Opgave: Rejsebureau PlusBus

# Introduktion

Formål med opgaven:

* Problemformulering og kravspecifikation i denne opgave ligner dem i eksamensopgaven.
* At løse denne opgave forbereder dig godt på eksamen.

Fremgangsmåde:

* Læs opgavestillingen nøjagtigt!
* Find detaljerne hvordan man gør i vores DanskCargo-opgave. Vi løser stort set de samme problemer som i DanskCargo-opgaven men nu får du ingen detaljerede anvisninger og heller ingen eksempel-kode.
* Kopiere skamløst kode fra din DanskCargo-løsning eller fra lærerens sidste DanskCargo-milepæl :-)
* Kommenter din kode rigeligt så du forstå den stadig når du kommer til at genbruge den. Dine kommentarer vil også være meget nyttige, når du skal forklare din kode til eksamen.

Hvis du sidder fast med noget:

* Læs altid hele opgaven før du begynder at løse den.
* Research selv (google, w3schools.com, stackoverflow.com, …).
* Spørg de andre elever.
* Spørg læreren. Senest når du har siddet fast i en opgave i en halv time, spørg læreren eller send læreren en Teams Chat besked som ”*jeg sidder fast i en opgave*”.

# Problemformulering

PlusBus er et busselskab for busrejser indenfor Danmark. Vi skal udvikle et program for PlusBus.

Den skal have en brugergrænseflade til håndtering af oprettelse og vedligeholdelse af kundekartoteket og af de rejsearrangementer kunderne kan booke. Derudover kan man i programmet gemme, redigere og slette bookinger.

# Kravspecifikation

De følgende oplysninger skal gemmes i en SQL-database og håndteres af vores program:

Om Rejsebureauets Kunder:

* Efternavn
* Kontakt (Telefonnummer eller E-mail)

Om de rejser som kunderne kan booke:

* Rute (i hvilken by begynder rejsen og i hvilken by enden den)
* Dato
* PladsKapacitet (for så mange personer er der plads ved denne rejse)

Om bookinger:

* KundeId
* RejseId
* Pladser (hvor mange pladser har kunden booket på denne rejse)

Alle nævnte oplysninger kan man oprette, redigere og slette i GUIen. (Sletning kunne være fuldstændigt eller logisk. Du beslutter hvad giver mere mening.)

Før man kan gemme en ny booking skal det tjekkes om rejsen har nok frie pladser tilbage.

# Opret filer

Opret en ny mappe kaldet plusbus i pycharm i dit løsningsprojekt.

Opret nu en indledningsvis tom python-fil i denne mappe for hver lag:

* Datastruktur-lag
* funktionelt lag
* Database-lag (SQL-lag)
* GUI-lag

Det er en god idé at også åbne din DanskCargo-solution eller lærerens sidste DanskCargo-milepæl parallelt så du kan sammenligne og genbruge kode.

# Planlæg databasen

Besvar de følgende spørgsmål. Helst først på en seddel.

* Hvilke database-tabeller har du brug for? Hvad skal de hedde?
* Hvilke kolonner har du brug for i tabellerne? Hvad skal de hedde?
* Hvilke (sqlalchemy-)datatyper har kolonnerne?
* Hvad er dine primærnøgle?
* Hvilke forbindelser er der mellem tabellerne (via fremmednøgler)?

Snak med læreren om din udkast før du opretter databasen.

# Opret klasser

# Skriv CRUD-funktioner

* Skriv de CRUD-metoder du har brug for.
* Tilføj test data manuelt i database tabeller.
* Test dine databasefunktioner.

Send læreren en Teams Chat besked: *jeg er færdig med ”skriv CRUD-funktioner”*

# Opret GUI

Følg kravspecifikationerne nøjagtig!

En god fremgangsmåde er:

1. Blev klar over hvilke GUI-objekter du har brug for.
2. Byg GUI med disse objekter.
3. Opret (tomme) funktioner, som bliver udført ved GUI-interaktioner (Klik, Udvalg, …).
4. Tilføj kode til disse funktioner og skriv funktioner til funktionalitets-laget.

Send læreren en Teams Chat besked: *jeg er færdig med ”opret GUI”*

# Milepæl 1: Læsning fra database, GUI til container

I dette kapitel opretter vi GUI'en til Container-klassen, opretter databasen med nogle testdata og læser disse data ind i GUI'en.

## danskcargo\_data.py

I denne fil

Definer databasens datastruktur med hjælp af biblioteket sqlalchemy.

Tilføj dine Data-Klasser en funktion \_\_repr\_\_(), så objekter af klasserne kan udskrives i en læsbar form.

def **convert\_to\_tuple**(self): # Convert Container to tuple

Definer nu en funktion, der finder ud af, om et objekt i klassen indeholder gyldige data.

def **valid**(self):

Som modstykke til den funktion, der konverterer klasseobjekter til en tupel, har du også brug for en funktion, der konverterer tupler til klasseobjekter.

def **convert\_from\_tuple**(tuple\_): # Convert tuple to Container

Vi bruger SQLite her, fordi denne database kræver ingen ekstra installation eller separate programmer. Ved hjælp af den globale variabel Database fortæller du sqlalchemy, hvilken type SQL-database du bruger (her SQLite), hvor databasen er placeret, og hvad den hedder.

Database = *'sqlite:///danskcargo.db'* # first part: database type, second part: file path

For at teste databasen, før du er færdig med at skrive GUI'en, skal du oprette nogle testdata.

def **create\_test\_data**(): # Optional. Used to test database functions before gui is ready.

Denne funktion returnerer en liste med alle poster i en tabel i vores SQL-database. Her er ”classparam” navnet på en af vores dataklasser (Container, Aircraft, Transport). Via attributten \_\_tablename\_\_ i dataklassen kender funktionen navnet på den pågældende tabel i SQL-databasen.

def **select\_all**(classparam): # https://docs.sqlalchemy.org/en/14/tutorial/data\_select.html

# return a list of all records in classparams table

Den følgende funktion fungerer på samme måde som den foregående funktion select\_all, men SQL-forespørgslen er desuden begrænset til et bestemt ID (her primærnøglen). Derfor returnerer funktionen kun én post.

def **get\_record**(classparam, record\_id): # https://docs.sqlalchemy.org/en/14/tutorial/data\_select.html

hovedprogrammet :

create\_test\_data()

print(select\_all(Container))

## danskcargo\_gui.py

tegn en udkast

hvilke widget typer benytter du?

Hvad er widget-hierarkiet? Dvs., besvar for hvert widget hvilken overordnede widget den er tilknyttet.

Hvilke regler følger dine widget-navne?

For det første beskæftiger vi os kun med den venstre tredjedel af GUI'en

### Funktioner til Entry Widgets

Først defineres de funktioner, der læser, tømmer og fylder dine indtastningswidgets med tekst.

### Link widgets med funktioner

Tilføj parameteren command til definitionen af en button

Med metoden bind kan du for treeviews (og nogle andre widgets) definere, hvilken funktion der skal udføres ved en bestemt hændelse

### Fælles funktioner

Den første funktion læser alle poster fra databasen og skriver de gyldige (ikke slettede) poster ind i treeviewen.

def **read\_table**(tree, class\_): # fill tree from database

Den anden funktion tømmer en dataliste (treeview).

def **empty\_treeview**(tree): # Clear treeview table

Tilføj en linje til hovedprogrammet, der initialiserer GUI'en ved at fylde treeviewen fra databasen.

Klik på nogle linjer i treeviewen og prøv knappen "Clear Entry Boxes".

GUI'en er nu klar, du kan læse data fra databasen til dit treeview, du kan kopiere data fra treeviewen til entry widgets med et enkelt klik, og du kan slette indholdet af entry widgets med et enkelt klik.

Nu har du nået den første milepæl. Du kan sammenligne din løsning med filerne i mappen \milestones\danskcargo\10\_S2040\_gui\_exercise4.

# Milepæl 2: Create, update, delete i datenbasen

I dette kapitel skriver vi funktioner til oprettelse, opdatering og sletning af poster i databasen.

Hard vs. Soft delete

def **delete\_soft\_container**(container):

# Milepæl 3: Opret, opdatering, sletning i GUI'en

I det foregående kapitel tilføjede du funktioner til oprettelse, opdatering og sletning af poster i databasen.

I dette kapitel udvider vi GUI'en til at kalde disse funktioner fra GUI'en.

Nu skal knapperne i den grafiske brugergrænseflade forbindes med de funktioner, vi lige har skrevet.

# Milepæl 4: Aircraft og Transport, funktionslag

kopiere koden for klassen Container to gange og derefter tilpasse kopien til henholdsvis klassen Aircraft og klassen Transport.

## danskcargo\_func.py

Tjek 1: Kan der bookes flere passagerer på bussen uden at overskride dets maksimale transportkapacitet?

# Gør dit program mere robust

Find et eller to eksempler hvor brugeren kunne gøre noget forkert og ville crashe programmet. Forhindre at programmet crasher ved denne fejlbetjening.

Send læreren en Teams Chat besked: *jeg er færdig med ”gør programmet mere robust”*