



KANDIDAT

10108

PRØVE

TDAT1001 A Programmering grunnkurs

Emnekode	TDAT1001
Vurderingsform	Skriftlig eksamen
Starttid	20.12.2018 08:00
Sluttid	20.12.2018 12:00
Sensurfrist	21.01.2019 23:59
PDF opprettet	24.11.2019 16:10

Informasjon	
Oppgave	Oppgavetype
i	Dokument
Oppgave 1	
Oppgave	Oppgavetype
2.1	Langsvar
2.2	Programmering
2.3	Programmering
2.4	Programmering
Vedlegg	
Oppgave	Oppgavetype
i	Dokument
i	Dokument

2



Case-beskrivelse

Du skal lage en applikasjon for Soppkontrollen. Soppkontrollen har oversikt over alle sopparter som finnes i norsk fauna og er behjelpelig med å klassifisere sopp som turgåere og sankere har observert og/eller plukket og tatt med seg hjem.

Du skal i denne eksamensoppgaven jobbe med 3 ulike klasser:

- Soppart
- Soppregister
- Klientprogram

Soppart

En soppart har et navn, beskrivelse og informasjon om soppen er giftig eller ikke.

Klassen skal være immutabel.

Eksempel på hva som kan registreres om en soppart:

Navn: Rød fluesopp

Beskrivelse: Rød sopp med prikker. Du finner den i skog med bjørk og gran.

Giftig: Ja

Soppregister

Har en liste over alle sopparter. Når det opprettes et nytt soppregister skal det opprettes et tomt register med plass til 10 ulike sopparter. Dersom det registreres flere enn 10 sopparter (listen er full) skal den utvides med 10 nye plasser.

Eksempel på en liste med fire registrerte sopparter:

```
Registrerte Sopparter (Navn Beskrivelse Spiselig):
Rød fluesopp Rød sopp med prikker. Du finner den i skog med bjørk og gran. Giftig
Grønn fluesopp Grønn sopp. Næringsrik løvskog med eik, men også med bøk og hassel. Giftig
Kantarell Hele soppen er eggeplommegul, kjøttfull og traktformet med gaffelgrenete nedløpende
ribber. Trives best i moserik blåbærgranskog og i gammel bjørkeskog. Matsopp
Kongesjampinjong Hatten er først nesten kulerund med avflatet topp, senere mer hvelvet.
Næringsrik jord i hager og parker. Matsopp
```

Klientprogram

Et menystyrt program for å teste klasser og metoder og har også klassemetoder for å lese/ skrive hele soppregisteret til fil.

For å forenkle oppgaven noe så er soppartene enten spiselige eller giftige (matsopp/ giftig).

1 Oppgave 1 - Modellering

- Les casebeskrivelse og hele oppgavesettet og lag ett utvidet klassediagram for alle klassene.
- Vis i klassediagrammet hvordan du mener de ulike klassene bør samarbeide med

Med unntak av begrunnelsen i oppgave 1b, løses oppgave 1 på eget ark.

Skriv ditt svar her...

En soppart kan bare ha et sopprester, mens et sopprester har mange sopparter. Derfor blir det en en til mange kobling der. Jeg velger å bruke komposisjon når jeg legger til soppartene i registeret for at det ikke bare skal peke til en referanse.

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

3 4 6 4 5 2 1

Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
3464521	20.12.18	TDAT1001	10108	1a)	1

0000000

1111111

2222222

3333333

4444444

5555555

6666666

7777777

8888888

9999999

Soppart

-private String navn

-private String beskrivelse

-private boolean giftig

+public Soppart(String navn, String beskrivelse, boolean giftig)

+public String getNavn()

+public String getBeskrivelse()

+public boolean getGiftig()

+public boolean equals(Object o)

+public String info()

Soppregister

-private Soppregister(Soppart[] soppliste

-private int antReg

+public Soppregister(Soppart[] soppliste)

+public Soppregister()

-private utvid()

+public leggTilSopp(String navn, String beskrivelse, boolean giftig)

+public Soppart[] matSopp()

+public String info()

+public String sok()

Klient program

+public Soppregister lesRegFraFil(filnavn)

+public static void main(String[] args)

2 Oppgave 2 - Klassen Soppart

sett opp klassens objektvariabler

- a. lag to ulike konstruktører, en som tar inn verdier til alle klassens objektvariabler og en konstruktør som tar inn et objekt av typen Soppart.
- b. sett opp klassens tilgangsmetoder.
- c. Lag en metode som sjekker om en soppart er lik en annet. To sopparter er like dersom navnet er likt.
- d. Lag en metode som sjekker om en gitt søkestreng finnes i beskrivelsen av sopparten. Metoden skal returnere true eller false avhengig av om søkestrengen finnes som en del av soppartens beskrivelse.

```
1 class Soppart {
2     private String navn;
3     private String beskrivelse;
4     private boolean giftig;
5
6     //konstruktør med navn, beskrivelse og giftig
7     public Soppart(String navn, String beskrivelse, boolean giftig) {
8         this.navn = navn;
9         this.beskrivelse = beskrivelse;
10        this.giftig = giftig;
11    }
12
13    //konstruktør med objekt av typen Soppart
14    public Soppart(Soppart sopp) {
15        this.navn = sopp.getNavn();
16        this.beskrivelse = sopp.getBeskrivelse();
17        this.giftig = sopp.getGiftig();
18    }
19
20    //get metoder
21    public String getNavn(){
22        return navn;
23    }
24
25    public String getBeskrivelse() {
26        return beskrivelse;
27    }
28
29    public boolean getGiftig() {
30        return giftig;
31    }
32
33    //equals
34    public boolean equals(Object o) {
35        //sjekker om det ikke er et Soppart objekt
36        if(!(o instanceof Soppart) return false;
37
38        //sjekker om referansen er lik
39        if(this == o) return true;
40
41        //caster
42        Soppart obj = (Soppart) o;
43
44        //bruker equals for å sjekke om navnene er like siden de er av typen String
45        if(navn.equals(obj.getNavn())) return true;
46
47        //om det ikke blir returnert noe før nå, er de ikke like
48        return false;
49    }
50
51    public boolean sokString(String sokStreng) {
52        //bruker split til å dele opp alt i beskrivelsen i ord og legger det i en tabell
53        String[] ordBeskrivelse = beskrivelse.split(" ");
54
55        //sjekker sokeStreng opp mot alle ordene i tabellen
56        for(int i = 0; i < ordBeskrivelse; i++) {
57            if(ordBeskrivelse[i].equals(sokStreng)) {
58                return true;
59            }
60        }
61
62        //om den ikke ble funnet i tabellen
63        return false;
64    }
65
66    public String info() {
67        //oppretter en string med all informasjon utenom giftig
68        String utskrift = navn + ": " + beskrivelse + ". ";
69        //sjekker om det er matsopp eller om den er giftig
70        if(giftig) {
71            utskrift += "Giftig.";
72        }
73        else {
74            utskrift += "Matsopp.";
75        }
76
77        return utskrift;
78    }
79 }
```

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?

8 7 6 4 6 7 7

3 Oppgave 3 - Klassen Soppregister

- a. Sett opp klassens objektvariabler
- b. Sett opp klassens konstruktør
- c. Lag en metode som registrer en ny soppart. Ny soppart kan kun registreres dersom den ikke er registrert fra før. Dersom det ikke er plass i listen skal denne utvides med 10 nye plasser.
- d. Lag en metode som returnerer alle matsoppene.
- e. Lag en metode som returnerer en tekststreng med all informasjon om alle registrerte sopparter. Eks på utskrift: se nedenfor.
- f. Lag en metode som søker gjennom alle registrerte sopparter om en gitt søkestreng finnes i beskrivelsen av den enkelte soppart. Metoden skal returnere en teststreng med informasjon om alle sopparter som svarer til søket

Alle registrerte Sopparter (Navn Beskrivelse Spiselig):

1: Rød fluesopp Rød sopp med prikker. Du finner den i skog med bjørk og gran. Giftig

2: Grønn fluesopp Grønn sopp. Næringsrik løvskog med eik, men også med bøk og hassel. Giftig

3: Kantarell Hele soppen er eggeplommegul, kjøttfull og traktformet med gaffelgrenete nedløpende ribber. Trives best i moserik blåbærgranskog og i gammel bjørkeskog. Matsopp

4: Kongesjampinjong Hatten er først nesten kulerund med avflatet topp. Næringsrik jord i hager og parker. Matsopp

Skriv ditt svar her...

```
1 class Soppregister {
2     private Soppart[] arter;
3     private int antReg;
4
5     public Soppregister(Soppart[] soppliste) {
6         //bruker komposisjon
7         Sopparter[] kopi = new Sopparter[soppliste.length];
8
9         //oppretter nye objekter i riktig kopi plass av objektet fra soppliste plass
10        for(int i = 0; i < soppliste.length; i++) {
11            kopi[i] = new Soppart(soppliste[i].getNavn(), soppliste[i].getBeskrivelse(),
12                                soppliste[i].getGiftig());
13        }
14        arter = kopi;
15        antReg = kopi.length;
16    }
17
18    public Soppregister() {
19        this.arter = new Soppart[10];
20        antReg = 0;
21    }
22
23    private Soppart[] utvid() {
24        //lager en tabell som heter kopi med 10 plasser mer enn arter
25        Soppart[] kopi = new Soppart[arter.length + 10];
26
27        //bruker komposisjon og legger over objektene fra arter tabellen til kopi
28        for(int i = 0; i < arter.length; i++) {
29            kopi[i] = new Soppart(arter[i].getNavn(), arter[i].getBeskrivelse, arter[i].getGiftig
30                                ());
31        }
32        return kopi;
33    }
34
35    public boolean leggTilSopp(String navn, String beskrivelse, boolean giftig) {
36        //er mulig å gjøre dette på to måter; slik som her eller lage objektet i klientprogrammet
37        //også sende det som parameter, jeg valgte å bruke første.
38
39        Soppart nySopp = new Soppart(navn, beskrivelse, giftig);
40
41        //sjekker om det er plass, om det ikke er det utvider jeg
42        if(antReg == arter.length) {
43            arter = utvid();
44        }
45
46        //går gjennom alle objekter i arter og sjekker om de er like som det nye objektet, om det
47        //er noen like returnerer den false
48        for(int i = 0; i < antReg; i++) {
49            if(arter[i].equals(nySopp)){
```



```
48     } return false;
49 }
50
51 //legger til nySopp i arter
52 arter[antReg] = nySopp;
53 antReg++;
54 return true;
55 }
56
57
58 public Soppart[] matsopp() {
59     int teller = 0;
60     //sjekker hvor mange som er giftige for å vite hvor stor tabellen må være
61     for(int i = 0; i < arter.length; i++) {
62         //sjekker om de er ikke er giftige
63         if(!(arter[i].getGiftig())){
64             teller++;
65         }
66     }
67
68     //sjekker om det var noen giftige
69     if(teller == 0) return null;
70
71     //lager en tabell for å legge til matsoppene
72     Soppart[] matsopp = new Soppart[teller];
73
74     int nyTeller= 0;
75     //legger til matsoppene
76     for(int i = 0; i < arter.length; i++) {
77         //sjekker om de er giftige igjen
78         if(!(arter[i].getGiftig())){
79             //bruker aggerigering
80             matsopp[nyTeller] = arter[i];
81             nyTeller++;
82         }
83     }
84     return matsopp;
85 }
86
87 public String info() {
88     //opprettet en utskrift der alt legges til som kan returneres på slutten
89     //velger å ikke ha med Alle registrerte Sopparter (Navn Beskrivelse Spiselig): med i
90     //metoden, da dette enkelt kan skrives i klientprogrammet og metoden blir med anvendlig.
91     String utskrift = "";
92
93     //bruker en løkke til å lese inn informajson om hver sopptype, bruker info metoden jeg
94     //lagde i Soppart
95     for(int i = 0; i < arter.length; i++) {
96         utskrift += arter[i].info();
97     }
98
99     //returnerer utskriften
100    return utskrift;
101 }
102
103 public String sok(String sokStreng) {
104     //lager en teller, slik at bruker kan få beskjed om hvor mange treff det var
105     int teller = 0;
106
107     //lager en utskrift der beskrivelser legges til om sokStreng er der
108     String utskrift = "";
109
110     //går gjennom arter og sjekker om hvert objekt har sokeStreng i beskrivelsen, bruker
111     //sokStreng som jeg lagde i Soppart
112     for(int i = 0; i < arter.length; i++) {
113         if(arter[i].sokStreng(sokStreng)) {
114             utskrift += arter[i].getBeskrivelse() + ". \n";
115             teller++;
116         }
117     }
118
119     //returnerer antall treff og utskrift, på denne måten vil bruker få beskjed om at den har
120     //fått 0 treff også.
121     return "Antall treff: " + teller + "\n" + utskrift;
122 }
123 }
```

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?

3 1 3 6 2 6 1

Bruk følgende kode:

4 Oppgave 4 - Klientprogram

- a. lag en klassemetode som leser et objekt av typen Soppregister fra ei fil og

returner dette.

- b. I vedlegg 2 finner du rammeverket til et menystyrt program som gir bruker følgende valgmuligheter: List alle registrerte sopparter, List alle matsopper, Søk, Legg til ny Soppart og Avslutt. Idenne oppgaven skal du legge inn kode for to (2) av menyvalgene:
1. Legg inn ny soppart
 2. Søk
- c. Lag et UML aktivitets diagram for menyvalget «List alle matsopper».

Oppgave 4c) løses på utdelt ark.

Skriv ditt svar her...

```
1 import static javax.swing.JOptionPane.*;
2 import java.io.*;
3 import java.util.*;
4
5 public Soppreregister lesRegFraFil(filnavn) {
6     try{
7         //åpner strømmen til filnavn
8         FileInputStream fis = new FileInputStream(filnavn);
9         ObjectInputStream ois = new ObjectInputStream(fis);
10
11         //caster objektet til Soppreregister og leser fra filen
12         Soppreregister reg = (Soppreregister) ois.readObject();
13         //lukker strømmen
14         ois.close();
15         return reg;
16     }
17     //catcher relevante exception
18     catch (IOException e) {
19         e.printStackTrace();
20     }
21     catch (ClassNotFoundException e) {
22         e.printStackTrace();
23     }
24     catch (NullPointerException {
25         e.printStackTrace();
26     }
27     //om det er noen om ikke ble catchet lager jeg denne for sikkerhets skyld
28     catch (Exception e) {
29         e.printStackTrace();
30     }
31     //om det ikke har blitt returnert noe før nå så returnerer jeg null
32     return null;
33 }
34
35 public static void main(String[] args){
36     String filnavn = "soppreregister.ser";
37     Soppreregister register = lesRegFraFil(filnavn);
38     if (register == null){
39         //velger å bruke en tom konstruktør for å indikere at den skal opprette en tom tabell med
40         //ti plasser og at det er 0 registrerte sopper.
41         register = new Soppreregister(); //opprettNyttRegister(); // metode som oppretter
42         //ett tomt register
43
44         String[] muligheter = {"List alle", "List matsopper", "Legg til ny", "Søk", "Avslutt"};
45         final int LIST_ALLE = 0;
46         final int LIST_MATSOPPER = 1;
47         final int REG_SOPP = 2;
48         final int SOK = 3;
49         final int AVSLUTT = 4;
50         int valg = showOptionDialog(null, "Velg", "Eksamen des 2018", YES_NO_OPTION,
51             INFORMATION_MESSAGE, null, muligheter, muligheter[0]);
52         while (valg != AVSLUTT){
53             switch (valg){
54                 case LIST_ALLE:
55                     /*Anta at koden eksisterer*/
56                     break;
57
58                 case LIST_MATSOPPER:
59                     /*Anta at koden eksisterer*/
60                     break;
61
62                 case REG_SOPP:
63                     // Oppgave 4c)
64                     //leser inn navn
65                     String nySoppNavn = showInputDialog("Navn til ny sopp:");
66
67                     //leser inn beskrivelse
68                     String nySoppBesk = showInputDialog("Beskrivelse til ny sopp:");
69
70                     //bruker skriver 1 for giftig og 2 for matsopp
71                     String nySoppGiftigLest = showInputDialog("1: Giftig \n 2: Matsopp");
```

```
69      int nySoppGiftig(Integer.parseInt(nySoppGiftigLest);
70
71      //oppretter en boolean giftig
72      boolean giftig = true;
73      if(nySoppGiftig == 1) giftig = true;
74      else if(nySoppGiftig == 2) giftig = false;
75
76      //lager lagtTil for å ta vare på booleanverdien som retunerers
77      boolean lagtTil = register.leggTilSopp(nySoppNavn, nySoppBesk, giftig);
78
79      //gir bruker beskjed om soppen ble lagt til eller ikke
80      if(lagtTil) System.out.println("Soppen er lagt til");
81      else System.out.println("Soppen ble ikke lagt til");
82      break;
83
84      case SOK:
85          // Oppgave 4c) fyll inn kode her
86          //leser inn det programet skal søke etter
87          String sokeOrd = showInputDialog("Søk etter:");
88
89          //bruker metoden sok og lagrer resultatet i en String resultat
90          String resultat = register.sok(sokeOrd);
91
92          //skriver ut resultatet
93          System.out.println(resultat);
94          break;
95      default: break;
96  }
97  valg = showOptionDialog(null, "Velg", "Eksamen des 2018", YES_NO_OPTION, INFORMATION_MESSAGE,
98      null, muligheter, muligheter[0]);
99  skrivRegTilfil(filnavn,register);
100 }
```

Maks poeng: 10

Knytte håndtegninger til denne oppgaven?
Bruk følgende kode:

1056519

Question Code
Oppgavekode

Date
Dato

Subject code
Emnekode

Candidate ID
KandidatID

Question nr
Oppgavenr

Page number
Sidetall

1056519

20.12.18

TDAT1001

10108

4c)

2

Det er to måter å skrive ut alle matsoppene på. Man kan bruke metoden `matsopp()` i `soppregister` også lage en løkke der alle soppene blir skrevet ut ved å bruke `getMetoden` i klassen `soppart`. Eller man kan bruke metoden `matsopp()` i klassen `soppregister` også ta resultatet og lage et nytt `soppregister` også bruke `info` metoden til å få all informasjon om matsoppene også skrive det ut. Jeg velger den siste.

```
graph TD; A[Oppretter tabell med resultat fra matsopp()] --> B[betingelse tabell != null]; B -- sant --> C[Oppretter et nytt objekt av soppregister der tabellen legges inn]; B -- ikke sant --> D[printer ut "Det er ingen matsopper"]; C --> E[Skriver ut resultat ved å bruke info() i soppregister]; D --> F(( )); E --> F; F --> G(( ))
```