Oppgave 4 Take - away tjeneste (45 poeng)

Du skal skrive (deler av) et program for en take - away tjeneste som tar imot bestillinger fra kunder på internett, gjennom et enkelt, terminalbasert grensesnitt. Programmet skal holde rede på en rekke faste kunder med telefonnummer og hvilke typer mat / ingredienser (innholdsstoffer) denne kunden ikke ønsker (for eksempel gluten, meieriprodukter eller svinekjøtt).

Videre skal programmet kjenne til hvilke matretter som kan leveres innenfor en rekke

Kategorier – som for eksempel forretter, hovedretter, eller desserter. For hver rett skal det lagres hvilke innholdsstoffer retten har som kan være problematiske for kunder. I denne oppgaven kan du anta at ordene som blir brukt om innholdsstoffer kunden vil unngå, og ordene som blir brukt om innholdsstoffene i en matrett, hentes fra samme vokabular slik at det er enkelt å sammenligne.

Programmet bruker disse dataene til å presentere en tilpasset meny til hver kunde, der alle

matrettene i menyen tilfredsstiller kundens krav til innhold.

Den delen av systemet du trenger å kjenne til består foruten TakeAway av klassene Rett, Kategori, Meny ogKunde. Du skal skrive alle metoder som er beskrevet for hver klasse om det ikke eksplisitt er oppgitt at de ikke skal skrives. Det anbefales å lage en tegning av strukturen for eget bruk

1. 10 poeng. Skriv klassen **Rett –** (oppg 18)

Klassen har et navn (streng), en pris (flyttall), og en liste (kan være tom) av innholdsstoffer (strenger). Alle instansvariable får startverdi fra parametere til konstruktøren\_\_init\_\_. Klassens grensesnitthar følgende metoderi tillegg til konstruktøren:

* **sjekkInnholdOK** har en parameter med en liste av innholdsstoffer. Metoden sjekker om noen av innholdsstoffene i parameteren fins i rettens liste over innhold. Hvis metoden finner et treff, returnerer den False. Hvis ingen av innholdsstoffene i parameteren finnes i retten,skal metoden returnere True.
* **\_\_str\_\_** returnerer en streng med rettens navn, pris og alle innholdsstofferpå en lesbar form.

1. 5 poeng. Skriv klassen **Kategori** – (oppg 19)

Klassen har to instansvariable; kategorinavn og en liste med referanser til Rett-objekter. Begge får verdi fra parametere til konstruktøren. Grensesnittet til klassen Kategori inneholder dessuten følgende metode:

* **hentOkRetter** med én parameter: En liste med innholdsstoffer som en kunde ønsker å unngå. Metoden går gjennom kategoriens liste over retter, og lager en ny liste med referanser til retter som ikke inneholder noen av innholds stoffene som skal unngås. Denne nye listen returneres.

(Du trenger ikke lage kopier av Rett-objektene som skal være med i den nye listen, kun referere til de samme objektene som instansvariabelen)

1. 10 poeng. Skriv klassen **Meny** – (oppg 20)

Klassen har én instansvariabel. Denne representerer hele menyen i en ordbok med alle kategorinavnene som nøkler og referanser til Kategori-objekter som verdier. Klassens konstruktør har én parameter;en liste med alle kategorinavni menyen. Alle data om en kategori leses fra en fil der filnavnetbestår av kategorinavnet etterfulgt av ".txt". Konstruktøren skal bygge opp menyen ved å kalle på den private (non-public) metoden \_lesKategoriFil som har et filnavn som parameter og returnerer et ferdig kategoriobjekt.   
Du skal ikke skrive metoden \_lesKategoriFil.

Foruten konstruktøren har klassen én metode:

* **hentRedusertMeny** tar én parameter; en liste over innholdsstoffer som skal unngås. Metoden går gjennom hele menyen kategori for kategori, og bygger opp en annen, redusert meny der ingen av rettene inneholder uønskede innholdsstoffer. Kategorier som ikke har noen retter igjen skal ikke være med i den reduserte menyen som returneres av metoden.

1. 8 poeng. Skriv klassen **Kunde** – (oppg 21)

Klassen har to instansvariable: Telefonnummer til kunden (en streng), og en liste med innholdsstoffer (strenger) som kunden ønsker å unngå (for eksempel på grunn av allergi). Begge instansvariable får verdier fra parametere til konstruktøren.   
Videre har klassen en metode:

* **velgRetter**, somtar et Meny-objektsom parameterog kaller på hentRedusertMeny på dette objektet for å få en skreddersydd meny for kunden. Deretter presenteres den reduserte menyen på terminalen for kunden, en kategori av gangen. Kunden velger en rett fra kategorien ved å taste inn navnet på retten – eller en tom linje for å hoppe videre til neste kategori. Hver ikke-tomme linje tastet inn av kunden lagres som en streng, og metoden returnerer disse strengene i en liste. Kundens input skal altså ikke sjekkes mot retter i systemet – dette for at kunden kan legge inn egne beskjeder til kjøkkenet om mengde eller tilberedning

1. 7 poeng. Skriv klassen **TakeAway** – (oppg 22)

Klassen **TakeAway** har to instansvariable; en referanse til et objekt av klassen Meny, og en ordbok med kunder der telefonnummer er nøkkel og verdiene er referanser til Kundeobjekter. Konstruktøren har to parametere; en liste av kategorinavn og navn på en kundefil, og skal bygge opp en ferdig meny en og kundekatalog. Metoden \_lesKundefil returnerer en kundekatalog med alle kunder og skal kalles i konstruktøren for TakeAway, men ikke skrives av deg. Forut en konstruktøren er grensesnittet til klassen som følger:

* **betjenKunde** med parameter telefonnummer for en kunde som har tatt kontakt. Metoden kaller på metoden velgRetter for riktig kunde (du kan anta at alle kunder er registrert på forhånd). Deretter kalles den private (non-public) metoden \_lagOgLeverMat med bestillingen fra velgRetter.
* **\_lagOgLeverMat** har en parameter bestilling og skal i denne versjonen kun skrive ut kundens bestilling (navn på alle rettene) på terminalen. Metoden \_lagOgLeverMat skal skrives av deg.

1. 5 poeng. Skriv et **hovedprogram** – (oppg 23)

Hovedprogrammet skal gjørefølgende:

* Starte en take-away tjeneste med kategoriene "Forretter", "Hovedretter" og "Desserter" på menyen, og en kundekatalog på filen "Kunder.txt". Du kan anta at alle nødvendige datafiler finnes.
* Beom telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

**Oppgave 5) 6 poeng**

Skriv en funksjon **godkjenn(aldre)** som tar som argument en liste av lister med aldre for personer i familier. Hver familie er representert med en liste av aldre for alle medlemmene i familien i vilkårlig rekkefølge. En familie med tre medlemmer av alder 30, 10 og 2 år, kan dermed f.eks. være representert av en liste*[10,2,30]*. Alle aldre er heltall. Funksjonen godkjenn skal ta inn en liste av slike lister (altså flere familier). Om man i tillegg til nevnte familie har en annen familie med aldre 20 og 1, vil dette altså kunne være representert som *[  [10,2,30], [20,1] ]*.

Skriv funksjonen godkjenn slik at den sjekker at alle familier har minst én myndig person, altså at alle lister har minst en verdi som er større eller lik 18. Så lenge alle familier har minst én myndig person, skal funksjonen returnere True. Dersom én eller flere familier mangler myndig person (at det for minst én av listene er slik at alle verdier er under 18) skal den returnere False.

For eksempel skal kallet *godkjenn([  [10,2,30], [20,1] ])* returnere *True*, mens kallet *godkjenn([ [10,2,30], [10,1] ])* skal returnere *False*.