3.sem DW (OO-Programming) E17

Obligatorisk afleveringsopgave 1 *Erik Sørensen 30. august 2017*

Som bekendt skal der være et antal porteføljeopgaver i kurset. I OOP delen vil der være to obligatoriske programmeringsopgaver, som skal løses, for at kunne deltage i den afsluttende eksamen:

Opg.1: En simpel udgave af det velkendte, gamle brætspil *Matador (Monopoly* på engelsk).
Opg.2: En simpel TCP/IP server, med adgang til databaser (jdbc og NoSql). Denne opgave kobles sammen med en klient fra Andrea's del af kurset.

Opgaverne løses i grupper på 2-3 studerende og afleveres på BlackBoard hhv. **25/10 under lektion 7** og **29/11 under lektion 12**. Rettidig aflevering er en betingelse for at kunne deltage i eksamen.

Tidsplan

Følgende vejledende tidsplan er lagt for projektet. Undervisningen under lektionerne er tilrettelagt, så de nogenlunde følger planen.

Lektion 2: Gruppedannelse og start på opgave 1.

Lektion 3: Kast med terninger og flytning af spillere rundt på spillepladen.

Lektion 4 - 5: Arv og Polymorphism på felterne.
Køb af Grunde/Bryggerier/Dampskibsselskaber
samt betaling for at lande på anden spillers ejendom.

Lektion 6: File I/0. Informationer skrives i / læses fra filer.

Lektion 7: Aflevering af NetBeans projekt på BlackBoard,
Fremlæggelse for klassen og (evt.) godkendelse.

Lektion 8: Start på opgave 2.
Simpel, trådet TCP/IP server

Lektion 9 - 10: JDBC: Informationer skrives i / læses fra SQL-database

Lektion 11: NoSQL: Informationer skrives i / læses fra NoSQL-database

Lektion 12: Færdiggørelse og aflevering.

Porteføljeopgave 1: Matador

Der skal udvikles en simpel udgave af det velkendte, gamle brætspil *Matador (Monopoly* på engelsk).

Herunder gives et forslag til arbejdet fra lektion til lektion, indtil det samlede projekt afleveres under lektion 7.

Opgave til Lektion 3

Efter lektion 3 bør (mindst) følgende være implementeret:

- En klasse `DiceCup` indeholdende to instancer af `Die`, hver repræsenterende en terning med 6 sider. `DiceCup` skal have en metode, `public int throwDice()`, som kaster de to terninger og returnerer det samlede antal øjne. Desuden en metode som, når den kaldes, fortæller om der blev slået to ens.
- En klasse `Field`, repræsenterende felterne på brættet. I første omgang behøver `Field` kun at indeholde disse indkapslede variable:
  - `String name` // et kort navn på feltet
  - `int number` // et tal i intervallet [0..39]

  Begge variable skal initialiseres i en constructor og der må kun være `getter()`-*metoder* på dem, da de aldrig skal ændres efter oprettelsen.

  Desuden skal der være en metode med signaturen `public String toString()`, så det er nemt at udskrive hvilket `Field` en spiller er landet på.

  NB: *Klassen er foreløbig og skal erstattes af et interface og et arve-hierarki senere i forløbet.*
- En klasse `Player`, som (mindst) indeholder variablene:
  - `String name` // et kort navn på spilleren
  - `Field currentField` // En reference til det felt spilleren står på

  `Player` skal indeholde en metode, `public int move(DiceCup cup)`, så det simuleres at spilleren får raflebægeret og selv slår med terningerne.

- Simulering af spillet skal implementeres i en driver-klasse, som opretter et statisk array med 40 pladser, hver indeholdende en reference til en instance af klassen `Field`. Desuden oprettes mindst 2 spillere, som placeres på "Start"-feltet, (indeks 0 i arrayet). I et loop skal spillerne kaste

terninger efter tur og det skal skrives på `System.out`, hvor de lander efter hvert slag. Find selv på en betingelse for at slutte spillet, fx kan vinderen være den der først når et antal omgange rundt.

Opgave til lektion 4 og 5

- *Arv, Interfaces og polymorphism på felter.*
- *Metode til consequence() implementeres. (Defineres i et Interface eller abstrakt klasse og implementeres i de konkrete klasser)*
- *player tilføjes en liste til købte grunde mv.*

Yderligere info kommer senere.