Skriftlig eksamen i IN1000
2021 HØST
Varighet: 3.12.2021, 9:00 til 3.12.2021, 13:00.
Det er viktig at du leser denne forsiden nøye før du starter.

Del 1&2 (samlet), del 3 og del 4 må bestås hver for seg. Pass på at du bruker tiden slik at du viser hva du kan på hver av disse.

Generell informasjon:

- Svar på spørsmål som kan være relevante for flere blir publisert under eksamen på denne nettsiden. Det er derfor viktig at du sjekker denne siden underveis og før du kontakter faglærer med spørsmål.
- Besvarelsen din skal reflektere ditt eget, selvstendige arbeid og skal være et resultat av din egen læring og arbeidsinnsats.
- Alle hjelpemidler er tillatt ved skriftlig hjemmeeksamen. Dersom du gjengir tekst fra bøker, forelesninger, nettartikler eller lignende, så må det henvises til disse kildene i besvarelsen for å unngå mistanke om ulovlig tekstlikhet. Dette gjelder også dersom du oversetter tekst fra andre språk.
- Du er selv ansvarlig for å sørge for at eksamensbesvarelsen din ikke er tilgjengelig for andre under eksamenstiden, hverken fysisk eller digitalt.
- Husk at besvarelsen skal være anonym, du skal ikke oppgi hverken ditt eller medstudenters navn.
- Om du vil trekke deg fra eksamen, trykk på hamburgermeny oppe til høyre i Inspera og velg "Jeg vil trekke meg".

Samarbeid under eksamen:
Det er ikke tillatt å samarbeide eller kommunisere med andre under eksamen. Samarbeid og kommunikasjon vil bli betraktet som forsøk på fusk. Det blir gjort plagiatkontroll av alle innleverte eksamener der tekstlikhet/ strukturlikhet mellom besvarelser blir sjekket. Om du bruker notater som er utarbeidet i samarbeid med andre før eksamen, kan dette gi treff i en plagiatsjekk. Slik tekstlikhet kan bli betraktet som lav selvstendighet eller forsøk på fusk. Unngå derfor klipp/lim fra notater laget i samarbeid med andre.

Fusk:
Les om hva som regnes som fusk på UiOs nettsider.

Kontaktinfo:
Brukerstøtte eksamen (tekniske/ praktiske/ administrative problemer)
Faglige spørsmål om oppgaven sendes i epost til faglærere (trykk på lenken eller send til siriamj@ifi.uio.no **og** geirksa@ifi.uio.no). Legg ved bilde av oppgaven du lurer på.

1(a)

```
# Ordbok med statene i USA:
ordbok = {                                      \
"AL":"Alabama", "AK":"Alaska", "AZ":"Arizona", "AR":"Arkansas", "CA":"California", \
"CO":"Colorado", "CT":"Connecticut", "DE":"Delaware", "FL":"Florida", \
"GA":"Georgia", "HI":"Hawaii", "ID":"Idaho", "IL":"Illinois", "IN":"Indiana", \
"IA":"Iowa", "KS":"Kansas", "KY":"Kentucky", "LA":"Louisiana", \
"ME":"Maine", "MD":"Maryland", "MA":"Massachusetts", "MI":"Michigan", \
"MN":"Minnesota", "MS":"Mississippi", "MO":"Missouri", "MT":"Montana", \
"NE":"Nebraska", "NV":"Nevada", "NH":"New Hampshire", "NJ":"New Jersey", \
"NM":"New Mexico", "NY":"New York", "NC":"North Carolina", "ND":"North Dakota", \
"OH":"Ohio", "OK":"Oklahoma", "OR":"Oregon", "PA":"Pennsylvania", \
"RI":"Rhode Island", "SC":"South Carolina", "SD":"South Dakota", "TN":"Tennessee", \
"TX":"Texas", "UT":"Utah", "VT":"Vermont", "VA":"Virginia","WA":"Washington", \
"WV":"West Virginia", "WI":"Wisconsin", "WY":"Wyoming", "AS":"American Samoa", \
"DC":"District of Columbia", "FM":"Federated States of Micronesia", "GU":"Guam", \
"MH":"Marshall Islands", "MP":"Northern Mariana Islands", "PW":"Palau", \
"PR":"Puerto Rico", "VI":"Virgin Islands" }

b= []
for a in ordbok:
    b.append(a)

for i in range(len(ordbok)):
    a = b[i]
    ordbok[a] = ordbok[b[i]]

print(ordbok[b[1]])
```

Hva skrives ut på terminalen av koden i bildet? Fyll ut svarboksen med nøyaktig samme tegn:

Maks poeng: 1

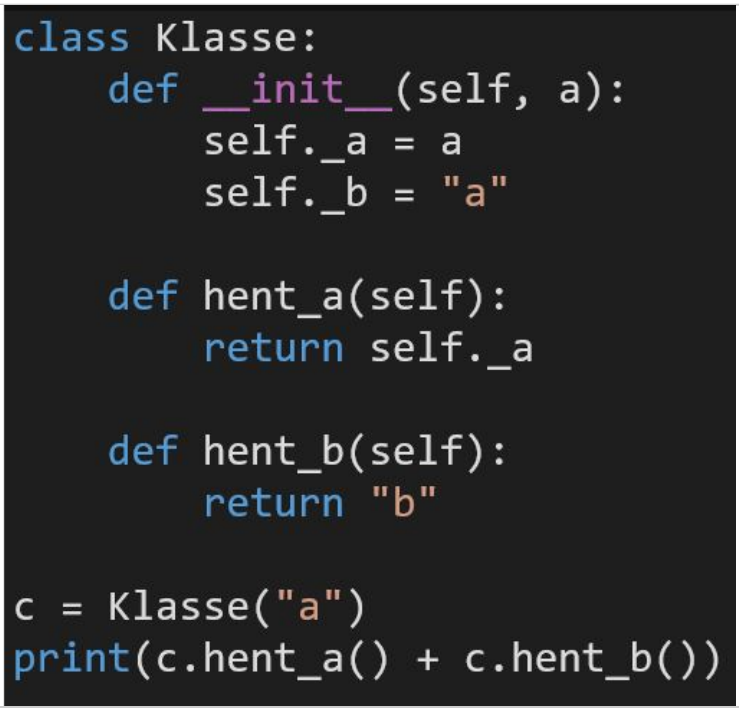
1(b)

```
# Ordbok med statene i USA:
ordbok = {                                      \
"AL":"Alabama", "AK":"Alaska", "AZ":"Arizona", "AR":"Arkansas", "CA":"California", \
"CO":"Colorado", "CT":"Connecticut", "DE":"Delaware", "FL":"Florida", \
"GA":"Georgia", "HI":"Hawaii", "ID":"Idaho", "IL":"Illinois", "IN":"Indiana", \
"IA":"Iowa", "KS":"Kansas", "KY":"Kentucky", "LA":"Louisiana", \
"ME":"Maine", "MD":"Maryland", "MA":"Massachusetts", "MI":"Michigan", \
"MN":"Minnesota", "MS":"Mississippi", "MO":"Missouri", "MT":"Montana", \
"NE":"Nebraska", "NV":"Nevada", "NH":"New Hampshire", "NJ":"New Jersey", \
"NM":"New Mexico", "NY":"New York", "NC":"North Carolina", "ND":"North Dakota", \
"OH":"Ohio", "OK":"Oklahoma", "OR":"Oregon", "PA":"Pennsylvania", \
"RI":"Rhode Island", "SC":"South Carolina", "SD":"South Dakota", "TN":"Tennessee", \
"TX":"Texas", "UT":"Utah", "VT":"Vermont", "VA":"Virginia","WA":"Washington", \
"WV":"West Virginia", "WI":"Wisconsin", "WY":"Wyoming", "AS":"American Samoa", \
"DC":"District of Columbia", "FM":"Federated States of Micronesia", "GU":"Guam", \
"MH":"Marshall Islands", "MP":"Northern Mariana Islands", "PW":"Palau", \
"PR":"Puerto Rico", "VI":"Virgin Islands" }
liste = []
for a in ordbok:
    liste.append(a)
i = 0
for b in liste:
    if ordbok[b] == "Alabama":
        print(liste[i])
    i += 1
```

Hva skrives ut på terminalen av denne koden? Svar her (nøyaktig de tegnene som skrives ut):

Maks poeng: 1

1(c)



```
class Klasse:
    def __init__(self, a):
        self._a = a
        self._b = "a"

    def hent_a(self):
        return self._a

    def hent_b(self):
        return "b"

c = Klasse("a")
print(c.hent_a() + c.hent_b())
```

Skriv nøyaktig de tegnene som skrives ut på terminalen i svarfeltet:

Maks poeng: 1

1(d)

```
s = ""
if True or False:
    s += "a"
    if True and False:
        s += "b"
        s += "c"
    if False or not False:
        s += "d"
a = False
b = not False
if a or b:
    if a and b:
        s += "e"
if not False and not False:
    s += "f"
    if not True and True:
        s += "g"
    if True or not True:
        s += "h"
print(s)
```

Skriv nøyaktig de tegnene som skrives ut i svarfeltet:

Maks poeng: 1

1(e)

Hva blir resultatet av å kjøre denne koden?

- 4 skrives ut
- 6 skrives ut
- IndexError: list index out of range
- 5 skrives ut

```
a=1
b=2
c=3
listeA = [a,b,c]
listeB = [1,2,3]
listeC = listeA
listeA.append(5)
listeC.append(6)
print(listeC[4])
```

Maks poeng: 1

1(f)

```
class Klasse:
    def __init__(self, a, b):
        self._a = b
        self._b = a

    def hent_a(self):
        return self._a

    def hent_b(self):
        return self._b

c = Klasse("a", "b")
print(c.hent_a() + c.hent_b())
```

Skriv nøyaktig de tegnene som skrives ut på terminalen i svarfeltet:

Maks poeng: 1

2(a) Gitt klassen Rektangel:

```
class Rektangel:
    def __init__(self, lengde, bredde):
        self._lengde = lengde
        self._bredde = bredde
        self._areal = lengde*bredde

    def sett_lengde(self, lengde):
        self._lengde = lengde

    def sett_bredde(self, bredde):
        self._bredde = bredde

    def hent_areal(self):
        return self._areal

    def oppdater_areal(self):
        self._areal = self._lengde * self._bredde
```

Fyll inn det som må stå i boksen nedenfor for at koden nedenfor vil skrive ut **6 12** på terminalen.

r1= [ ] her.

areal1 = r1.hent_areal()
r1.sett_lengde(4)
r1.oppdater_areal()
areal2 = r1.hent_areal()
print(areal1, areal2)

Maks poeng: 2

2(b) Fyll inn det som må stå i boksen under for at variabelen d2 ville fått verdien **{'a': 3, 'b': 4, 'c': 2}** etter at koden under var kjørt.

```
d1 = {"a": [1, 2, 6], "b": [1, 4, 2, 8], "c": [1, 5]}
d2 = {}
for key in d1:
    value = d1[key]
    d2[key] =
```

Maks poeng: 2

2(c)

```
def funk(ordbok, ord, tall):
        ordbok[ord] = tall
        return 2*tall

bokA = {"a":3, "b":6, "c":8, "d":10}
bokB = {"a":3, "b":6, "c":8, "d":10}
c = "b"
d = c
x = 2
y = funk(bokB, c, 3)
print(bokB[d])
```

Fyll inn ditt svar her:

Maks poeng: 2

2(d)

	Uttrykk som evaluerer til int	Ikke et gyldig uttrykk	Uttrykk som evaluerer til string
for i in range(4):	○	○	○
print(navn)	○	○	○
int(input("Alder: "))	○	○	○
713	○	○	○

Maks poeng: 2

3(a) Et land har følgende regler for reisekarantene:
Dersom en person er vaksinert, trenger personen ikke å gå i karantene (0 dager i karantene)
Dersom en uvaksinert person har vært i et land som har fargekode rød eller oransje, skal peresonen 10 dager i karantene.
Dersom en uvaksinert person har vært i et land som har fargekode grønn, skal personen 3 dager i karantene.

Skriv en funksjon karantene(vaksinert, farge) som returner antall dager (datatype int) en person må i karantene i henhold til reglene over, hvor parameteren vaksinert er enten True eller False (datatype bool) og parameteren farge er enten "roed", "oransje" eller "groenn" (datatype str).
Altså skal for eksempel kallet *karantene(True, "roed")* returnere 0, mens kallet karantene(False, "oransje") skal returnere 10.
Skriv ditt svar her

```
1
```

Maks poeng: 6

3(b)

Skriv en funksjon tell_grader(fag, bsc, msc) som tar inn tre parametre av datatypen tekst (str) og sjekker om en Bsc-grad og/eller en Msc-grad er innen et bestemt fag. Dersom parameteren fag har samme verdi som både parametrene bsc og msc, skal funksjonen returnere 2. Dersom parameteren fag har samme verdi som én av parametrene bsc eller msc skal funksjonen returnere 1. Dersom parameteren fag hverken er lik verdien til parametrene bsc eller msc, skal funksjonen returnere 0.

For eksempel skal kallet *tell_grader("informatikk", "informatikk", "informatikk")* returnere 2, mens kallet *tell_grader("historie", "informatikk", "informatikk")* skal returnere 0.

Skriv ditt svar her

```
1
```

Maks poeng: 6

3(c)

Skriv en funksjon fjern_vokaler(setning, vokal_liste) som tar inn to parametre - en setning (datatype str) og en liste med vokaler (datatype list, hvor hvert element er en str av lengde 1). Funksjonen skal returnere en kopi av teksten setning hvor alle vokaler fra vokal_liste er fjernet fra teksten. Du kan anta at både setning og vokal_liste kun inneholder små bokstaver.

For eksempel skal kallet *fjern_vokaler("ha det fint", ["a", "e", "i", "o", "u"])* returnere teksten "h dt fnt".

Skriv ditt svar her

```
1
```

Maks poeng: 8

3(d) Skriv en funksjon summer_rabatt(vareliste, forpris, nypris) som tar som argument en liste med varenavn (vareliste) , en ordbok med varenavn som nøkler og deres førpris som verdier (forpris), samt en en ordbok med varenavn som nøkler og deres nypris som verdier (nypris). Funksjonen skal regne ut hvor mye rabatt en kunde har fått i sum dersom kunden har kjøpt varene i vareliste (én av hvert element i lista). Rabatten for en gitt vare er forskjellen mellom førpris og nypris for det gitte varenavnet, hvor disse prisene ligger i hver sin ordbok forpris og nypris. Du kan anta at alle varenavn i vareliste finnes som nøkler i både ordboken forpris og nypris, og du kan anta at nypris alltid er lavere eller lik førpris.

For eksempel skal kallet *summer_rabatt(["laptop", "ryggsekk"], {"laptop":5000, "ryggsekk":900}, {"laptop":4000, "ryggsekk":600})* returnere 1300.

Skriv ditt svar her

```
1
```

Maks poeng: 8

3(e) Definer en funksjon **sjekk_reise(reise)** som tar inn en nøstet liste *reise,* hvor hvert element i denne lista igjen er en liste med to elementer av datatype str (et par av reiseopprinnelse og reisemål). Funksjonen skal returnere True dersom *reise* er en gyldig reiserute, og False ellers. Reiseruten er gyldig når hvert reisemål (andre element i en indre liste) er likt reiseopprinnelsen (første element) i den etterfølgende indre listen. Du kan anta at alle elementene i den ytre listen er en indre liste av lengde 2.

For eksempel skal kallet *sjekk_reise([["Spania", "Frankrike"], ["Frankrike", "Norge"]])* returnere True da den beskriver reisen Spania -> Frankrike -> Norge. Kallet *sjekk_reise*([["Russland", "Tyskland"], ["Tyskland","Sverige"], ["Norge", "Belgia"]]) skal derimot returnere False, fordi andre etappe ender i Sverige mens neste etappe starter i Norge. Merk at også listen [["Russland", "Tyskland"], ["Norge", "Tyskland"]] skal returnere False, fordi den andre etappen skulle ha startet i Tyskland (i stedet slutter den der, og er dermed ikke gyldig).

Skriv ditt svar her

```
1
```

Maks poeng: 8

4 I denne oppgaven skal du skrive et større objektorientert system som beskrevet i vedlagte pdf. Du skal skrive løsning til alle deloppgaver i svar-feltet til denne oppgaven. Angi oppgavenummer for hver klasse du skriver.

Du finner også PDF'en HER (klikk på "HER") , dersom du ønsker å åpne den i eget vindu.

Du står fritt til å innføre egne variabler, metoder og klasser utover de som er beskrevet i oppgaveteksten for å løse oppgaven. Husk å unngå særnorske tegn og bokstaver i koden din, og pass på at innrykk i koden blir riktige.

Skriv ditt svar her

```
1
```

Maks poeng: 50

Question 24
Attached

Del 4: Julegavefikseren: En kombinert ønskeliste og juleferiekalender

Totalt 50 poeng

Du kan anta i programmet du skriver at input (argumenter til metodene dine eller input fra fil/ bruker) er lovlige verdier av riktig type, du trenger altså ikke sjekke disse om det ikke nevnes spesielt.

Dolly Duck har tenkt ut en løsning på to problemer: Hun er lei av å få julegaver hun ikke ønsker seg og som bare står og støver ned – og hun har ingenting å gjøre i juleferien etter julaften. Løsningen hennes er en app som du skal få utvikle – der hun kan registrere ønsker om opplevelser og aktiviteter i en ønskeliste, og de som vil gi henne gaver kan velge et ønske fra ønskelisten som så legges i en «juleferiekalender».

I juleferiekalenderen kan hun åpne en ny gave hver dag, fra og med 25. desember og så lenge juleferien varer. «Gavene» som ligger i kalenderen er bare en tekst som beskriver opplevelsen eller aktiviteten, sammen med navnet på giveren – men da vet hun jo hvem hun kan kontakte. En vanlig julekalender kan åpnes hver dag fra 1. til 24. desember – men her skal vi altså lage en kalender som brukes *etter* julaften i stedet.

Julegavefikseren er bygget opp av 5 klasser som du skal skrive:

- En klasse Onskeliste som inneholder ønskelisten (en liste med Onske-objekter) og metoder for å legge inn ønsker, velge et ønske å oppfylle, og for å hente ut en liste med ønsker.
- En klasse Onske med metoder for å sjekke om ønsket passer for en giver eller ikke, for å velge et ønske og for returnere ønsket som en string.
- En klasse Juleferiekalender med gaver for hver dag i juleferien, fra og med 25. desember og et valgfritt antall dager fremover (men aldri mer enn 31 dager). Klassen har metoder for å legge til en ny gave, hente gaven for en bestemt dag, og lese av hvor mange dager ferie juleferiekalenderen er laget for.
- En klasse Gave som inneholder beskrivelsen av gaven og hvem den er fra. Denne informasjonen kan hentes ut som en streng.
- Den siste klassen er Julegavefikseren som tar antall dager i juleferien som parameter til konstruktøren. Klassen har metoder for å lese inn Dollys ønsker i ønskelisten fra fil, for å velge et ønske å oppfylle og legge som en gave i juleferiekalenderen (for givere) og for å åpne en ny dag i kalenderen (for Dolly).

Du trenger ikke skrive noe hovedprogram – dette vil lese inn alle ønsker fra fil og ha et brukergrensesnitt der en som vil gi Dolly en gave oppgir navnet sitt og velger et ønske å oppfylle (og hvilken dag det skal gis), og der Dolly senere (når ferien starter) kan «åpne» en ny dag i kalenderen for hver dag i ferien.

Pass på å bruke metoder du allerede har skrevet i andre klasser der det er naturlig. Gave-objekter skal kun refereres til og brukes i Juleferiekalenderklassen, og Onske-objekter skal kun refereres til og brukes i Onskeliste-klassen. Instansvariabler skal aldri aksesseres direkte fra utsiden av klassen de hører til.

4a) 10 poeng. Skriv klassen **Onske**. Klassen har instansvariabler beskrivelse, antall (om hun gjerne vil ha ønsket oppfylt flere ganger) og minimumspris (den billigste måten å oppfylle ønsket på). Initialverdiene oppgis i parametere til konstruktøren. Klassen har metodene (som du skal skrive):

- **passer** med parameter som angir maksimumspris. Denne metoden sjekker om et ønske er for dyrt for giveren (minimumsprisen til gaven er høyere enn maksimumskostnaden giveren kan akseptere) eller om ønsket er «brukt opp» (dvs antall = 0)
- **valgt** som kalles når en giver har valgt ønsket. Denne oppdaterer antall og returnerer beskrivelsen
- **__str__** som returnerer beskrivelse og minimumspris

4b) 10 poeng. Skriv klassen **Onskeliste** med metoder (som du skal skrive):

- **nytt_onske** med parametere etter behov, oppretter et nytt onske og legger det til onskelisten
- **hent_onsker** som tar maksimumspris (det meste en giver har råd til) som parameter og returnerer en liste med tekststrenger, der hver streng representerer et ønske i ønskelisten. For *valgbare* ønsker (som kan oppfylles innenfor giverens maksimumspris, og som ikke er «brukt opp») skal strengen være en representasjon av objektet med beskrivelse og pris, mens for ønsker som ikke oppfyller kravene skal det bare stå «Ikke valgbart onske».
- **onske_oppfylles** kalles når en giver har valgt hvilket ønske hen vil oppfylle. Metoden tar som parameter hvilket ønske som er valgt (et heltall), oppdaterer ønsket (at det er «brukt» en gang), og returnerer beskrivelsen av ønsket. Heltallet kan for eksempel være indeks til listen av ønsker.

4c) 5 poeng. Skriv klassen **Gave**. Konstruktøren tar parametere med initialverdier til to instansvariabler: beskrivelse og giver (hvem gaven er fra). Klassen har dessuten en metode som returnerer en streng med beskrivelse og giver (som du også skal skrive)

4d) 10 poeng. Skriv klassen **Juleferiekalender** med en konstruktør som tar antall dager i kalenderen som parameter og oppretter en ordbok der dagnummer er nøkkel og verdiene foreløpig er None. Dagnummer er et tall som angir datoen i desember eller januar: Fra 25 (siden kalenderen alltid starter 25. desember) til siste dag i ferien (juleferien varer max 31 dager dvs til og med dagnummer 24, men kan også være kortere enn dette). Tallet 28 representerer altså 28.desember, mens tallet 8 representerer 8.januar. Klassen har metodene (som du skal skrive):

- **ny_gave** som oppretter en ny Gave og legger den på en dag i juleferiekalenderen. Parametere er beskrivelse av gaven, giver (hvem gir gaven) og dagen den skal plasseres på (heltallet dagnummer). Du kan anta at den oppgitte dagen er gyldig og trenger ikke sjekke den (selv om dette skulle bety at du overskriver en eksisterende gave).
- **hent_dagens_gave** som returnerer en tekst som skal skrives ut når gaven på en bestemt dag (angitt av parameter dagnummer) åpnes. Teksten skal starte med datoen på formen dagnummer.månedsnavn (for eksempel 25. desember eller 1. januar). Hvis det ikke ligger noen gave i kalenderen for denne dagen skal teksten opplyse om det, ellers skal den inneholde beskrivelsen av gaven og hvem den er fra.
- **hent_ant_dager** returnerer antall dager i denne kalenderen

4e) 7 poeng. Skriv klassen **Julegavefikser** med en konstruktør som tar antall dager i kalenderen som parameter, og oppretter en Juleferiekalender og en Onskeliste. Konstruktøren skal også initialisere en instansvariabel neste_dag som holder rede på hvilken dag i julegavekalenderen som skal åpnes neste gang. Skriv også metoden:

- **les_onsker_fra_fil**. Metoden tar en parameter filnavn, og leser ett og ett ønske fra filen som har følgende format på hver linje:

  beskrivelse; antall; minimumspris

  For hver linje opprettes et ønske som legges til i ønskelisten.

4f) 8 poeng. Utvid klassen **Julegavefikser** med metodene (som du skal skrive):

- **velg_gave** leser inn maksimumspris fra en bruker som vil gi en gave, og skriver ut på terminalen en nummerert oversikt over alle ønskene tilpasset denne. Brukeren kan deretter velge et ønske å oppfylle ved å oppgi nummeret, dette registreres i ønskelisten som gitt én gang, og beskrivelsen tas vare på. Til slutt spørres brukeren om hvilken dag i kalenderen gaven skal «legges» i, og en ny gave som oppfyller det valgte ønsket legges der. Du trenger ikke sjekke om det allerede finnes en gave på angitt dag – det er greit at denne i så fall overskrives.
- **ny_dag** kalles når Dolly vil åpne neste dag i kalenderen. Metoden returnerer informasjon om gaven på denne dagen i kalenderen (om det er noen der) og oppdaterer instansvariabelen som holder rede på hvilken dag i kalenderen som skal åpnes neste gang. Husk at etter 31. desember (dagnummer 31) kommer 1. januar (dagnummer 1).

Question 25
Attached

Del 4: Julegavefikseren: En kombinert ønskeliste og julegavekalender

Totalt 50 poeng

Du kan anta i programmet du skriver at input (argumenter til metodene dine eller input fra fil/ bruker) er lovlige verdier av riktig type, du trenger altså ikke sjekke disse om det ikke nevnes spesielt.

Dolly Duck har tenkt ut en løsning på to problemer: Hun er lei av å få julegaver hun ikke ønsker seg og som bare står og støver ned – og hun har ingenting å gjøre i juleferien etter julaften. Løsningen hennes er en app som du skal få utvikle – der hun kan registrere ønsker om opplevelser og aktiviteter i en ønskeliste, og de som vil gi henne gaver kan velge et ønske fra ønskelisten som så legges i en «julegavekalender».

I julegavekalenderen kan hun åpne en ny gave hver dag, fra og med 25. desember og så lenge juleferien varer. «Gavene» som ligger i kalenderen er bare en tekst som beskriver opplevelsen eller aktiviteten, sammen med navnet på giveren – men da vet hun jo hvem hun kan kontakte. En vanlig julekalender kan åpnes hver dag fra 1. til 24. desember – men her skal vi altså lage en kalender som brukes *etter* julaften i stedet.

Julegavefikseren er bygget opp av 5 klasser som du skal skrive:

- En klasse Onskeliste som inneholder ønskelisten (en liste med Onske-objekter) og metoder for å legge inn ønsker, velge et ønske å oppfylle, og for å hente ut en liste med ønsker.
- En klasse Onske med metoder for å sjekke om ønsket passer for en giver eller ikke, for å velge et ønske og for returnere ønsket som en string.
- En klasse Julegavekalender med gaver for hver dag i juleferien, fra og med 25. desember og et valgfritt antall dager fremover (men aldri mer enn 21 dager). Klassen har metoder for å legge til en ny gave, hente gaven for en bestemt dag, og lese av hvor mange dager ferie julegavekalenderen er laget for.
- En klasse Gave som inneholder beskrivelsen av gaven og hvem den er fra. Denne informasjonen kan hentes ut som en streng.
- Den siste klassen er Julegavefikseren som tar antall dager i juleferien som parameter til konstruktøren. Klassen har metoder for å lese inn Dollys ønsker i ønskelisten fra fil, for å velge et ønske å oppfylle og legge som en gave i julegavekalenderen (for givere) og for å åpne en ny dag i kalenderen (for Dolly).

Du trenger ikke skrive noe hovedprogram – dette vil lese inn alle ønsker fra fil og ha et brukergrensesnitt der en som vil gi Dolly en gave oppgir navnet sitt og velger et ønske å oppfylle (og hvilken dag det skal gis), og der Dolly senere (når ferien starter) kan «åpne» en ny dag i kalenderen for hver dag i ferien.

Pass på å bruke metoder du allerede har skrevet i andre klasser der det er naturlig. Gave-objekter skal kun refereres til og brukes i Julegavekalenderklassen, og Onske-objekter skal kun refereres til og brukes i Onskeliste-klassen. Instansvariabler skal aldri aksesseres direkte fra utsiden av klassen de hører til.

4a) 10 poeng. Skriv klassen **Onske**. Klassen har instansvariabler beskrivelse, antall (om hun gjerne vil ha ønsket oppfylt flere ganger) og minimumspris (den billigste måten å oppfylle ønsket på). Initialverdiene oppgis i parametere til konstruktøren. Klassen har metodene (som du skal skrive):

- **passer** med parameter som angir maksimumspris. Denne metoden sjekker om et ønske er for dyrt for giveren (minimumsprisen til gaven er høyere enn maksimumskostnaden giveren kan akseptere) eller om ønsket er «brukt opp» (dvs antall = 0)
- **valgt** som kalles når en giver har valgt ønsket. Denne oppdaterer antall og returnerer beskrivelsen
- **__str__** som returnerer beskrivelse og minimumspris

4b) 10 poeng. Skriv klassen **Onskeliste** med metoder (som du skal skrive):

- **nytt_onske** med parametere etter behov, oppretter et nytt onske og legger det til onskelisten
- **hent_onsker** som tar maksimumspris (det meste en giver har råd til) som parameter og returnerer en liste med tekststrenger, der hver streng representerer et ønske i ønskelisten. For *valgbare* ønsker (som kan oppfylles innenfor giverens maksimumspris, og som ikke er «brukt opp») skal strengen være en representasjon av objektet med beskrivelse og pris, mens for ønsker som ikke oppfyller kravene skal det bare stå «Ikke valgbart onske».
- **onske_oppfylles** kalles når en giver har valgt hvilket ønske hen vil oppfylle. Metoden tar som parameter hvilket ønske som er valgt (et heltall), oppdaterer ønsket (at det er «brukt» en gang), og returnerer beskrivelsen av ønsket. Heltallet kan for eksempel være indeks til listen av ønsker.

4c) 5 poeng. Skriv klassen **Gave**. Konstruktøren tar parametere med initialverdier til to instansvariabler: beskrivelse og giver (hvem gaven er fra). Klassen har dessuten en metode som returnerer en streng med beskrivelse og giver (som du også skal skrive)

4d) 10 poeng. Skriv klassen **Julegavekalender** med en konstruktør som tar antall dager i kalenderen som parameter og oppretter en ordbok der dagnummer er nøkkel og verdiene foreløpig er None. Dagnummer er et tall som angir datoen i desember eller januar: Fra 25 (siden kalenderen alltid starter 25. desember) til siste dag i ferien (juleferien varer max 21 dager dvs til og med dagnummer 14, men kan også være kortere enn dette). Tallet 28 representerer altså 28.desember, mens tallet 8 representerer 8.januar. Klassen har metodene (som du skal skrive):

- **ny_gave** som oppretter en ny Gave og legger den på en dag i julegavekalenderen. Parametere er beskrivelse av gaven, giver (hvem gir gaven) og dagen den skal plasseres på (heltallet dagnummer). Du kan anta at den oppgitte dagen er gyldig og trenger ikke sjekke den (selv om dette skulle bety at du overskriver en eksisterende gave).
- **hent_dagens_gave** som returnerer en tekst som skal skrives ut når gaven på en bestemt dag (angitt av parameter dagnummer) åpnes. Teksten skal starte med datoen på formen dagnummer.månedsnavn (for eksempel 25. desember eller 1. januar). Hvis det ikke ligger noen gave i kalenderen for denne dagen skal teksten opplyse om det, ellers skal den inneholde beskrivelsen av gaven og hvem den er fra.
- **hent_ant_dager** returnerer antall dager i denne kalenderen

4e) 7 poeng. Skriv klassen **Julegavefikser** med en konstruktør som tar antall dager i kalenderen som parameter, og oppretter en Julegavekalender og en Onskeliste. Konstruktøren skal også initialisere en instansvariabel neste_dag som holder rede på hvilken dag i julegavekalenderen som skal åpnes neste gang. Skriv også metoden:

- **les_onsker_fra_fil**. Metoden tar en parameter filnavn, og leser ett og ett ønske fra filen som har følgende format på hver linje:

  beskrivelse; antall; minimumspris

  For hver linje opprettes et ønske som legges til i ønskelisten.

4f) 8 poeng. Utvid klassen **Julegavefikser** med metodene (som du skal skrive):

- **velg_gave** leser inn maksimumspris fra en bruker som vil gi en gave, og skriver ut på terminalen en nummerert oversikt over alle ønskene tilpasset denne. Brukeren kan deretter velge et ønske å oppfylle ved å oppgi nummeret, dette registreres i ønskelisten som gitt én gang, og beskrivelsen tas vare på. Til slutt spørres brukeren om hvilken dag i kalenderen gaven skal «legges» i, og en ny gave som oppfyller det valgte ønsket legges der. Du trenger ikke sjekke om det allerede finnes en gave på angitt dag – det er greit at denne i så fall overskrives.
- **ny_dag** kalles når Dolly vil åpne neste dag i kalenderen. Metoden returnerer informasjon om gaven på denne dagen i kalenderen (om det er noen der) og oppdaterer instansvariabelen som holder rede på hvilken dag i kalenderen som skal åpnes neste gang. Husk at etter 31. desember (dagnummer 31) kommer 1. januar (dagnummer 1).