Tre-på-rad (Bondesjakk)

I denne oppgaven skal du lage en forenklet utgave av spillet tre-på-rad, også kalt bondesjakk. Du skal gjøre dette ved å benytte objektorienterte prinsipper, altså med klasser og objekter.

Tre-på-rad, eller bondesjakk, foregår på et rutenett av 3*3 ruter. To spillere konkurrerer, og i hvert trekk får de plassere en brikke på rutenettet. Vinneren av spillet er den første spilleren som har tre brikker på rad, enten vertikalt, horisontalt eller diagonalt.

Vi skal bygge programmet vårt fra bunnen og opp (bottom-up), og begynner med å modellere spillere og ruter. Vi går deretter over til å bygge opp brettet som helhet.

Oppgave A: Klassen Spiller

Lag en klasse `Spiller`. En spiller skal representeres på brettet enten med symbolet `X` eller symbolet `0`, angitt ved opprettelse av en ny spiller. Om symbolet som gis ikke er `X` eller `0` skal det printes en beskjed om at symbolet er ugyldig.

Oppgave B: Klassen Rute

Lag en klasse `Rute`. En rute representerer en enkelt rute på et tre-på-rad-brett. Ruter er enten ledige eller opptatt, og om de er opptatt har de en referanse til spilleren som har plassert brikken sin der. Når en rute opprettes settes denne referansen til `None`.

Ruter har en metode `plasser_brikke(self, spiller)` som setter spiller-referansen til spilleren som gis som parameter.

Ruter har en metode `er_opptatt(self)` som returnerer True dersom ruten er tatt av en spiller, ellers False.

Ruter har en metode `hent_brikkeeier(self)`. Denne returnerer spiller-objektet som ligger i spiller-referansen, eventuelt `None`.

Oppgave C: Klassen Brett

Lag klassen `Brett`. Et brett er et rutenett på 3*3 ruter, hvor hver rute representeres ved et objekt av klassen `Rute`. Datastrukturen er en nøstet liste, og rute-objekter opprettes i konstruktøren. Konstruktøren oppretter også to referanser til spiller-objekter som i utgangspunktet settes til `None`.

Skriv en metode `legg_til_spiller(self, symbol)`. Metoden oppretter et `Spiller`-objekt med symbolet gitt som parameter, og setter inn dette objektet i en ledig spiller-referanse. Om ingen spiller-referanser er ledig, printes en beskjed om at det allerede eksisterer to spillere.

Skriv en metode `plasser_brikke(self, spiller, x, y)`. Spilleren gitt i parameteren forsøker å plassere en brikke i ruten på plass `[x][y]`. Sjekk om det er mulig å plassere en brikke på denne plassen, og send i så fall korrekt spiller-referanse til ruten i posisjonen. Om ruten allerede er tatt, printes en beskjed om dette.

(...bla om)

Oppgave D: sjekk_vinner

Skriv en metode `sjekk_vinner(self)` i klassen `Brett`. Denne sjekker alle ruter horisontalt og vertikalt, og returnerer `None` dersom ingen har vunnet. Om en spiller har tre brikker på rad i noen av retningene, returnerer den symbolet til spilleren som har vunnet.

Ekstra: Sjekk også om en spiller har tre brikker på rad diagonalt.

Oppgave E: spill

Skriv en metode `spill(self)` i klassen `Brett`. Denne gjør følgende:

- Sjekker at det eksisterer to spillere.
- I en løkke:
  - Lar spillerne vekselvis forsøke å plassere brikke ved input av x- og y-koordinat på spillebrettet. Om valget er ugyldig (utenfor brettet eller opptatt rute) skal spilleren få forsøke igjen.
  - Hver gang en brikke er plassert, sjekkes det om noen har vunnet. Om det finnes en vinner, printes vinnerens symbol og spillet er over.
  - Om alle ruter er utfylt og det ikke finnes en vinner, printes det «Ingen vinner», og spillet avsluttes.