Sensorveiledning – TDAT1001 – 20. des 2018

Oppgave 1

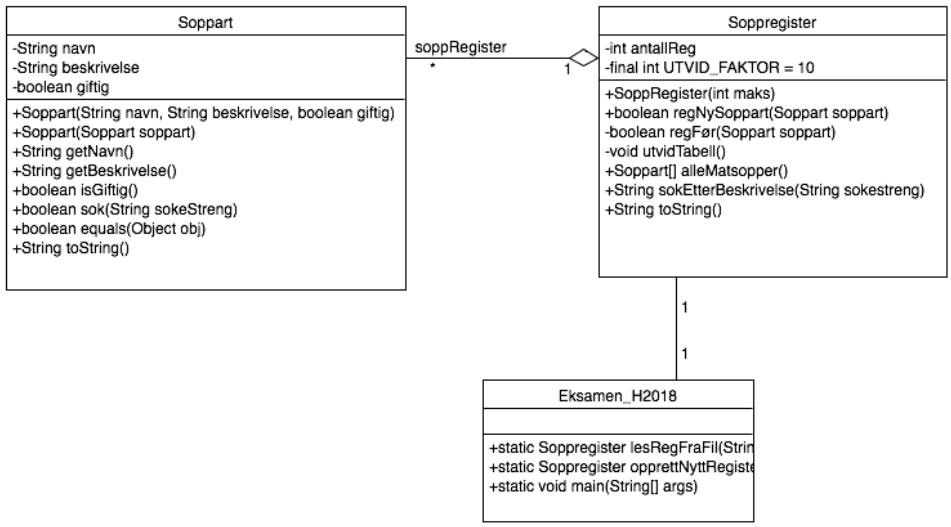
a)

syntaks feil - vil gi trekk
objektvariabler private, datatyper
metoder - argumenter, returverdier, tilgang private/ public
en-del-av-forhold aggregering/ komposisjon
assosiasjon

b) begrunnelse for forholdet mellom soppart og soppregister.
skal soppart eksistere uavhengig av soppregisteret. Hvorfor/ hvorfor ikke?
A-besvarelsen diskuterer også at Soppart er immutabel og hvilken betydning det har for valg av komposisjon/ aggregering.

Eksamen TDAT1001 20. des 2018,
Løsningsskisse oppgave 1a)

Kandidaten må ha med riktig detaljeringsnivå
Samsvar mellom det som er programmert i oppgaven - mtp metoder, attributter, en-del-av-forhold



Oppgave 2

a) to konstruktører - må tenke over komposisjon/ aggregering i kopi-konstruktøren - avhengig av hva som er beskrevet valgt i oppgave 1.
b) Trekk for set-metoder..
c) må ha riktig metode-hode, sjekk av this, instanceof, likhetssjekk må være på plass for å få full uttelling
d) riktig metode-hode og signatur og bruk av indexof eller annen string-metode. Trekk for unødvendig kompliserte løsninger.

Oppgave 3

a) for full uttelling må det være tabell av objekter og maksstørrelse og antallreg-teller, + for bruk av final MAKS_STR=10
b) konstruktør skal opprette ett tomt register med ti tomme plasser -
c) for full uttelling må det generaliseres ved å lage hjelpemetoder her -

private utvid-metode - der tabellen utvides med 10 plasser
private metode for å sjekke om sopp er registrert fra før - equals-metode fra oppgave 2 skal brukes
skal sjekke om plass (antallreg%10 ==0 el antallreg== register.length), sopp reg fra før, sopp != null..

d) full score: alle motsopper returneres i en tabell i hht til oppgave 1 (komp vs aggr..). halv-uttelling dersom en String returneres.
e) Her forventes toString() - må da lage toString også i klassen Soppart for å få full uttelling. Trekk for feil metode-signatur.
f) Må bruke metode fra oppgave 2 her og toString-metode fra forrige deloppgave for å få full uttelling

Oppgave 4
a)
A-kandidaten bruker try with resources/flere catch-finally og evt. logging av feil.
B-kandidaten har med unntakshåndtering og klassemetode (static)
C-kandidaten riktig metode signatur og retur-verdi, lukking av ressurser

```
public static Soppregister lesRegFraFil(String filnavn){
try (FileInputStream innstrom = new FileInputStream(filnavn);
        ObjectInputStream inn = new ObjectInputStream(innstrom)) {
        Soppregister register = (Soppregister)inn.readObject();
        inn.close();  // unødvendig ved try with resources
        return register;
}catch(FileNotFoundException e){
        System.out.println("Fil ikke funnet! (lesRegFraFil())");
}catch(EOFException e){
        System.out.println("Fil funnet, men tom! (lesRegFraFil())");
}catch(IOException ioe){
        System.out.println("IO-feil (lesRegFraFil())")
}catch (Exception e){
        System.out.println("Noe som ikke har med IO er feil. (lesRegFraFil())");
}
return null; // kommer kun hit naar noe har feilet
}
```

b)
case REG_SOPP:

```
String navn = showInputDialog("Navn: ");
String beskrivelse = showInputDialog("Beskrivelse: ");
boolean giftig = false;
int giftigLest = showConfirmDialog(null, "Giftig?", "Soppkontrollen", YES_NO_OPTION);
if (giftigLest == YES_OPTION) giftig= true;

Soppart s = new Soppart(navn, beskrivelse, giftig);
boolean ok = register.regNySoppart(s);
```

```
if (ok) showMessageDialog(null, "Sopp registrert");
else showMessageDialog (null, "Noe gikk galt");
break;
```

case SOK:

```
        String sokeStreng = showInputDialog("Hva vil du søke etter? ");
        String utskrift = register.sokEtterBeskrivelse(sokeStreng);
        showMessageDialog(null, utskrift);
        break;
```

4c) aktivitetsdiagram

- Her forventes ikke kode - kun diagram. Flere har misforstått denne oppgaven og laget diagram for ulike deler av koden Det som var forventet var diagram for en del av koden som ikke var implementert (LIST_MATSOPP) – en skulle da tenke seg til hvordan denne ville se ut uten å lage koden.
  Fordi at så mange missforsto oppgaven har vi valgt å gi uttelling for de som har fått vist aktivitetsdiagram med sekvens, valg og løkke-struktur uavhengig hvilken del av koden de har laget diagram for.
-

```
case LIST_MATSOPPER:
        Soppart[] matsopper = register.alleMatsopper();
        if (matsopper != null){
                String res = "Alle registrerte matsopper:\n";
                for (int i=0; i<matsopper.length; i++){
                        res += matsopper[i] + "\n";
                }
                showMessageDialog(null,res);
        } else showMessageDialog(null, "Ingen matsopper funnet");
        break;
```

Eksamen TDAT1001 20. des 2018,
Løsningsskisse oppgave 4c)

Kandidaten må ha med riktig detaljeringsnivå, bruke uml,
Løkke og valgstruktur må gå klart fram. Trenger ikke bruke kode - kan bruke tekst.
Samsvar med koden i oppgave 3 - fornuftig bruk av metoden fra oppgave 3d)