Oppgavetekst med sensorveiledning
Ny og utsatt prøve IN1010, august 2022

Nettdugnad for innsamling av støymålinger

En av dine medstudenter, Anne, er på besøk i hjembyen sin. Hun er vant til at det er en rolig by når hun er hjemme på besøk, men i den siste tiden har hun merket at det er mye støy og bråk i byen. Også vennene hennes, som bor permanent i byen, synes dette. De har henvendt seg til kommunen og sagt at støynivået, spesielt fra all gravingen og anleggsarbeidet, antagelig er over det som er tillatt, men kommunen mener at dette er helt i orden. For å undersøke saken nærmere har Anne bedt deg om hjelp. Hun har mange venner, arbeidskolleger og familie i byen, og du foreslår at alle kan installere en app på telefonen sin som kan måle støy når de går rundt i byen. Ved å samle disse målingene vil dere få håndfaste data som kan vises til kommunen. Slik *nettdugnad* (engelsk: «crowdsourcing») vet hun blir gjort til mange andre formål også.

I denne oppgaven skal du skrive deler av et program som skal ta imot måledata som appene produserer. Svar bare på det oppgavene ber om. Det du lager, blir ikke noe fullstendig program. Lever alle klassene som svar på Inspera-oppgave 1 og alle tegningene som svar på Inspera-oppgave 2. Skriv kommentarer i koden din som gjør det klart hvilke oppgaver du svarer på.

Noen av oppgavene bli kanskje klarere om du leser hele oppgaveteksten først. Om noe er uklart i oppgaven, kan du gjøre dine egne fornuftige antagelser, men skriv dem ned i besvarelsen din.

Denne innledningen til oppgavesettet er ment som en motivasjon og hjelp til forståelse, men du kan også løse alle oppgavene ved bare å lese beskrivelsen av hva de enkelte klassene og metodene skal inneholde og gjøre.

Når appen måler støynivået, er dette knyttet til et sted i byen. Du skal derfor skrive klassene Sted og Måling. Klassen Sted skal ta vare på målinger på to forskjellige måter: Ved hjelp av klassen MestStøyANT og i en enkelkjedet liste. Vi antar i dette oppgavesettet at det er OK å skrive klassenavn med æ, ø og å.

Når det utføres en støymåling, gjøres dette på et bestemt tidspunkt. I dette programmet skal all tid måles i hele minutter, med minutt 0 i det programmet starter. Vi tar altså ikke med hverken sekunder, timer eller dager i denne oppgaven, bare minutter.

Målinger
Et måledata-objekt (en måling) skal i tillegg til tidspunktet inneholde støynivået (i desibel (dB), et kommatall, altså en double). I klassen Måling skal disse to verdiene bli satt av konstruktøren.

I senere oppgaver skal målinger kjedes sammen i enkelkjedede lister, så klassen Måling må også inneholde en referanse som kan brukes til dette. Senere skal vi slå sammen noen målinger, så klassen må inneholde et antall som skal være 1 i det et objekt opprettes.

Oppgave 1. Vekt 2
Skriv klassen Måling.

Læringsmål oppgave 1. I denne oppgaven skal du vise at du kan programmere en meget enkel klasse med en enkel konstruktør. Det er viktig at det er deklarert en neste-peker av samme type som klassen selv.

Mest støy
Du skal nå lage en hjelpeklasse kalt MestStøyANT som skal holde orden på de mest støyende målingene, dvs de med høyest støynivåverdi. En konstant, ANT, i klassen skal angi det antallet målinger et objekt av klassen MestStøyANT skal holde orden på. Konstruktøren til kassen skal ha denne verdien som parameter. Programmet ditt skal bruke en array av lengde ANT til å lagre disse målingene. Klassen skal inneholde to public-metoder: void settInnANT(Måling m) og Måling[] hentUtANT(). I det et objekt av klassen blir opprettet, inneholder det ingen målinger, og de ANT første målingene som blir satt inn, blir alle tatt vare på. Pass på at du programmerer settInnANT() riktig også for disse første målingene som settes inn. Når flere enn ANT målinger settes inn, er det bare målingene med de høyeste støyverdiene som skal ligge lagret i objektet. Metoden hentUtANT() returnerer en array med disse største målingene. Skriv ned en invariant i klassen MestStøyANT som beskriver hvordan data blir lagret i arrayen i klassen.

Oppgave 2. Vekt 13
Skriv klassen MestStøyANT med datastruktur, konstruktør, metoden void settInnANT(Måling m) og metoden Måling[] hentUtANT().

Læringsmål oppgave 2. I denne oppgaven skal du vise at du kan programmere en beholder der data er lagret i en array og skrive en invariant som beskriver hvordan data er lagret i denne arrayen. Invarianten må si hvor data (målingene) er lagret hvis arrayen ikke er full og hvis målingene er sortert i arrayen må invarianten fortelle det.
Konstruktøren må gi verdi til konstanten ANT og opprette arrayen.
De to metoden settInnANT() og hentUtANT må programmeres etter spesifikasjonene, og når arrayen ikke er full er det fint om sistnevnte metode returnerer en array med færre enn ANT elementer.

Steder
Når en måling er gjort av appen, sendes data til ditt program som skal sette målingen (et objekt av klassen Måling) inn i programmets datastruktur. Stedet i byen der målingen gjøres, beskrives av klassen Sted. Hvert sted skal ta vare på målingene på to forskjellige måter: Et objekt av klassen Sted skal både inneholder en referanse til et objekt av klassen MestStøyANT og det skal inneholde en første-peker og en siste-peker som angir en enkelkjedet liste av målinger.

Et sted har et unikt navn (en String) som er parameter til klassen.

Den enkelkjedete listen over støymålinger på stedet skal være sortert på minutt, og første-pekeren peker på den målingen som ble lagt inn først, dvs. den eldste målingen, mens siste-pekeren peker på målingen som ble lagt inn sist, dvs. den nyeste målingen. En ny måling vil nesten alltid komme med en lik eller større minutt-verdi enn den siste som ligger i listen. Du skal skrive metoden void settInnMåling(Måling inn) i klassen Sted som setter målingen som er parameter, inn sist i den enkeltkjedede listen. Hvis målingen som skal legges inn i listen, har en minutt-tid som er mindre enn målingen som allerede ligger sist i listen skal den nye målingen ikke settes inn. Listen av målinger vil da alltid være sortert med nyeste måling sist.

Klassen Sted skal også bruke klassen MestStøyANT fra forrige oppgave til å ta vare på de ANT mest støyende målingene som er gjort på dette stedet. ANT er en konstant i klassen Sted og skal være en parameter til konstruktøren. Metoden settInnMåling() skal også sørge for at datastrukturen med disse ANT høyeste målingene blir oppdatert. Klassen Sted skal også inneholde en metode Måling[] hentUtANT() som returnerer verdiene i stedets MestStøyANT-objekt.

Ta med i klasen Sted en instansvariabel som til enhver tid angir den høyeste støyverdien i den lenkede listen av målinger og en public-metode som returnerer denne verdien.

Oppgave 3. Vekt 8
Basert på klassene i oppgavene 1 og 2 og det som er beskrevet over om klassen Sted, tegn opp en datastruktur med et sted der det er lagt inn 4 målinger og der ANT er 3. Angi klassenavnene til alle objektene og tegn alle instansvariabler med passende verdier. Du behøver ikke skrive hvilke typer instansvariablene har. I Måling-objektene behøver du heller ikke skrive navn på variablene hvis du lar støyverdiene være under 150 og minutt-verdiene over 200. Ikke tegn noen metoder.

Læringsmål oppgave 3. I denne oppgaven skal du vise at du forstår og kan tegne en datastruktur med både en array og en lenket liste. Det er viktig at tegningen inneholder ett objekt av hver av de to klassene Sted *og* MestStøyANT, *samt fire objekter av klassen* Måling. *Disse fire objektene må være lenket sammen i en enkelkjedet liste, og tre av dem må pekes på av arrayen med lengde 3.* MestStøyANT-*objektet må inneholde en referanse til denne arrayen, mens* Sted-

objektet må inneholde to referanser til hhv. objektet med lavest og høyest minutt verdi.

Oppgave 4. Vekt 9
Skriv klassen Sted med datastruktur, konstruktør, metoden som returnerer den høyeste støyverdien, metoden void settInnMåling(Måling inn) og metoden Måling[] hentUtANT().

Læringsmål oppgave 4. I denne oppgaven skal du vise at du kan programmere en litt komplisert klasse. Konstruktøren må opprette et MestStøyANT*-objektet og sette konstanten* ANT. *Klassen må deklarere første- og siste-pekere av typen* Måling. *Metoden* settInnMåling() *må sette objektet som parameteren refererer både inn i* MestStøyANT*-objektet og i den enkeltkjedede listen på riktig måte.*

For å forberede en datastruktur som kan gi bedre oversikt over steder med mye støy skal programmet ditt kunne slå sammen støymålinger gjort på samme tidspunkt. Dette skal gjøres i en metode void slåSammen() i klassen Sted. Metoden skal gå gjennom den lenkede listen av målinger og slå sammen dataene i Måling-objekter med samme tid til ett objekt. Dette gjenværende objektet skal bruke instansvariablen antall til å vise hvor mange objekter som er slått sammen. Støynivået (dB-verdien) i det gjenværende Måling-objektet skal være gjennomsnittet av støyverdiene til de sammenslåtte objektene med samme tid. Gjennomsnittet finner du ved å summere støyverdiene og dele på antallet. Den resulterende lenkede listen vil fremdeles være sortert på tid, men alle objektene vil ha forskjellig tid.

Oppgave 5. Vekt 11
Skriv metoden void slåSammen() i klassen Sted.

Læringsmål oppgave 5. I denne oppgaven skal du vise at du kan programmere en litt komplisert algoritme. Løkka som går gjennom hele listen bør terminerer riktig når listen er slutt. Det er viktig at objekter med like minutt-verdier blir funnet. Om dette gjøres med en egen løkke (så lenge minuttverdien er den samme) eller som en del av den ytre løkka spiller ingen rolle så lenge resultatet blir riktig.
Av de objektet som har like minuttverdier skal det bare være ett igjen og dette må ha riktig dB-verdi og riktig antall. Dette betyr bl.a. at de objektene som ikke skal være med i lista blir fjernet.
Programmet bør passe på å ikke forfølge referanseverdier som er null.

Gater
Mange mener at det er langs noen gater det er mest støy. Programmet ditt skal derfor analysere støyen langs gatene gjennom byen.

Programmet skal inneholde en klasse Gate som skal ta vare på stedene langs denne gaten. Navnet på gaten (en String) er parameter til konstruktøren i klassen.

Mange steder er knyttet til flere gater (f.eks. er et gatekryss et sted som er knyttet til begge de to gatene som krysser). I programmet ditt må det derfor være mulig å knytte et sted til flere gater, og vi skal ikke sette noen begrensning på antallet gater et sted kan knyttes til og ikke hvor mange steder det kan være i en gate.

Du skal skrive en klasse C med en indre klasse StedNode som gjør at steder kan lenkes sammen i dobbeltkjedede lister. Klassen StedListe skal inneholde referanser til første og siste node i en liste av steder. Klassen Gate skal inneholde en StedListe som skal inneholde alle stedene i gaten. Når et Gate-objekt opprettes inneholder det ingen steder.

Oppgave 6. Vekt 6.
Tegn en datastruktur med én gate med tre steder. Angi klassenavnene til alle objektene og tegn alle instansvariabler med navn og verdier, men du behøver ikke angi typene til variablene. Ikke tegn noe data inne i Sted-objektene, dvs. la Sted-objektene være tomme. Ikke tegn noen metoder.

Læringsmål oppgave 6. I denne oppgaven skal du vise at du forstår og kan tegne en datastruktur med en dobbeltkjedet liste av node-objekter. Lister av Node-objekter er kjent stoff i IN1010, så dette må være riktig tegnet for denne oppgaven (se løsningsforslaget) for å gi uttelling. StedNode*-objektene må inneholde neste- og forrige-pekere og en referanse til et* Sted*-objekt. Det må være tegnet et* Gate*-objekt og et* StedListe*-objekt med riktige instansvariable.*

Oppgave 7. Vekt 7
Skriv klassen Gate og klassen StedListe med den indre klassen StedNode. I denne oppgaven skal du ikke skrive noen metoder, men konstruktører hvis du trenger det.

Læringsmål oppgave 7. I denne oppgaven skal du vise at du kan deklarere en klasse, StedListe*, med en datastruktur for en dobbeltkjedet liste av objekter av en indre klasse. Dette er kjent stoff i IN1010, så dette må være gjort riktig for akkurat denne oppgaven for å få uttelling. Konstruktøren til klassen* Gate *må opprette et objekt av klassen* StedListe.

Oppgave 8. Vekt 8
Skriv en iterator over alle Sted-objektene i en StedListe.

Læringsmål oppgave 8. I denne oppgaven skal du vise at du kan skrive en iterator. Også dette er kjent stoff i IN1010 og iteratoren her er ikke spesielt vanskelig, så dette må være riktig gjort for problemstillingen i denne oppgaven for

å få uttelling. Det er viktig at iteratoren går gjennom node-objektene men at next-metoden returnerer en referanse til et Sted*-objekt.*

I de neste oppgavene kan du anta at det er fullt av gater og steder i datastrukturen selv om du ikke har skrevet metoder for å legge inn slike. Anta at alle steder er laget med den samme verdien på ANT og at det er minst ANT støyverdier lagret alle steder.

Programmet skal gå gjennom en gate for å finne hvor mye støy det er i denne gaten. Dette skal du først gjøre ved å bruke iteratoren du laget i forrige oppgave (dette kan du gjøre selv om du ikke har løst den oppgaven), deretter ved hjelp av en rekursiv metode. Du skal først bruke iteratoren og skrive metoden Måling[] finnMestStøyANT() som finner støyverdiene på det stedet i gaten som har de ANT mest støyende målingene. Husk at hvert sted allerede inneholder sine ANT mest støyende målinger, slik at du behøver bare å summere disse målingene og så finne hvilket sted som har den høyeste summen (men metoden skal returnere hele arrayen).

Oppgave 9. Vekt 8
Skriv metoden Måling[] finnMestStøyANT() i klassen Gate.

Læringsmål oppgave 9. I denne oppgaven skal du vise at du kan programmere en litt komplisert metode som bruker en løkke med en egendefinert iterator. Innmaten i løkka må være riktig programmert både for første gjennomløp av løkka og for de senere gjennomløpene. Summen av støyverdiene i MestStøyANT*-objektet i hvert sted må beregnes, antagelig best ved å lage en egen metode i klassen* MestStøyANT.

Nå skal du finne det stedet i gaten som har den mest støyende målingen noen sinne. Dette skal programmet gjøre ved hjelp av en rekursiv metode. Du skal skrive den rekursive metoden maksListeR() i klassen StedNode. Metoden skal returnere denne høyeste støyverdien i Sted-objektet i denne noden eller i resten av Sted-ene i listen.

Oppgave 10. Vekt 8
Skriv den rekursive metoden double maksListeR() i klassen StedNode.

Læringsmål oppgave 10. I denne oppgaven (og de to neste oppgavene) skal du vise at du kan programmere og bruke en rekursiv metode. Den metoden som skal skrives i oppgave 10 er den egentlige rekursive metoden. Det er viktig at den har en rekursjonsbunn (når det er slutt på listen) som da returnerer stedets maksimale støyverdi. Det er også viktig at metoden kalles rekursivt i neste node i listen, og resultatet som returneres fra dette kallet sammenlignes med nodens sted sin egen maksimale støyverdi. Det er den høyeste verdien av disse to som returneres fra metoden.

Oppgave 11. Vekt 4
Skriv metoden double maksListeR() i klassen StedListe som bruker den rekursive metoden du skrev i forrige oppgave til å finne den høyeste støyverdien i listen.

Læringsmål oppgave 11. I denne oppgaven skal du vise at du kan skrive et program som starter en rekursjon. Det er viktig at denne metoden kaller den rekursive metoden fra oppgave 10 i det første objektet i den lenkede listen.

Oppgave 12. Vekt 3.
Skriv metoden double maksStøyR( ) i klassen Gate som kaller metoden du skrev i forrige oppgave til å finne den høyeste støyverdien i gata.

Læringsmål oppgave 12. Også i denne oppgaven skal du vise at du kan skrive et program som starter en rekursjon. Denne metoden må kall metoden fra forrige oppgave i gatens StedListe *-objekt.*

Byer
Du skal nå lage en klasse By som inneholder alle gatene og alle stedene i byen. Gate-ne skal lagres i en ArrayList, mens alle Sted-ene skal lagres i en HashMap. Nøkkelen i HashMap-en er stedets navn.

Oppgave 13. Vekt 1
Deklarer datastrukturen beskrevet over i klassen By.

Læringsmål oppgave 13. I denne oppgaven skal du vise at du kan skrive noen enkle deklarasjoner. ArrayList*-en og* HashMap*-en må deklareres slik det er beskrevet i løsningsforslaget.*

GUI
I tillegg til å legge inn data via appen, skal det være mulig å bruke et GUI-vindu til å registrere støy.

Du skal skrive en klasse GUI med en metode void lagStøyVindu() som kan brukes til å opprette et vindu der en bruker kan registrere støymålinger. Konstruktøren til GUI-klassen skal ta en referanse til By-objektet der målingene gjøres, som parameter.

Når metoden lagStøyVindu() kalles, skal det opprettes et vindu der brukeren kan taste inn navnet på et sted, et tidspunkt og en støyverdi (i dB).
Når brukeren har tastet inn disse dataene i tre forskjellige felt og trykket på «OK»-knappen, skal programmet lage et Måling-objekt og sette dette inn i datastrukturen i byen.

Vinduet skal kunne ta imot flere målinger. Brukeren kan derfor taste inn nye data og trykke på «OK»-knappen på nytt for å registrere en ny måling.

Oppgave 14. Vekt 12
Skriv klassen GUI med metoden void lagStøyVindu() slik som beskrevet over. Gjør eventuelt andre forandringer i programmet ditt slik at GUI-et virker som beskrevet over.

Hint: For å lage et GUI-felt der brukeren kan skrive, kan vi bruke klassen JTextField. Den fungerer som JButton men har i tillegg metoden String getText() for å få tak i det som brukeren har skrevet.

Læringsmål oppgave 14. I denne oppgaven skal du vise at du kan skrive et program som lager og bruker et grafisk brukergrensesnitt og en hendelseshåndterer. Det er viktig at programmet inneholder en deklarasjon av en klasse som implementere grensesnittet ActionListener *og at denne inneholder metoden* actionPerfomed() *som igjen leser av verdiene som er fylt inn i vinduet og sørger for at det blir laget et* Måling*-objekt med disse dataene og at dette objektet blir lagt inn i datastrukturen i byen. Handlingene som skal til for å lage et* Måling*-objekt og legge dette inn i byen kan antagelig best gjøres med en metode i klassen* By. *Det er videre viktig at metoden som skal skrives i denne oppgaven oppretter et vindu med felt for å fylle inn data og en «OK»-knapp som kobles opp mot en instans av klassen som implementerer grensesnittet* ActionListener. *Programmet bør bruke hintene i oppgaven, men ellers er det ikke viktig at oppsettingen av vinduet er helt riktig programmert eller at data hentes ut helt riktig.*