Projekt 4

Klasser og Arkitektur

Erhvervsakademiet Aarhus
Datamatikeruddannelsen
April 2023

Projektbeskrivelse for projekt Klasser og Arkitektur

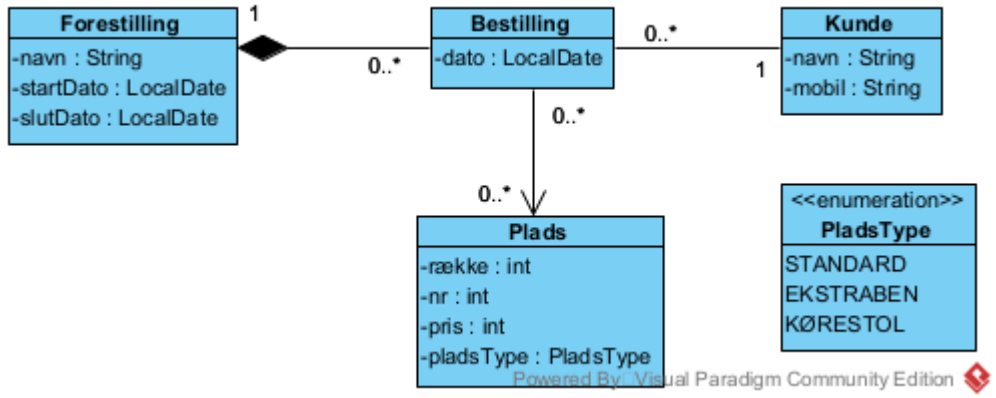
Produkt mål

Denne opgave omhandler et system, der skal holde styr på bestillinger til forestillinger på et teater. En forestilling opføres fra og med startdatoen til og med slutdatoen. En bestilling afgives af en kunde til en forestilling, der spilles på en bestemt dato. En bestilling indeholder et antal pladser i den sal, hvor forestillingen opføres. Nogle pladser har bedre benplads end andre og nogle pladser kan købes til kørestolsbrugere.
Systemet skal have en grafisk brugergrænseflade hvor det er muligt at oprette kunder og forestillinger samt at lave bestilling af billetter.

Opgave 1

Programmering af klasserne i model pakken

Betragt nedenstående UML- designklassediagram for systemet.



Programmér klasserne *Plads*, *Forestilling*, *Bestilling* og *Kunde* i pakken *model*.
Klasserne skal have attributter med tilhørende get-metoder, og en konstruktor som initialiserer alle attributter. Lav ikke set-metoder, men tilføj en set-metode senere, hvis du får brug for den.

Programmér sammenhængen mellem klasserne – både linkattributter og metoder til at sætte dem. . Metoder til at opdatere/slette sammenhænge skal ikke med, da sammenhænge ikke kan opdateres/slettes i denne applikation.

Multipliciteter og retninger fremgår af figuren (bemærk, at der er to dobbeltrettede sammenhænge).

Opgave 2

Programmering af Storage

Tilføj klassen *Storage* i pakken *storage*. Klassen skal indeholde lister med systemets forestillinger, kunder og pladser. Klassen skal også indeholde metoder til at gemme objekter af klasserne *Forestilling*, *Kunde* og *Plads*, og metoder til at hente alle forestillinger, kunder og pladser.

Opgave 3

Programmering af Controller

Tilføj klassen `Controller` i pakken `controller`. Klassen skal indeholde metoder til at oprette objekter af klasserne `Forestilling`, `Kunde` og `Plads`.

Opgave 4

Programmering af App klasse med main() metode

Lav en *App* klasse med en *main()* metode i pakken *gui*.
Tilføj til *App* klassen en metode `initStorage()`, der under anvendelse af opret-metoderne opretter og gemmer data svarende til nedenstående:

Forestillinger:	Navn: *Evita*	Spiller fra 2023-08-10 til 2023-08-20
	Navn: *Lykke Per*	Spiller fra 2023-09-01 til 2023-09-10
	Navn: *Chess*	Spiller fra 2023-08-21 til 2023-08-30
Kunder:	Navn: *Anders Hansen*	Mobil: *11223344*
	Navn: *Peter Jensen*	Mobil: *12345678*
	Navn: *Niels Madsen*	Mobil: *12341234*

Pladser svarende til 15 rækker og 20 pladser i hver række:
De gule koster 500 kroner, de grønne koster 450 kroner og de blå pladser koster 400 kroner

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10								K	K	K	K	K								
11								EB	EB	EB	EB	EB								
12																				
13																				
14																				
15																				

K: Plads til kørestol. EB: Ekstra benlængde. De øvrige pladser er standard pladser.
Tilføj en metode *testPrint()* til *App* klassen, der udskriver forestillingerne, kunderne og pladserne, som er oprettet i *initStorage()* metoden.

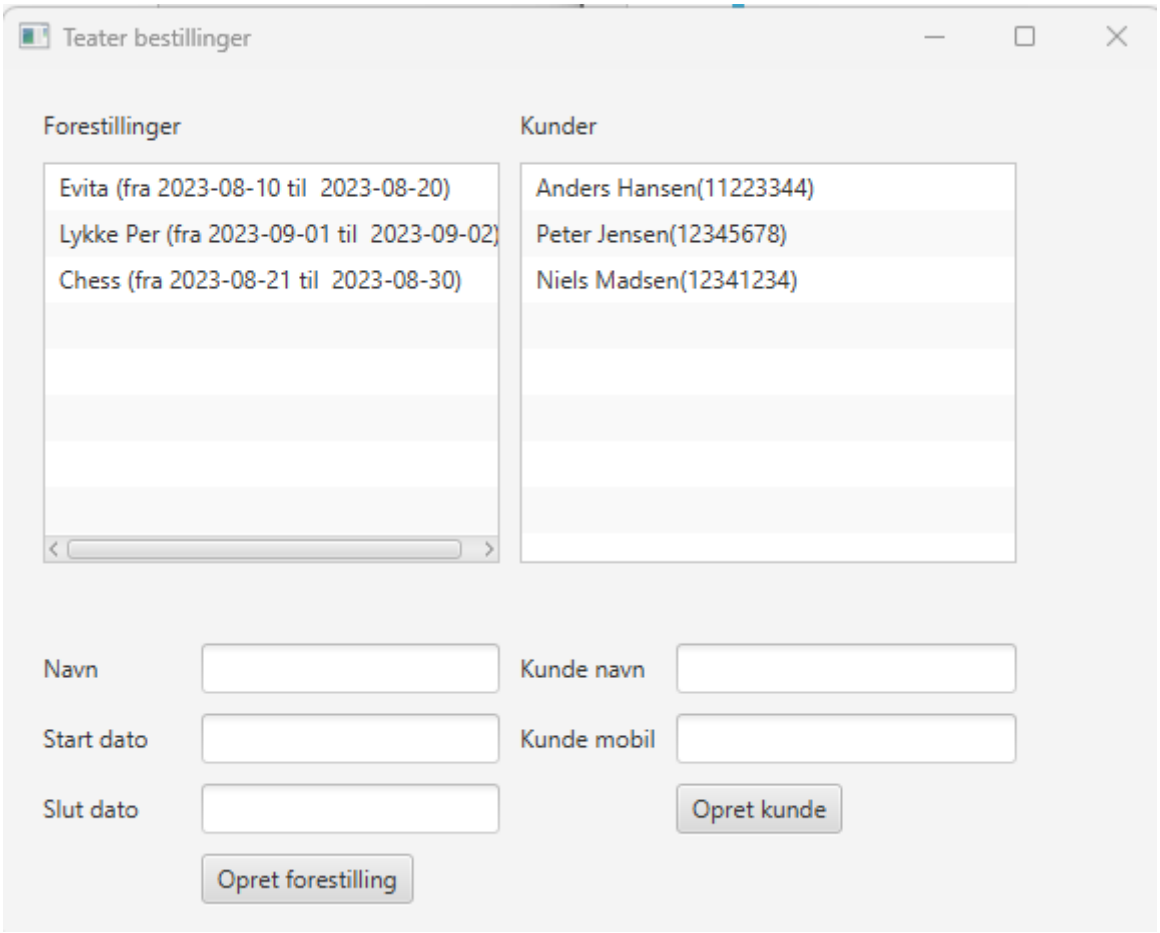
Opgave 5

Programmering af GUI til administrationsdel

Programmér en grænseflade i pakken *gui* under anvendelse af JavaFX, så følgende use cases kan udføres under anvendelse af storage, controller og model:

- Vise en liste over alle forestillinger (i et ListView).
- Vise en liste over alle kunder (i et ListView).
- Operette en forestilling.
- Oprette en kunde.

Et vindue der indeholder ovenstående kunne for eksempel se ud som vist i figuren herunder.



Opgave 6

Programmering af use casen: Bestil billetter

1) Tilføj til klassen *Forestilling* metoden *erPladsLedig(int række,int nr, LocalDate dato): boolean*, der returnerer om den pågældende plads er ledig til forestillingen på den angivne dato.

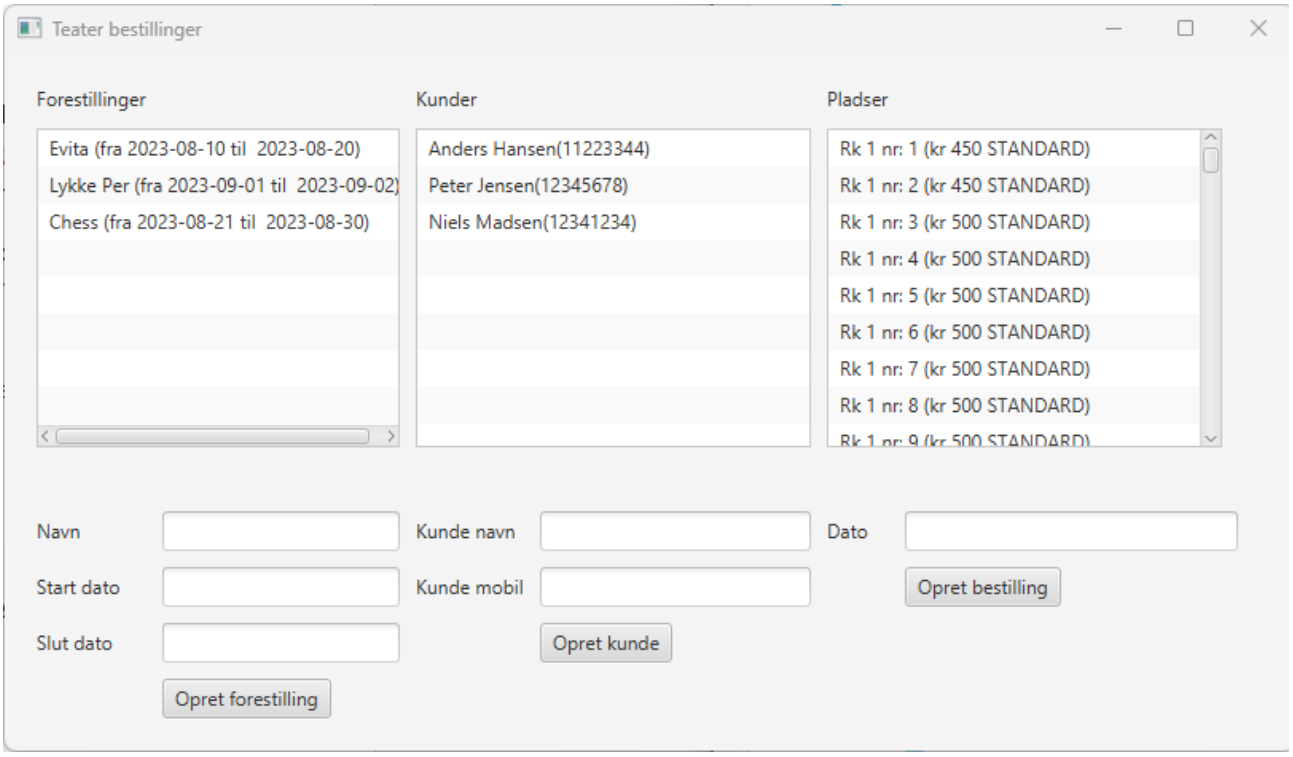
2) Tilføj til klassen *Controller* en metode til at oprette en bestilling med et antal pladser. Metodens hoved ser således ud:

```
public static Bestilling opretBestillingMedPladser(
                        Forestilling forestilling, Kunde kunde,
                        LocalDate dato, ArrayList<Plads> pladser)
```

   Metoden skal checke, om de givne pladser er ledige, og at den ønskede dato er i den periode, hvor forestillingen vises. Hvis en af de givne pladser ikke er ledige på den givne dato, eller forestillingen ikke vises på den givne dato, skal metoden ikke oprette en bestilling men returnere *null*.

3) Tilføj i pakken gui kode der gør det muligt at vælge en forestilling, en kunde og en dato samt en antal pladser, hvorefter der oprettes en bestilling til kunden med de angivne pladser. Hvis datoen ikke er i forestillingens spilleperiode, eller hvis ikke alle pladser er ledige, skal bestillingen ikke oprettes. Det er et krav, at programmet åbner et dialogvindue med besked om, hvad der er oprettet (eller ikke oprettet).

   En *gui* der indeholder ovenstående kunne for eksempel se ud som vist i figuren herunder.



Opgave 7

Programmering af metoder til statistik

1) Tilføj til klassen *Bestilling* metoden *samletPris(): int*, der returnerer den samlede pris for en bestilling.

2) Tilføj til klassen *Forestilling* metoden
   antalBestiltePladserPåDag(LocalDate dato): int
   der returnerer, hvor mange pladser der er bestilt til forestillingen på den pågældende dato.

3) Tilføj til klassen *Forestilling* en metode *succesDato*, der returnerer datoen for den dag, hvor der har været flest pladser bestilt til forestillingen. Hvis der er flere dage, der har haft dette antal, returneres blot en af datoerne.

4) Tilføj til klassen *Kunde* en metode *bestiltePladserTilForestillingPådag*. Metoden skal tage en forestilling og en dato som parameter og returnere en liste med alle pladser, kunden har bestilt til forestillingen på datoen. Bemærk, at en kunde kan have flere bestillinger til forestillingen på dagen.

Opgave 8 - Ekstra

GUI til statistik

Tilføj i pakken gui kode der gør det muligt at vise resultatet af kaldene til metoderne fra opgave 7 i en grafisk brugergrænseflade.

En anden mulighed er at tilføje kode i App klassen, så metoderne i opgave 7 kan testes.

Faglige mål

At du efter projektforløbet

- Kan programmere en design klassemodel
- Kan programmere i en lagdelt arkitektur
- Kan lave en simpel grafisk brugergrænseflade med JavaFx
- **Benspænd**: Programmer uden brug af kommandoen *break* og med højst et *return* statement i hver metode.

Proces mål

At du arbejder sammen med nogle andre studerende fra klassen og dermed kommer til at kende dine klassekammerater bedre.

Rammer for projektet

Projektet starter onsdag d. 26. april.
Der skal arbejdes i grupper med 3 studerende i hver gruppe. **Alle** i gruppen skriver koden på **deres egen maskine**. Alle i gruppen hjælper hinanden med at finde ud af, hvad koden skal indeholde.

Aflevering

Projektet afsluttes med en aflevering af de programmerede klasser. Projektet afleveres senest søndag d. 30. april kl. 16.00 på Canvas (nærmere detaljer følger på klassen). Afleveringen er en zip-fil indeholdende source koden for projektet (navngiv zip-filen med gruppens fornavne).