UTSATT OG NY EKSAMEN I IN1010, AUGUST 2021

Les hele oppgavesettet før du begynner å svare, det kan være nyttig informasjon om en deloppgave senere i teksten.

Du har søkt jobb i et reisebyrå som skal styrke satsingen sin med å selge reiser på nettet.
I intervjuprosessen må du vise at du er en god programmerer og du skal løse oppgavene nedenfor. Reisebyrået lager programmer i Java og bruker en kodestil som er den samme som den som brukes i IN1010.

Du får i oppdrag å lage et meget enkelt program der kundene selv kan søke i reiser. Dette er charterreiser der fly og hotellopphold vanligvis selges sammen.
Du skal skrive klassehierarkiet for reiser, en beholder som kan inneholde reiser, et GUI der kundene kan søke i reiser og et program med tråder som gjør at søking i mange beholdere med mange reiser kan gå raskere.
Om noe er uklart, kan du gjøre dine egne fornuftige antagelser og skrive dem ned i besvarelsen din.

Om innlevering:
Svarene dine på oppgavene 1a – 1j skal settes sammen til et fullstendig program med en hovedklasse og en main()-metode og leveres samlet i en zip-fil som svar på oppgave 1. Du kan selv velge om du vil ha klassene i hver sin fil eller om du vil samle alle klassene i én fil. Du skal ikke levere .class-filer. Tegninger kan være håndtegning fotografert med mobil eller generert av et tegneprogram. Filer med tegning skal ha navn etter oppgaven de svarer på, og være på .png, .jpg eller .pdf-format – for eksempel 1a.jpg.
På samme måte skal svarene på oppgavene 2a og 2b til sammen utgjøre et fullstendig program som leveres samlet i én zip-fil som svar på oppgave 2.
Merk: Det er ikke et krav at programmene du leverer er kjørbare. Du blir ikke trukket for ubetydelige feil, selv om de gjør at programmet ikke kompilerer eller virker helt som det skal.

I de fleste oppgaver blir du bedt om å programmere en metode eller en klasse. Det kan hende at den beste måten å gjøre dette på inkluderer å forandre eller legge til kode også i andre metoder eller klasser. Skriv i så fall en kommentar i slik kode som forklarer hvilken oppgave du svarer på.

Reisebyrå på nettet

Alt reisebyrået selger er reiser. Alle reiser har et navn, en pris og et antall ledige plasser. Du skal anta at alle reiser har forskjellige navn.

Mange reiser (som vi heretter vil kalle «vanlige reiser» eller bare «reiser») har bare de egenskapene nevnt over. Klassen som beskriver slike reiser skal du kalle Reise.
I tillegg er det to typer spesielle reiser:
Noen reiser selges som «all-inclusive-reiser», dvs at mat følger med i prisen på hotellet.
Noen reiser selges uten hotell, såkalte «bare-fly-reiser».

Prisen som lagres i reise-objekter er nettopris (pris uten merverdiavgift, mva). Når programmet viser frem prisen på en reise, må du legge til 25% mva. Et unntak er bare-fly-reiser der det ikke er mva (med andre ord 0% mva). Alle Reise-objekter skal ha en metode som gir den prisen som vises frem, og du skal her bruke polymorfi. Alle priser oppgis i hele kroner og det er ikke viktig hvordan du eventuelt fjerner øre-delen av en pris med mva.

Ut over dette har bare-fly-reiser ingen andre egenskaper enn vanlige reiser. All-inclusive-reiser derimot har i tillegg en kvalitetsvurdering, et tall mellom 1 og 6 som sier noe om kvaliteten på maten på hotellet.

Alle objekter skal få startverdier på alle sine instansvariabler gjennom parametere til konstruktørene.

Oppgave 1a vekt 1
Tegn opp klassehierarkiet for reiser.
Slutt oppgave 1

Oppgave 1b vekt 4
Programmer alle klassene i dette klassehierarkiet.
Slutt oppgave 1b

Det ligger 9 filer på eksamensområdet på IN1010s hjemmeside: reiser.txt, reiser0.txt, reiser1.txt, reiser2.txt, reiser3.txt, reiser4.txt, reiser5.txt, reiser6.txt og reiser7.txt. Den første skal du bruke i oppgave 1, mens de 8 andre skal brukes i oppgave 2.

Hver linje i disse filene inneholder data om en reise. Hver linje inneholder flere felt adskilt av komma. Husk at du kan splitte opp en slik linje med metoden split(",") i klassen String. Det er også en del overflødige blanke på hver linje som du kan bli kvitt med trim()-metoden i String.

Først på linjen står det unike navnet på reisen. Deretter kommer nettoprisen og så antallet plasser som selges av denne reisen. For vanlige reiser er det ikke mer på linjen. For all-inclusive-reiser står det INCL i neste felt og det siste feltet er kvalitetsvurderingen. For bare-fly-reiser står det BF og dette er det siste feltet. Du kan se dette i filene på IN1010-området.

Oppgave 1c vekt 6
Skriv en statisk metode les() som leser inn alle linjene i en slik fil og oppretter et objekt for hver linje. Når et objekt er laget, skal settInn()-metoden du lager i oppgave 1e kalles for å sette objektet inn i en beholder du lager der. Navnet på filen og en referanse til en slik beholder er parametere til metoden les(). Denne metoden skal leveres som en del av klassen Reisebyra, se lenger nede i oppgave 1 for mer informasjon om dette.
Slutt oppgave 1c

Beholder for reiser

Du skal skrive en beholder som kan lagre reiser. Reiser skal kunne lagres, finnes igjen og fjernes basert på en nøkkel som skal være en String.

Beholderen skal ha fire offentlige metoder: settInn(nokkel, reise), hent(nokkel), fjern(nokkel) og antall(). Parameteren nokkel er en referanse til et String-objekt, mens parameteren reise er en referanse til et objekt av klassen Reise eller en subklasse. Metoden settInn() skal ikke returnere noe, antall() skal returnere et heltall, mens de to metodene hent() og fjern() skal returnere en referanse av typen Reise.

Oppgave 1d vekt 2
Skriv et interface kalt Hash som definerer en beholder med de fire metodene over.
Slutt oppgave 1d

Du skal implementere interfacet Hash i en klasse HashListe, som også skal implementere Iterable<Reise>.

I dette oppgavesettet vil vi bruke det unike reisenavnet (en String) som nøkkel. Når du programmerer beholderen skal du ikke bruke klasser du må importere fra Java-biblioteket annet enn dem som kommer automatisk i java.lang samt java.util.Iterator. Du skal ikke lage beholderen generisk, den skal bare kunne ta vare på objekter av klassen Reise og dens subklasser. Du skal ikke lese eller endre noen instansvariabler i Reise, hverken direkte eller indirekte (dvs gjennom metoder) når du programmerer HashListe.

Implementasjonen skal videre gjøres slik:
Når metoden settInn(nokkel,reise) kalles i beholderen, skal både reisen og den tilsvarende nøkkelen lagres i et objekt av en indre klasse med navn Node. Beholderen skal bruke en array til å ta vare på Node-objekter. La navnet på denne arrayen være alleN, og la lengden være en parameter til konstruktøren til klassen HashListe.

I klassen Object finnes en metode hashCode() som beregner en såkalt hash-verdi av objektet. På norsk kalles det ofte en *foldet verdi*. Du kan slå opp hashCode() i Javas bibliotek. I IN1010 er vi vant til å bruke klassen HashMap<K,V>, der den første parameteren, K, er nøkkelen, som ofte er en String. Metoden put(nokkel,verdi) i HashMap kaller nokkel.hashCode() og finner den foldede verdien (hash-verdien) av objektet som nokkel peker på. I denne oppgaven skal du lage en klasse som ligner på klassen HashMap.

Når metoden settInn(nokkel,reise) kalles i beholderen, skal programmet som sagt kalle nokkel.hashCode(). Svaret er et positivt eller negativt heltall vi kan kalle x. Hvis tallet er negativt, fjern fortegnet slik at det blir positivt. Du skal beregne en verdi, kalt y, som er x modulo lengden av arrayen i beholderen (y = x % alleN.length;). På denne måten blir y en gyldig indeks i arrayen alleN. Deretter skal du sette alleN[y] til å peke på det Node-objektet som skal lagres.

Det er en sjanse for at to Node-objekter skal inn på samme plass i arrayen, dvs. at y er den samme for to forskjellige nøkler. Hvis arrayen er kort, er denne sjansen større enn om arrayen er lang. Node-objekter med samme y-verdi skal lenkes sammen i en liste der første Node i listen pekes på av alleN[y]. Du kan selv velge hvordan denne listen er organisert, men du kan gjerne anta at lengden av arrayen er stor i forhold til antall reiser i beholderen, slik at listen vanligvis er kort. Husk, du må selv programmere denne lenkede listen (i oppgave 1f), du kan ikke bruke noen klasser fra Java-biblioteket.
Beholderen skal ikke kunne inneholde to reiser som settes inn med samme nøkkel.

Velg selv (og dokumenter i koden din) hvordan programmet ditt håndterer dette.

Oppgave 1e vekt 4
Tegn opp datastrukturen inne i et HashListe-objekt med arrayen alleN og med 4 Node-objekter med instansvariabler. La 3 nøkler ha samme y-verdi. Du trenger bare tegne ett nøkkel-objekt og ett Reise-objekt, men i dette Reise-objektet skal du ha med alle instansvariabler. Velg en passende kort lengde (større enn 3) for alleN i denne tegningen. Bruk piler til å tegne referanser til objekter på tegningen. Du må skrive navn på, men du behøver ikke angi typen til instansvariablene du tegner. Ikke tegn metoder.
Slutt oppgave 1e

Oppgave 1f vekt 20
Skriv hele klassen HashListe (bare unntatt iteratoren som du skal du programmere i neste oppgave)
Slutt oppgave 1f

Oppgave 1g vekt 10
Skriv en iterator i klassen HashListe som kan brukes til å finne alle Reise-objektene i beholderen.
Slutt oppgave 1g

Grafisk brukergrensesnitt - GUI

I oppgave 1i skal du skrive et GUI som samhandler med beholderen med reiser. Her trenger du tre Swing-klasser som ikke er gjennomgått på forelesningene, men som er beskrevet i læreboken og selvfølgelig i Java-dokumentasjonen.

JTextField opprettes slik: JTextField t = new JTextField("initiell tekst");
Den fungerer omtrent som JLabel men lar brukeren redigere teksten. Programmet kan få tak i teksten med metoden *getText()*. (Hint: Denne klassen er velegnet om brukeren skal oppgi et tall.)

JTextArea opprettes slik: JTextArea a = new JTextArea(10,50);
(Parametrene er antall rader og kolonner det settes av plass til.)
Den brukes til å skrive ut flere linjer tekst på skjermen. De to mest aktuelle metodene er:
  * *setText(null)* vil blanke området.
  * *append("xxx" + "\n")* vil skrive ut tekst i området; husk linjeskift.

JScrollPane opprettes slik: JScrollPane s = new JScrollPane(a) der a er en referanse til et JTextArea. Den brukes til å gi tekstvinduet muligheter til å kunne rulles.

GUI-vinduet skal inneholde et område (f.eks. et JTextArea) der reiser kan vises frem. I tillegg skal vinduet inneholde knapper og felt som begrenser hvilke reiser som vises.
Brukeren skal oppgi en nedre grense og en øvre grense for hva reisene som vises skal koste (med mva). I tillegg skal brukeren kunne velge å bare få vist all-inclusive-reiser eller bare-fly-reiser. Du må sørge for at ikke «all inclusive» og «bare fly» kan velges samtidig.

Oppgave 1h vekt 6

Skriv en klasse Filter med en offentlig metode med signaturen Reise filtrer(Reise r) som returnerer null hvis reisen som er parameter ikke slippes gjennom filteret og returnerer reisen selv (r) om reisen slipper gjennom. Parametere til konstruktøren skal være nedre grense og øvre grense for prisintervallet der reiser skal slippe gjennom. Parametrene skal også gi informasjon om filteret skal slippe gjennom alle reiser, bare all-inclusive-reiser eller bare-fly-reiser.
Slutt oppgave 1h

Oppgave 1i vekt 17
Skriv klassen ReisebyraVindu som inneholder GUI-delen av programmet som beskrevet over. Parameteren til konstruktøren av ReisebyraVindu er en referanse til HashListen som inneholder alle reisene. Når brukeren trykker på en knapp «Finn» skal programmet lage et filter basert på feltene som er satt i GUI-et. Deretter skal det gå gjennom alle reisene i beholderen men bare vise frem i vinduet de reisene som passerer filteret.
Hint: Opprett vinduet og alle komponentene og koble komponentene sammen med hendelseshåndteringsobjektene i konstruktøren til ReisebyraVindu.
Slutt oppgave 1i

Oppgave 1j vekt 4
Skriv til slutt klassen Reisebyra med en main()-metode som leser inn data om alle reisene til en HashListe. Til slutt operettes et ReisebyraVindu-objekt som oppretter vinduet der brukeren kan oppgi sine preferanser og få listet ut ønskede reiser. Alt du har skrevet i oppgave 1 skal til sammen utgjøre et fullstendig program.
Slutt oppgave 1j

Levering av oppgave 1: Alle programfiler (eller bare én om du velger å ha alle klassene i samme fil) og filer med tegninger samles i en zip-fil med navn Oppgave1.zip

Oppgave 2: Leting med tråder

I denne oppgaven skal du lete i reiser ved hjelp av tråder. Svarene på oppgavene 2a og 2b leveres som et fullstendig program som svar på oppgave 2.

Det fullstendige programmet i oppgave 2 vil på mange måter være likt og vil gjenbruke mange av delene i programmet fra oppgavene 1. Hvordan du har gjort oppgave 1 vil imidlertid ikke virke inne på din karakter på oppgave 2. Du skal ikke forandre klassene for reiser, HashListen eller Filter-klassen. Du skal gjøre forandringer i Reisebyra-klassen og i ReisebyraVindu- klassen og du skal skrive noen nye klasser. Som svar på oppgave 2 kan du velge om du vil levere de klassene som ikke forandres på nytt (slik at det hele kan bli et virkende program) eller du kan velge å bare levere de endrede og de nye klassene. Legg inn tydelige kommentarer i de endrede klassene hvor du svarer på spørsmålene fra oppgave 2.

Reisebyrået regner med å utvide med mange flere reisetilbud, og vet at det er viktig for kundene å få søkeresultatene raskt fram på skjermen. Du må derfor vise at du kan programmere med tråder. Hun som er leder for programmererne og som intervjuer deg, sier at de nok kommer til å lage et produksjonsprogram der tråder ligger og venter på å få leteoppgaver. Nå vil hun imidlertid at du skal vise at du kan lage et enklere program der GUI-

et lager nye tråder for hver gang brukeren starter et søk. Nedenfor skal du behandle 8 filer og du skal lage 8 tråder, men la dette antallet være en konstant (som du nå setter til 8) slik at du eventuelt senere kan utvide med flere filer og flere tråder.

Oppgave 2a vekt 3
Skriv en metode lesAlle() som leser de 8 filene reiser0.txt, reiser1.txt, reiser2.txt, reiser3.txt, reiser4.txt, reiser5.txt, reiser6.txt og reiser7.txt og legger dataene i hver sin beholder av klassen HashListe. Lag metoden så generell at om du senere skal lese flere filer (som heter reiser8.txt, reiser9.txt, resier10.txt, osv), så klarer metoden din det. Selv om du ikke har programmert les()-metoden i oppgave 1, kan du bruke denne. Parameter til metoden lesAlle() er en referanse til en array av de tomme HashListe-beholdere. Lengden på denne arrayen gir hvor mange filer det er, og tallet i filnavnet er det samme som indeksen i arrayen. Du skal ikke bruke tråder i oppgave 2a.
Slutt oppgave 2a

Nå skal du bl.a. skrive en monitor og en trådklasse slik at du kan opprette de 8 trådene beskrevet over. Hver tråd skal få ett (og det samme) filteret som parameter til konstruktøren sin og deretter skal trådene samtidig lete gjennom hver sin beholder. Monitoren skal inneholde én HashListe-beholder som er tom i det monitoren genereres. Hver gang en tråd finner en reise som passerer filteret, skal den kalle en metode i monitoren som tar imot og lagrer denne reisen i monitorens HashListe. På denne måten vil til slutt denne HashListe-beholderen inneholde alle reisene fra alle trådene som passerer filteret. Når en tråd er ferdig med å lete i beholderen sin, skal den terminere.

Programmet ditt skal opprette en array med de 8 beholderne, og innlesing til beholderne skal gjøres én gang for alle i det programmet ditt starter opp.

Trådene og monitoren skal imidlertid opprettes og startes først når brukeren trykker på knappen «Finn» og ønsker å lete i alle beholderne. Deretter skal GUI-programmet vente på at alle trådene er ferdige med å lete før det går videre (selv om dette gir et litt dårlig GUI som ikke responderer hvis letingen tar lang tid). Når alle trådene er ferdige, henter GUI-programmet ut HashListe-beholderen med alle resultatene som har passert filteret, fra monitoren og viser reisene fram i vinduet på samme måte som i oppgave 1.

Oppgave 2b vekt 23
Skriv klassene for monitoren og trådene. Skriv om klassen ReisebyraVindu slik at GUI-et oppfører seg slik som beskrevet over. Gjør også de nødvendige forandringene i Reisebyra-klassen slik at det hele kan virke sammen som et fullstendig program.
Slutt oppgave 2b

Levering av oppgave 2: Alle programfilene (eller bare én om du velger å ha alle klassene i samme fil) samles i en zip-fil med navn Oppgave2.zip

Slutt hele oppgavesettet