Informasjon

Eksamen i IN1000 og IN1001 høsten 2018

Tid

30. november kl. 14.30 (4 timer)

Faglærere vil besøke lokalet ca kl 15-16.

Oppgavene

Oppgave **1a-f** er kortsvarsoppgaver som rettes automatisk. Pass på å skrive inn nøyaktig svar med riktig type (ikke desimalpunktum eller -komma for heltall). Tekster kan skrives inn direkte (som her) eller med doble anførselstegn ("som her"), men ikke med enkle anførselstegn ('som her').

Oppgave **2a-d** er flervalgsoppgaver. Her får man det angitte antall poeng om man svarer riktig; ved galt svar eller intet svar får man 0 poeng.

Dersom du mener en programmeringsoppgave er uklar, kan du gjøre egne forutsetninger og beskrive disse. Du kan også legge til egne funksjoner eller metoder ved behov, med begrunnelse. Videre kan du bruke løsninger fra tidligere deloppgaver selv om du selv ikke har skrevet disse.

Tillatte hjelpemidler

Alle trykte og skrevne hjelpemidler. Ingen elektroniske.

1(a) # 1a) 1 poeng

Hva er verdien til tall etter at følgende kode er utført?

```
tall = (3+1) * 2
tall = tall - 5
```

Maks poeng: 1

1(b) # 1b) 1 poeng

Hva er verdien til variabelen tekst etter at følgende kode er utført?

```
tall = 7
tekst = "a"
if tall>10:
  tekst = tekst + "b"
elif tall<5:
  tekst = tekst + "c"
else:
  tekst = tekst + "d"
```

Maks poeng: 1

1(c)

1c) 2 poeng

Hva er verdien til variabelen a etter at følgende kode er utført?

```
a = 0
for b in [2,4,1]:
    a = 2*a + b
```

Maks poeng: 2

1(d)

1d) 2 poeng

Hva skrives ut på skjermen når følgende kode utføres?

```
tallene = [ ]
a = 0
b = 1
while a<4:
    tallene.append(b)
    b = b*2
    a = a+1
print(tallene[0] + tallene[3])
```

Maks poeng: 2

1(e)

1e) 2 poeng

VI har en funksjon kalkuler som vist nedenfor:

```
def kalkuler(tall):
    tall = tall + 1
    return tall*2
```

Hva skrives ut på skjermen når følgende kode utføres?

```
print( kalkuler(2) + kalkuler(4) )
```

Maks poeng: 2

1(f)

1f) 3 poeng

Hva skrives ut på skjermen når følgende kode utføres?

```
class Tall:
    def __init__(self, a):
        self._a = a
```

```
        def m1(self, c):
            self._a = self._a + c
        def m2(self):
            self._a = self._a * 2
        def m3(self):
            return self._a + 10

    t1 = Tall(5)
    t2 = Tall(2)
    t1. m2()
    t2.m1( t1.m3() )
    print( t2.m3() )
```

Maks poeng: 3

2(a) 2a) 2 poeng

#Anta at klassen Person er definert slik:

```
class Person:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder
    def bursdag(self):
        self._alder += 1
    def hentAlder(self):
        return self._alder
    def settAlder(self, nyAlder):
        self._alder = nyAlder
```

Hva skrives ut når følgende kode kjøres?

```
far = Person("Gjert", 48)
trener = far
trener.bursdag()
print( far.hentAlder() )
```

Velg ett alternativ

- 49
- 60
- 48

Maks poeng: 2

2(b) 2b) 1 poeng

#Gitt klassen Person definert som i oppgave 2a

Hva skrives ut når følgende kode kjøres?

```
far = Person("Gjert", 48)
trener = far
trener.settAlder(60)
print( far.hentAlder() )
```

Velg ett alternativ

- 60
- 49
- 48

Maks poeng: 1

2(c) **2c) 1 poeng**

#Gitt klassen Person definert som i oppgave 2a.

Hva skrives ut når følgende kode kjøres?

```
far = Person("Gjert", 48)
trener = far
trener.bursdag()
trener = Person("Tone", 60)
print( far.hentAlder() )
```

Velg ett alternativ

- 60
- 48
- 49

Maks poeng: 1

2(d) **2d) 1 poeng**

#Gitt klassen Person som definert i oppgave 2a.

Hva skrives ut når følgende kode kjøres?

```
def feiring(p):
    p.bursdag()

far = Person("Gjert", 48)
feiring(far)
print( far.hentAlder() )
```

Velg ett alternativ

- 49
- 48
- 60

Maks poeng: 1

3(a) **3a) 5 poeng**

Skriv en funksjon **vinnerlag(hjemmelag, bortelag, hjemmemaal, bortemaal)** som tar som argument navn og antall mål for hjemme- og bortelag, og returnerer navnet på laget som vant (høyest antall mål). Om de to lagene har likt antall mål skal funksjonen returnere strengen "uavgjort". Du kan gå ut fra at argumentene til hjemmelag og bortelag er strenger (type str), og at argumentene til hjemmemaal og bortemaal er heltall (type int). For eksempel skal kallet *vinnerlag("Brann", "Molde", 2,3)* returnere "Molde", mens kallet *vinnerlag("Brann", "Molde", 2,2)* skal returnere "uavgjort".

Skriv ditt svar her...

```
1
```

Maks poeng: 5

3(b) 3b) 4 poeng

Skriv en funksjon **forkort_lagliste(lagliste)** som tar som argument en liste av strenger som er lagnavn og returnerer en ny liste hvor ingen lagnavn finnes flere ganger i lista. Med andre ord, dersom samme streng opptrer flere ganger i lista som funksjonen mottar som argument, skal denne bare opptre én gang i lista som blir returnert. For eksempel skal kallet f*orkort_lagliste(["Brann", "Molde", "Brann"])* returnere en liste *["Brann", "Molde"]*.

Skriv ditt svar her...

```
1
```

Maks poeng: 4

3(c)

3c) 3 poeng

Skriv en funksjon **legg_inn_null_maal(lagliste)** som tar som argument en liste av strenger (lagnavn) og returnerer en ordbok som har hver streng fra lista (hvert lagnavn) som nøkkelverdi, med heltallsverdien 0 som innholdsverdi. For eksempel skal kallet *legg_inn_null_maal(["Brann", "Molde"])* returnere en ordbok *{"Brann" : 0, "Molde" : 0}*.

Skriv ditt svar her...

```
1
```

Maks poeng: 3

3(d) # 3d) 6 poeng

Skriv en funksjon **ekstraher_lagliste(fn)** som tar et filnavn som argument og returnerer en liste med alle lagnavn den finner i filen. Funksjonen skal lese gjennom en fil (gitt av filnavnet mottatt som argument) hvor hver linje består av et navn på hjemmelag, navn på bortelag, antall mål hjemmelag, antall mål bortelag (separert med mellomrom). Funksjonen skal returnere en liste som inneholder alle lagnavn som finnes i filen (som enten hjemmelag eller bortelag i ovennevnte format). Det gjør ikke noe (er uten betydning) om hvert lag opptrer flere ganger i den returnerte lista. Rekkefølgen av lagene i lista som returneres er også uten betydning.

En fil som skal leses gjennom kan altså se ut for eksempel som følger:
brann molde 2 0
sarpsborg molde 1 1

Funksjonen skal for dette eksempelet returnere en liste som inneholder lagnavnene brann, molde og sarpsborg, hvor eventuelle repetisjoner er uproblematisk - altså blir både ["brann", "molde", "sarpsborg"] og ["brann", "molde", "sarpsborg", "molde"] godtatt som returverdi.

Skriv ditt svar her...

```
1
```

Maks poeng: 6

3(e) # 3e) 7 poeng

Skriv en funksjon **regn_poengsum(fn)** som tar et filnavn som argument og returnerer en ordbok med antall poeng for hvert lag ut fra kampresultat funnet i filen. Filen har samme format som i oppgave 3d, dvs hvor hver linje består av et navn på hjemmelag, navn på bortelag, antall mål hjemmelag, antall mål bortelag (separert med mellomrom). Funksjonen regn_poengsum skal fortrinnsvis initielt kalle på funksjonene ekstraher_lagliste, forkort_lagliste og legg_inn_null_maal (fra hhv oppgave 3d, 3b og 3c) for å konstruere en ordbok med null poeng for alle lag oppført i filen. Deretter skal den gå gjennom alle kampresultat (linjer) i filen og gi 3 poeng for vinnende lag, 0 poeng for tapende lag, samt 1 poeng til hvert lag i tilfeller av uavgjort. Man kan gjerne kalle på funksjonen vinnerlag fra oppgave 3a for å hjelpe med utregning av poeng, men dette tillegges ikke vekt.

(Du kan kalle på funksjoner/prosedyrer fra andre deloppgaver ut fra den beskrevne funksjonaliteten, uavhengig av om du selv har løst disse deloppgavene eller ikke)

Skriv ditt svar her...

```
1
```

Maks poeng: 7

3(f) **3f) 5 poeng**

Skriv en funksjon **gull(lagoversikt)** som tar som argument en ordbok med lagnavn og antall mål (strenger som nøkkelverdier og heltall som innholdsverdier) og returnerer navnet på laget med høyest poengsum (nøkkelverdien med høyeste tilhørende innholdsverdi). Du kan anta at ett lag har høyest poengsum (ikke to lag med samme poengsum), og at det finnes noen lag med positiv poengsum i ordboken mottatt som argument. For eksempel skal kallet *gull({"Brann" : 2, "Molde" : 3})* returnere *"Molde"*.

Skriv ditt svar her...

```
1
```

Maks poeng: 5

3(g)

3g) 3 poeng

Skriv en prosedyre **finn_gull(fn)** som tar et filnavn som argument og skriver navnet på laget med flest poeng ut til terminalen. Filen har samme format som i oppgave 3d. Poeng regnes ut basert på vinnende og tapende lag for hver kamp som beskrevet i oppg 3e. Som i oppg 3f kan du anta at ett lag har høyest poengsum (ikke to lag med samme poengsum), og at det finnes minst ett lag med positiv poengsum. Du kan benytte deg av kall på hvilke funksjoner/prosedyrer du ønsker fra deloppgaver 3a-3f for å lage en enklest mulig løsning for prosedyren finn_gull (du kan kalle på funksjoner/prosedyrer fra andre deloppgaver ut fra den beskrevne funksjonaliteten, uavhengig av om du selv har løst disse deloppgavene eller ikke).

Skriv ditt svar her...

```
1
```

Maks poeng: 3

4(a)

4a) 10 poeng

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

Skriv svar på deloppgave a) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 10

4(b)

4b) 5 poeng

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

Skriv svar på deloppgave b) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 5

4(c)

4c) 10 poeng

4(c) **4c) 10 poeng**

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

NB! Menyen som returneres fra metoden hentRedusertMeny skal representeres som en ordbok av kategorier, ikke et objekt av klassen Meny.

Skriv svar på deloppgave c) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 10

4(d) **4d) 8 poeng**

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

Skriv svar på deloppgave d) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 8

4(e) 4e) 7 poeng

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

Skriv svar på deloppgave e) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 7

4(f)

4f) 5 poeng

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave).

Skriv svar på deloppgave f) fra vedlegget her:

Skriv ditt svar her...

```
1
```

Maks poeng: 5

5

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*.

Skriv ditt svar her...

```
1
```

Maks poeng: 6

Question 4.a
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

Question 4.b
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

Question 4.c
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

Question 4.d
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

Question 4.e
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.

Question 4.f
Attached

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** og **Kunde**. 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.

a) 10 poeng. Skriv klassen Rett

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 grensesnitt har følgende metoder i 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 innholdsstoffer på en lesbar form.

b) 5 poeng. Skriv klassen Kategori

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 innholdsstoffene 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).

c) 10 poeng. Skriv klassen Meny

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 kategorinavn i menyen. Alle data om en kategori leses fra en fil der filnavnet består 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.

d) 8 poeng. Skriv klassen Kunde

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**, som tar et Meny-objekt som parameter og 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.

e) 7 poeng. Skriv klassen TakeAway

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. Foruten 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.

f) 5 poeng. Skriv et hovedprogram

Hovedprogrammet skal gjøre følgende:

- 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.
- Be om telefonnummer (på terminalen) for, og betjene, en og en kunde inntil bruker gir inn en tom streng som telefonnummer - dette avslutter programmet.