Review information

* Jag har gjort en review på TicTacoToe
* Repot: <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>

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 strukturen hade jag brutit ut UI i en egen metod som anropats från Main
* Förslag är att 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

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

Funktionalitet

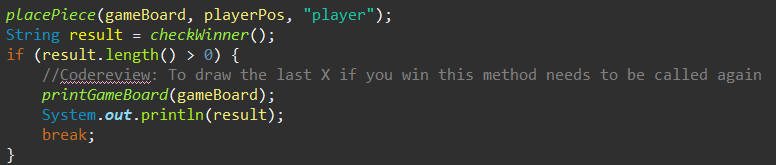
* All grundläggande funktionalitet i kravspecen är inte implementerad
  + Saknar omstart och att startande spelare slumpas

Versionhantering

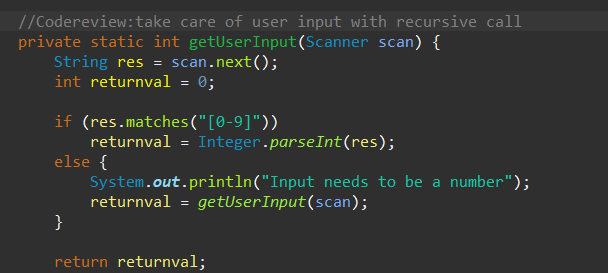
* Ingen versionhantering
  + Rekommendation är att använda versionshantering även på mindre projekt som bara har en klass, då är det enklare att se vilka ändringar man har gjort

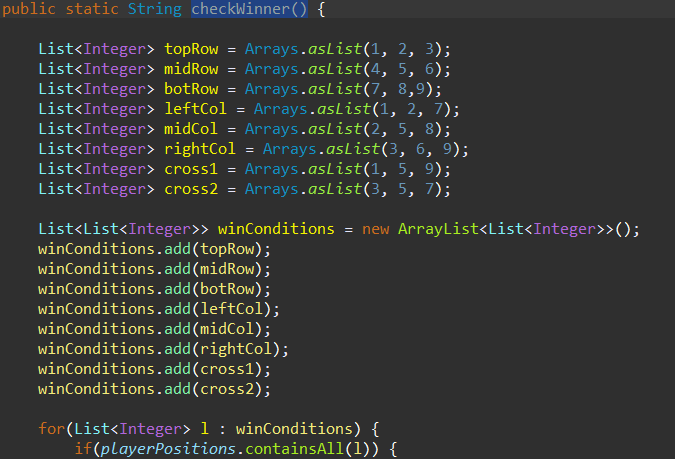
Förslag till förändringar

* Om man inputar bokstäver 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
* Ingen bugg i detta program, men Scanner.close() används inte. Kan orsaka problem med minnesläckage
* Om användaren vinner så ritas inte sista X:et ut, man kan göra det genom att lägga till en printGameBoard(gameBoard) innan programmet avslutas



* Inputs kan göras i en egen metod som rekursivt anropar sig själv till rätt input kommer



* Warnings i checkWinner(), borttagna med att definiera typen i listan:

Tester

* Inga testar implementerade, det vore bra om det fanns några enklare tester tex
  + Testa vilka inputs programmet klarar av

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

Sammanfattning:

- Är koden design- och strukturmässigt bra?

- Funktionaliteten stämmer överens med krav eller syfte.

- Alla UI förändringar är logiska och enbart förbättringar ur princip-syfte.

- Koden är inte mer komplex än vad som krävs.

- Använder sig utvecklaren av metoder som gör det möjligt att vidareutveckla produkten

på ett säkert och effektivt sätt i framtiden?

- Koden har unit tests, om inte föreslå det.

- Testerna är välgjorda, om inte finns gjorda föreslå det.

- Utvecklaren har använt sig utav lämpliga namn till klasser, variabler etc.

- Kommentarerna är konsista och informativa, framför vad koden gör simpelt.

- Koden är dokumenterat på ett effektivt sätt, om inte föreslå det.