|  |
| --- |
|  |
| Eksamensopgave |
| Application Programming - S2 |

|  |  |
| --- | --- |
| Formalia: |  |
| Afdeling | AspIT Storkøbenhavn |
| Underviser | Uli Scheuss |
| Censor | Jens Clausen |
| Forløb | S2-2-M1-2022 |
| Opgaven udleveres | Onsdag den 14. september 2022 kl. 8:45. |
| Opgaven afleveres | Torsdag den 15. september 2022 kl. 15:00. |
| Der arbejdes følgende dage i tidsrummet 8:45 – 15:00 | 14. og 15. september 2022 |

Indholdsfortegnelse

[1 Problemformulering 3](#_Toc113306319)

[2 Kravspecifikation 4](#_Toc113306320)

[2.1 Database 4](#_Toc113306321)

[2.2 Basisprogram 5](#_Toc113306322)

[2.3 Upgrade 5](#_Toc113306323)

[3 Eksempel 6](#_Toc113306324)

# Problemformulering

Det nye danske fragt-airline DanskCargo har købt nogle flyvemaskiner og på deres gård står allerede container fyldt med varer, som skal transporteres fra København til forskellige byer.

DanskCargo må desværre ikke transportere varer tilbage til København.

DanskCargo har nu brug for et program, som administrerer, hvornår hvilket fly transporterer hvilke containere hvorhenne.

Til det formål skal du oprette databasen og skrive basisprogrammet og GUI.

Derefter programmer også programmets upgrade (kapitel 2.3).

# Kravspecifikation

Programmet skal skrives i Python, skal gemme oplysninger i en SQL-database, skal benytte sqlalchemy for object–relational mapping og den skal have en tkinter-GUI.

Benyt lagdeling for at adskille

* interaktion med databasen
* programmets funktionalitet
* interaktion med GUI

## Database

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

Om containere:

* Bruttovægt (i kilogram)
* Lufthavnen, den skal hen

Om fly:

* Maksimale samlede vægt af containere, der kan transporteres på en gang (i kilogram)
* Registreringsnummeret

Om transporter:

* Datoen
* Fly
* Container

## Basisprogram

Containere, fly og transporter kan oprettes, redigeres og slettes i GUIen. (Sletning kunne være fuldstændigt eller logisk. Du beslutter hvad giver mere mening.)

## Upgrade

### Funktionelle tjek

Hvert fly kan på en dag bare udføre en transport og vende tilbage til København. Før man kan gemme en ny transport skal det altså tjekkes om den nye container skal til den samme lufthavn som de containere, der er allerede booket på samme fly og samme dato.

Før man kan gemme en ny transport skal det også tjekkes om flyet har nok frie kapacitet tilbage den dag.

### Robusthed

Programmet skal være robust i forhold til fejlagtigt betjening (i det mindste for nogle eksempler). Fang (fx. med try/except) i det mindste de følgende fejlbetjeninger:

* Klik på ”Gem fly” selvom kapaciteten mangler eller er negativ
* Klik på ”Gem transport ” selvom ingen fly blev udvalgt

### Unit test

Skriv en unit test. Den skal vise at programmet forhindrer bookning af en transport hvis flyet ikke har nok kapacitet tilbage (containeren er for tung).

# Eksempel

I databasen findes 4 container, et fly og en transport:

|  |  |  |
| --- | --- | --- |
| Id | Vægt | Mål |
| 11 | 1200 | Oslo |
| 12 | 700 | Helsinki |
| 13 | 1800 | Helsinki |
| 14 | 1000 | Helsinki |

Tabel 1: Container

|  |  |  |
| --- | --- | --- |
| Id | Kapacitet | RegNummer |
| 20 | 2000 | OY-CBS |

Tabel 2: Fly

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Dato | FlyId | ContainerId |
| 3 | 25/11/2022 | 20 | 12 |

Tabel 3: Transport

Hver transport relaterer til en container og et fly.

Hvis brugeren nu forsøgte at booke container 11 på samme fly og på samme dato som transport 3 har ville det provokere en fejlmelding fordi fly 20 er allerede booket til Helsinki den dag og kan derfor ikke til Oslo den samme dag.

At booke container 13 på samme fly på samme dato ville heller ikke lykkes fordi den er for tungt - flyet har ikke nok kapacitet til det.

Container 14 må godt bookes på same fly på same dato:

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Dato | FlyId | ContainerId |
| 3 | 25/11/2022 | 20 | 12 |
| 4 | 25/11/2022 | 20 | 14 |

GUI kunne fx se sådan ud:

