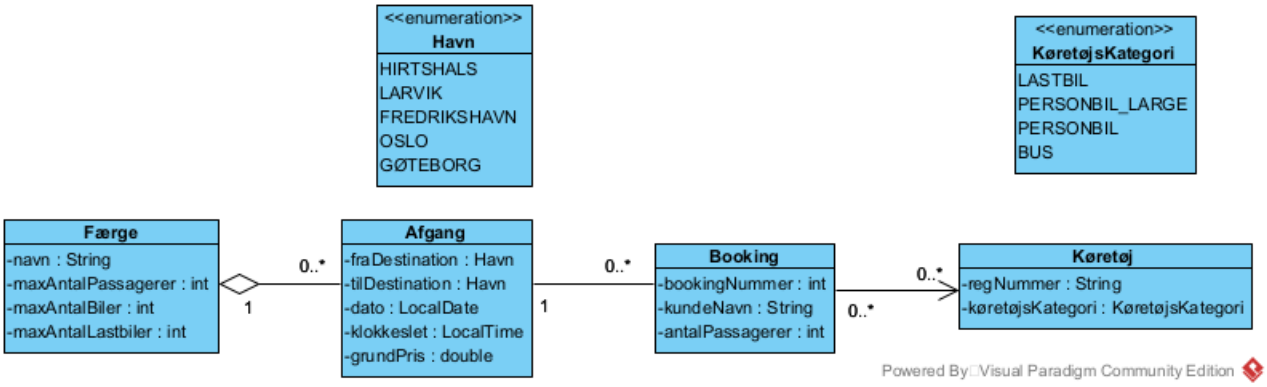
ERHVERVSAKADEMI
AARHUS

Semesterprøve Programmering januar 2025 (100 points)

Denne opgave omhandler et system programmeret under anvendelse af 3-lags modellen, der kan administrere bookinger af færger mellem sejlruter i norden.
Ved en booking skal man kunne vælge ønsket afgang på en rute, der skal kunne bookes med en eller flere køretøjer og et antal passagerer. Køretøjerne kan være af forskellig slags: lastbil, almindelig personbil, stor personbil eller en bus. Prisen for en booking afhænger af afgangens grundpris, antallet af køretøjer på bookingen, antallet af passagerer og hvilket slags køretøj det drejer sig om. Køretøjet skal være registreret med et registreringsnummer.
Figuren nedenfor viser et simpelt design klassediagram for systemet. Bemærk, at metoder ikke er vist på diagrammet.



Opgave S1 (25 point)

Programmér klasserne *`Færge`*, *`Afgang`*, *`Booking`* og *`Køretøj`* i pakken *model*. Klasserne skal have attributter med tilhørende get-metoder og en konstruktor, som initialiserer attributter. Lav ikke set-metoder, men tilføj en set-metode senere, hvis du får brug for den. Den første Booking der oprettes skal have nummer 101 og hver gang der efterfølgende oprettes en ny booking skal den have et *bookingNummer* der ikke har været anvendt tidligere.

Programmér sammenhængen mellem klasserne – både linkattributter og metoder, idet det **skal** være muligt såvel at oprette samt opdatere/fjerne sammenhængene.

Opgave S2 (3 point)

Tilføj klassen `Storage` i pakken *storage*. Klassen skal indeholde tre `ArrayList'er` indeholdende systemets færger, bookinger og køretøjer. Programmér klassen `Storage` med de tilhørende metoder til at gemme og slette objekter af klasserne `Færge, Booking` og `Køretøj`. Programmer også metoder til at hente listerne.

Opgave S3 (11 point)

Tilføj klassen `Controller` i pakken *controller*. Klassen skal indeholde metoder til at oprette de objekter, der indgår i modellen, og skabe sammenhæng mellem dem.
Tilføj klassen *App* med en *main*() metode i pakken gui. Klassen skal have en metode `initStorage()`, der opretter og gemmer data svarende til nedenstående:

Færger:

Jutlandica der maksimalt har plads til *3000* passagerer *500* biler og *50* lastbiler
Danica der maksimalt har plads til *4000* passagerer *600* biler og *70* lastbiler

Køretøjer:

AH 12 223	*Personbil (PERSONBIL)*
DE 33 123	*Stor personbil (PERSONBIL_LARGE)*
GH 33 444	*Lastbil (LASTBIL)*
JK 55 434	*Bus (BUS)*

Afgange:

Jutlandica har afgangene:
10/1 - 2025 kl 8.00 fra Frederikshavn til Gøteborg med grundpris 1200 kroner
3/2 - 2025 kl 8.00 fra Frederikshavn til Gøteborg med grundpris 1200 kroner
3/2 - 2025 kl 13.00 fra Gøteborg til Frederikshavn med grundpris 1200 kroner
3/2 - 2025 kl 18.00 fra Frederikshavn til Gøteborg med grundpris 1000 kroner
3/2 - 2025 kl 23.00 fra Gøteborg til Frederikshavn med grundpris 1000 kroner

Danica har afgangene:
10/2 - 2025 kl 8.00 fra Frederikshavn til Gøteborg med grundpris 1200 kroner
10/2 - 2025 kl 13.00 fra Gøteborg til Frederikshavn med grundpris 1200 kroner
10/2 - 2025 kl 18.00 fra Frederikshavn til Gøteborg med grundpris 1000 kroner
10/2 - 2025 kl 23.00 fra Gøteborg til Frederikshavn med grundpris 1000 kroner

Bookinger:
Jens Jensen har to bookinger, begge bookinger har AH 12 223 og DE 33 123 som køretøjer og der er i alt 8 personer tilknyttet bookingerne. Han sejler fra Frederikshavn den 3/2 kl 08.00 og sejler fra Gøteborg den 10/2 kl 13.00

Opgave S4 (4 point)

Tilføj til klassen `Afgang` metoden `antalPassagerer: int,` der returnerer, hvor mange passagerer der er registreret på bookinger hørende til den aktuelle afgang.

Opgave S5 (8 point)

Tilføj til klassen `Booking` metoden `samletPris() : double`, der returnerer den samlede pris for en booking, idet prisen beregnes ud fra nedenstående regler.

Der betales grundpris for hvert af køretøjerne på bookingen. For store personbiler er der et tillæg på 10% af grundprisen, for en bus betales 3 gange grundprisen og for en lastbil 5 gange grundprisen. I grundprisen er der inkluderet en person for hvert køretøj, for alle yderligere personer betales 50 kr.

Bemærk: Det kan antages at der ikke er gående passagerer.

Opgave S6 (7 point)

I denne opgave skal der tilføjes to metoder:

Tilføj til klassen `Afgang` metoden `antalLedigePersonbilPladser() : int`, der returnerer hvor mange ledige pladser der er til personbiler på afgangen(der er ikke forskel på store og almindelige personbiler). Bemærk de enkelte færger har en attribut, der fortæller hvor mange personbiler der maksimalt er plads til på færgen.

Tilføj til klassen `Afgang` metoden `antalLedigeLastbilPladser() : int`, der returnerer hvor mange ledige pladser der er til busser og lastbiler på afgangen (en bus svarer i plads til en lastbil) . Bemærk de enkelte færger har en attribut, der fortæller hvor mange lastbiler der maksimalt er plads til på færgen.

Opgave S7 (10point)

Tilføj til klassen *Controller* en metode der kan lave en ombooking. I en ombooking kan der vælges en anden færgeafgang og det er det eneste der kan ændres ved en ombooking. Metoden skal som parameter tage den booking der skal ændres og den afgang som bookingen skal ændres til. For at en ombooking kan udføres skal følgende være overholdt:

- tidspunktet for den oprindelige afgang er i fremtiden
- der er mindst 15 minutter til afgang, for den afgang der ønskes ombooket til
- fra og til destination er de samme for den oprindelige og den nye afgang
- der skal være plads til passagerer og køretøjer på den afgang der ønskes at ombooke til

Hvis ovenstående ikke er overholdt skal der kastes en *passende RuntimeException*, med en passende fejlbesked og der laves ikke ombooking.

Opgave S8 (6 point)

Tilføj til klassen `Færge` metoden `findBooking(int bookingNummer) : Booking`, der returnerer den booking der har det pågældende *bookingNummer*. Hvis en sådan booking ikke findes, skal der returneres *null*. Det er et **krav** at løsningen implementeres under anvendelse af søgeskabelonen.

Opgave S9 (6 point)

Tilføj til *Controller* klassen en metode der i en fil udskriver informationer om alle færger og deres afgange. Oversigten skal for hver afgang udskrive, hvor mange passagerer der er på afgangen samt hvor mange ledige pladser der er til henholdsvis personbiler og lastbiler. Derudover skal der også for hver færge vises, summen af passagerer, summen af ledige personbil pladser og summen af ledige lastbil pladser. Nedenfor er vist et udklip af hvordan filen kunne se ud:

Jutlandica
FREDERIKSHAVN-GOTEBORG:2025-10-01 08:00 Antal passager: 0 Ledige pladser, personbiler: 500 Ledige pladser, lastbiler: 50
FREDERIKSHAVN-GOTEBORG:2025-02-03 08:00 Antal passager: 8 Ledige pladser, personbiler: 498 Ledige pladser, lastbiler: 50
GOTEBORG-FREDERIKSHAVN:2025-02-03 13:00 Antal passager: 0 Ledige pladser, personbiler: 500 Ledige pladser, lastbiler: 50
FREDERIKSHAVN-GOTEBORG:2025-02-03 18:00 Antal passager: 0 Ledige pladser, personbiler: 500 Ledige pladser, lastbiler: 50
GOTEBORG-FREDERIKSHAVN:2025-02-03 21:00 Antal passager: 0 Ledige pladser, personbiler: 500 Ledige pladser, lastbiler: 50
Antal passager ialt: 8. Samlet antal ledige personbilpladser: 2498. Samlet antal ledige lastbilpladser: 250

Danica
FREDERIKSHAVN-GOTEBORG:2025-02-10 08:00 Antal passager: 0 Ledige pladser, personbiler: 600 Ledige pladser, lastbiler: 70
GOTEBORG-FREDERIKSHAVN:2025-02-10 13:00 Antal passager: 8 Ledige pladser, personbiler: 598 Ledige pladser, lastbiler: 70
FREDERIKSHAVN-GOTEBORG:2025-02-10 18:00 Antal passager: 0 Ledige pladser, personbiler: 600 Ledige pladser, lastbiler: 70
GOTEBORG-FREDERIKSHAVN:2025-02-10 23:00 Antal passager: 0 Ledige pladser, personbiler: 600 Ledige pladser, lastbiler: 70
Antal passager ialt: 8. Samlet antal ledige personbilpladser: 2398. Samlet antal ledige lastbilpladser: 280

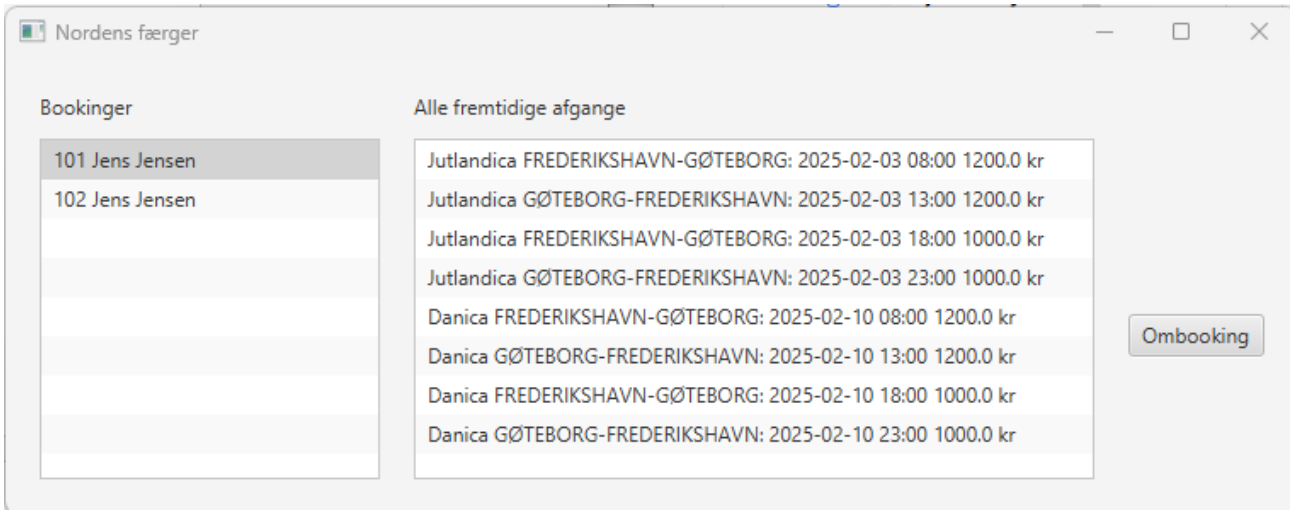
Opgave S10 (20 point)

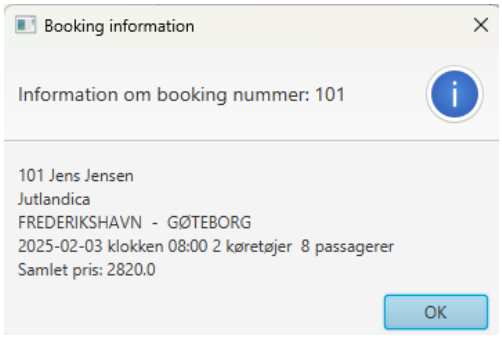
Programmér en GUI under anvendelse af JavaFX, så følgende use cases kan udføres under anvendelse af storage, controller og model:

1. Vise en liste over alle bookinger (ListView)
2. Vise en liste over alle **fremtidige** afgange (ListView)
3. Vise relevant information om en valgt booking (Informations dialog -skal her som minimum vise, hvor der sejles fra og til og hvad den samlede pris er)
4. Lave en ombooking af den valgte booking til den valgte afgang

Et vindue der indeholder ovenstående kunne for eksempel se ud på følgende måde:

Bemærk: Du skal fange relevante exceptions og håndtere disse. Du må/skal gerne tilføje metoder/kode andre steder i applikationen for at løse denne opgave.





Du skal aflevere én zip fil på Wiseflow indeholdende pakkerne med alle de programmerede klasser.