitet i kravspecen är inte implementerad
  - Saknar omstart och att startande spelare slumpas

Versionhantering

- Ingen versionhantering
  - Rekommendation är att använda versionshantering även på mindre projekt som bara har en klass, då är det enklare att se vilka ändringar man har gjort

Förslag till förändringar

- Om man som användare skriver in bokstäver till GUI:t kraschar applikationen --> använd Try-Catch till Scanner eller ta in en String och kolla värdet
- Metoderna som anropas i samma klass kan vara private, dom behöver inte ge access utanför main klassen
- Scanner.close() används inte. Det medför ingen bugg i detta program, men kan orsaka problem med minnesläckage
- Förslagsvis skulle man kunna ta en stor del av koden i main() och lägga in i en gameEngine() metod. Det skulle kunna göra koden mer flexibel och lättläslig.

```java
//Codereview:might be easier to read if there is a method for taking care of the inputs and checking winner
private static String gameEngine(char[][] gameBoard, int pos, String user) {
    Random rand = null;
    int cpuPos = 0;

    if (user.equals("cpu")) {
        rand = new Random();
        cpuPos = rand.nextInt(9) + 1;
        while(playerPositions.contains(cpuPos) || cpuPositions.contains(cpuPos)) {
            cpuPos = rand.nextInt(9) + 1;
        }
        placePiece(gameBoard, cpuPos, user);
        printGameBoard(gameBoard);
    }else {
        placePiece(gameBoard, pos, user);
    }


    String result = checkWinner();
    if (result.length() > 0) {
        if (user.equals("player"))
            printGameBoard(gameBoard);
        System.out.println(result);
        return result;
    }

    return "";
}
```

- Inputs i GUI:t skulle kunna göras i en egen metod som rekursivt anropar sig själv. Då kan det vara lättare att kolla att rätt värde skickas in i GUI:t och att rätt input kommer till spelmotorn

```java
//Codereview:take care of user input with recursive call
private static int getUserInput(Scanner scan) {
    String res = scan.next();
    int returnval = 0;

    if (res.matches("[0-9]"))
        returnval = Integer.parseInt(res);
    else {
        System.out.println("Input needs to be a number");
        returnval = getUserInput(scan);
    }

    return returnval;
```

- Warnings i `checkWinner()`, borttagna med att definiera typen Integer i listan:

```java
public static String checkWinner() {

    List<Integer> topRow = Arrays.asList(1, 2, 3);
    List<Integer> midRow = Arrays.asList(4, 5, 6);
    List<Integer> botRow = Arrays.asList(7, 8,9);
    List<Integer> leftCol = Arrays.asList(1, 2, 7);
    List<Integer> midCol = Arrays.asList(2, 5, 8);
    List<Integer> rightCol = Arrays.asList(3, 6, 9);
    List<Integer> cross1 = Arrays.asList(1, 5, 9);
    List<Integer> cross2 = Arrays.asList(3, 5, 7);

    List<List<Integer>> winConditions = new ArrayList<List<Integer>>();
    winConditions.add(topRow);
    winConditions.add(midRow);
    winConditions.add(botRow);
    winConditions.add(leftCol);
    winConditions.add(midCol);
    winConditions.add(rightCol);
    winConditions.add(cross1);
    winConditions.add(cross2);

    for(List<Integer> l : winConditions) {
        if(playerPositions.containsAll(l)) {
```

Unit tester

- Inga testar implementerade, det vore bra om det fanns några enklare tester tex
    - Testa vilka inputs programmet klarar av, tex bokstäver eller nummer utanför 1-9

Dokumentation

- Ingen dokumentation om hur programmet ska användas, men det är intuitivt.
- Dock skulle man kunna ha någon enkel text när programmet startar hur det ska användas
- Jag har lagt till ett enkelt init UI med kort dokumentation om hur programmet fungerar

```java
//Codereview:added a small initial UI so the user can understand the rules
  private static char[][] initUI() {
      char[][] initGameBoard = {
              {' ', '|', ' ', '|', ' '},
              {'-', '+', '-', '+', '-'},
              {' ', '|', ' ', '|', ' '},
              {'-', '+', '-', '+', '-'},
              {' ', '|', ' ', '|', ' '}
      };
      char[][] gameBoardNumbers = {
              {'1', '|', '2', '|', '3'},
              {'-', '+', '-', '+', '-'},
              {'4', '|', '5', '|', '6'},
              {'-', '+', '-', '+', '-'},
              {'7', '|', '8', '|', '9'}
      };
      System.out.println("Welcome to Tic Tac Toe");
      System.out.println("Type a nr between 1 and 9");
      System.out.println("1 is the first box in the first row");
      System.out.println("9 is the last box in the last row");
      System.out.println("");
      printGameBoard(gameBoardNumbers);
      System.out.println("");
      System.out.println("You are X");
      System.out.println("CPU is O");
      System.out.println("First player to get 3 in a row wins");
      System.out.println("Good Luck!");
      System.out.println("");

      //printGameBoard(initGameBoard);
      System.out.println("");
      return initGameBoard;
```