# Hotell

*Sensurveiledning (med oppgaveteksten)*

*• Skrivefeil ignoreres vanligvis, men feil som kan tyde på manglende forståelse (for*

*eksempel kopiering av kode som ikke passer inn eller som ikke svarer på oppgaven) vil gi*

*trekk. Norske tegn er helt OK*

*• Gjennomgående litt større feil/ mangler vil det bli trukket for (selv om de er små hver for*

*seg)*

*• Eksempler på feil/unøyaktigheter som ignoreres:*

*• Mangler (noen få) ';' , '}' eller '{' om koden ellers gir mening*

*• Småfeil i (gjenkjennbare) navn*

*• Import-setninger mangler*

*• Det legges lite vekt på riktige aksessmodifikatorer (public og private)*

Du har søkt om jobb i dataavdelingen til en stor hotellkjede, og som en del av jobbintervjuet

blir du bedt om å lage noen programdeler. Hotellkjedens datasystemer blir programmert

etter gode objektorienterte prinsipper, slik som i kurset IN1010 på Universitetet i Oslo, og

du skal vise at du behersker slik programmering. Det du skal programmere utgjør ikke et

fullstendig program. Husk derfor å se nøye etter hva oppgavene ber om og programmer

bare dette. Svar utover det som det blir bedt om, vil ikke telle med i beregningen av

karakteren din. Hvis noe er uklart i oppgaven, så gjør, og skriv ned i besvarelsen din, dine

egne fornuftige forutsetninger.

Alle hotellrom er subklasser av klassen Rom men det skal ikke finnes objekter av klassen

Rom. Det er tre typer rom: enkle rom, vanlige rom og suiter.

I tillegg har noen vanlige rom og noen suiter kjøkken, og da må du vite antall kvadratmeter

på dette kjøkkenet (et heltall). Ingen enkle rom har kjøkken. Implementer det å ha kjøkken

som et interface.

**Oppgave 1.** 5 poeng

Tegn opp klassehierarkiet. Ta med interfacet.

*Læringsmål oppgave 1.*

*Her skal du vise at du vet hva et klassehierarki er og at du kan tegne opp et slikt. I tillegg skal*

*du vise at du kan tegne interface (f.eks. kalt Kjøkken) i et slikt hierarki. Rom skal helst være*

*merket som abstract. Det skal være tre nivåer, Rom øverst, de tre klassenen f.eks. EnkleRom,*

*VanligRom og Suite på nivå to, og nederst (på nivå tre) to klasser som også implementerer*

*interface-et Kjøkken.*

Alle rom har et romnummer, et antall kvadratmeter (inklusive kjøkkenet hvis rommet har

kjøkken), et antall sengeplasser og en etasjeangivelse. Alt dette er konstanter som er heltall

og som skal gis verdier av konstruktørene. I tillegg har alle rom en boolsk variabel som sier

om rommet er opptatt eller ledig og en referanse til et annet rom (slik at rom kan lenkes

sammen i en liste). Bortsett fra navnene på klassene skal enkle rom, vanlige rom og suiter ha

de samme egenskapene i programmet ditt.

Alle klassene skal ha en toString()-metode. Den skal gjøre det klart om det er et enkelt

rom, et vanlig rom eller en suite, og den skal ta med alle verdier til alle instansvariabler og

konstanter unntatt referansevariabelen til neste rom i listen. Ikke legg vekt på at resultatet

skal bli pent hvis det skrives ut.

**Oppgave 2.** 8 poeng

Programmer alle klassene og interfacet.

*Læringsmål oppgave 2.*

*Her skal du vise at du kan programmere klasser og interface. Det legges vekt på at alle*

*instansvariabler er riktig deklarert og at alle untatt nestepekeren og den boolske variabelen*

*opptatt er deklarert som konstanter (med final), at konstruktørene er riktig programmert*

*(med super()), at interfacet er riktig programmert med en metodesignatur og at klassene*

*som implementerer interface-et har en konstant som angir antall m2 og at metoden i*

*interfacet returnerer denne verdien.*

*I tillegg skal den polymorfe metoden toString() defineres riktig med riktig kall på den samme*

*metoden i superklassen (super.toString()).*

## Hotell med én etasje.

Du skal videre i oppgaven programmere deler av klassen Hotell, i første omgang med én

etasje og siden med flere. Klassen skal inneholde en datastruktur som skal holde orden på

alle rommene på hotellet. Hvert rom er beskrevet av et objekt av en subklasse til klassen

Rom.

I et hotell med én etasje vil instansvariablen etasje være 1 i alle rom-objektene. Videre vil

alle rommene i et slikt hotell være lenket sammen i en enkelkjedet liste der en referanse

med navn forsteRom peker på det første rommet i denne listen.

**Oppgave 3.** 4 poeng

Anta at det er fire rom på hotellet. Tegn opp datastrukturen med disse fire rommene som

alle skal være objekter av forskjellige klasser. Tegn opp alle instansvariabler med passende

innhold men ikke tegn metoder.

*Læringsmål oppgave 3.*

*Her skal du vise at du skjønner hvordan en enkel datastruktur med*

*en enkeltkjedet liste ser ut. Minst ett av objektene må ha egenskapen Kjøkken. Variabelen*

*forsteRom må peke på det første objektet i listen . Objektene i denne listen henger sammen*

*ved hjelp av neste-pekere i objektene selv og alle instansvariablene (og konstanter) er med.*

**Oppgave 4.** 1 poeng

I klassen Hotell vil konstanten MAX\_ANT\_SENGEPLASSER (denne kan du sette til 8 i

programmet ditt) angi det maksimale antallet sengeplasser det kan være i et rom. Deklarer

denne konstanten og referansen forsteRom i klassen Hotell.

*Læringsmål oppgave 4.*

*Her skal du vise at du kan skrive meget enkle deklarasjoner.*

*Referansen må være av typen Rom.*

## Gjester og reservasjoner

Anta at listen som forsteRom peker ut, er fylt med Rom-objekter som beskriver alle

rommene i hotellet. Du skal nå programmere deler av et meget enkelt system med

romreservasjoner og tildeling av rom til gjester.

Gjestene på hotellet er beskrevet av klassen Gjest. For enkelhets skyld skal den i ditt

program bare inneholde et navn (en String) og, når en gjest har fått et rom, en referanse

til dette rommet.

En romreservasjon er beskrevet av klassen Reservasjon. I denne oppgaven skal vi lage

svært enkle reservasjoner uten noen datoer eller lengde på oppholdet. Objekter av klassen

Reservasjon skal derfor bare inneholder en referanse til gjesten som har bestilt et rom,

det antallet sengeplasser gjesten trenger på rommet sitt, en boolsk variabel som er true om

gjesten ønsker et rom med kjøkken samt to pekere forrigeR (forrige reservasjon) og

nesteR siden reservasjoner skal lagres i en dobbeltlinket liste. Den første reservasjonen

pekes ut av instansvariablen forsteR og den siste reservasjonen pekes ut av variabelen

sisteR i klassen Hotell.

**Oppgave 5.** 8 poeng

Deklarer de to variablene forsteR og sisteR i klassen Hotell. Programmer de to

klassene Gjest og Reservasjon. For enkelhets skyld skal du ikke lage noen konstruktører

eller metoder, og alle instansvariabler skal være åpent tilgjengelig i klassen Hotell. (Anta

at hele programmet ditt er deklarert i samme mappe/pakke.)

*Læringsmål oppgave 5.*

*Referansene forsteR og sisteR må være av typen Reservasjon.*

*Du skal også vise at du kan programmere to enkle klasser. Klassen Reservasjon må ha riktige*

*instansvariabler slik som angitt i oppgaveteksten. Instansvariablene børikke være private,*

*og siden det står de skal være tilgjengelig i hele mappe/pakken bør de være “friendly”, dvs.*

*uten aksessmodifikator (men dette vurderes mildt siden de har fått beskjed om at*

*aksessmodifikatorer ikke er viktige på eksamen)*

I det følgende kan du regne med at det i programmet allerede er opprettet objekter for

gjester og reservasjoner og at alle reservasjoner er kjedet sammen i den dobbeltlenkede

listen.

Du skal skrive en metode i klassen Hotell: void tildelRom(String navn).

Den skal brukes når en ny gjest som allerede har en reservasjon, kommer til hotellet.

Metoden skal først gå gjennom listen av reservasjoner og finne den reservasjonen som er

gjort av gjesten med dette navnet. Deretter skal reservasjonen tas ut av listen av

reservasjoner. La metoden kaste et egendefinert runtime-unntak med gjestens navn om

gjesten ikke har noen reservasjon.

Så skal metoden bruke hjelpemetoden Rom finnRom(int antSeng, boolean

kjokken) til å finne et egnet rom. finnRom() skal gå gjennom alle rommene i hotellet for

å finne et rom som er ledig og som tilfredsstiller kriteriene i reservasjonen. Antall

sengeplasser i rommet skal være det samme som i reservasjonen. Eventuelt ønske om

kjøkken må også tilfredsstilles, men en reservasjon som ikke har markert for ønske om

kjøkken, kan gjerne få det allikevel. Hvis det ikke finnes et egnet rom, skal finnRom()

returnere null.

Hvis finnRom() ikke finner et ledig rom med riktig antall sengeplasser, skal metoden

tildelRom() gå i løkke og øke antall sengeplasser med én og kalle finnRom() på nytt

helt til et rom er funnet.

Når et rom er funnet, settes det til opptatt, informasjon om rommet skrives ut i

terminalvinduet (bruk toString()-metoden til dette) og en referanse til rommet settes

inn i gjestens objekt.

Hvis tildelRom() har prøvd antall sengeplasser opp til MAX\_ANTALL\_SENGEPLASSER og

det ikke er funnet noe ledig rom, skal metoden kaste et egendefinert runtime-unntak med

navnet på gjesten og antall sengeplasser som ønskes.

**Opppgave 6.** 10 poeng

Skriv hjelpemetoden med signaturen Rom finnRom(int antSeng, boolean

kjokken) som går gjennom listen av rom og returnerer referansen til et ledig rom med

antall sengeplasser som angitt i parameteren og med kjøkken hvis andre parameter er

true. Om det ikke finnes noe slikt rom, skal metoden returnere null.

*Læringsmål oppgave 6.*

*Her skal du vise at du kan skrive en metode som leter gjennom en*

*lenket liste. Operatoren instanceof kan brukes for å teste om objektet har egenskapen*

*Kjøkken (polymorfi er også en mulighet). Løkken må startes på riktig måte, løkken skal*

*terminere når et Rom er funnet eller på slutten av listen. Riktig verdi må returneres.*

**Oppgave 7.** 10 poeng

Skriv hjelpemetoden void taUtRes(Reservasjon r) som tar ut reservasjonen r fra

den dobbeltkjedede listen av reservasjoner. Vi kan anta at r er i listen.

*Læringsmål oppgave 7.*

*Her skal du vise at du kan programmere en algoritme som fjerner et*

*element fra en dobbeltkjedet liste. Det er viktig at programmet behandler tilfellet at det bare*

*er én reservasjon i listen og at objektet først og sist i listen tas ut riktig.*

**Oppgave 8.** 8 poeng

Skriv metoden void tildelRom(String navn) i klassen Hotell. Metoden skal kalle

hjelpemetodene finnRom() og taUtRes(). Deklarer også de to egendefinerte unntakene.

*Læringsmål oppgave 8.*

*Her skal du vise at du kan programmere en litt mer kompleks*

*algoritme og bruke metoder som allerede er skrevet. Uten at det står eksplisitt i oppgaven*

*bør tilfellet at det ikke finnes noe passende rom også håndteres riktig. Metoden må først*

*lete etter reservasjon med riktig navn, så kalle taUtRes() og så gå i løkke og kalle finnRom()*

*med økende antall sengeplasser til et rom er funnet eller maksimum antall sengeplasser er*

*nådd. Instansvariablen i Gjest-objektet må settes til å peke på dette rommet. Det må lages*

*to Unntaks-klasser og unntak må deklareres og kastes riktig.*

## Hotell med flere etasjer

Vi antar nå at hotellet har flere etasjer. For å holde oversikt over hvilke rom som er i hvilke

etasjer, skal du innføre en ny datastruktur i hotellet. Hotellet har et antall etasjer som skal

være en konstant (ANTALL\_ETASJER) i klassen Hotell. Verdien til ANTALL\_ETASJER skal

settes i konstruktøren.

Alle rommene i samme etasje skal lenkes sammen i en enkelkjedet liste.

Referansen til det første rommet i hver etasje er inneholdt i en array kalt

forsteRomEtasje. Array-elementet med indeks k peker således på listen av rom i etasje

k. Vi skal også ta med etasje 0 (kjelleren), slik at arrayen skal være ANTALL\_ETASJER+1

lang.

I hele resten av dette oppgavesettet skal du regne med at klassen Hotell er slik at alle

rommene er organisert i etasjer slik som beskrevet her.

**Oppgave 9.** 5 poeng

Tegn opp datastrukturen i et hotell med fire etasjer (ANTALL\_ETASJER = 4), der det er to

eller tre rom i hver etasje, men ingen rom i andre etasje eller i kjelleren. Tegn opp alle

referanser, men av andre instansvariabler skal du bare tegne opp etasjenummer og antall

kvadratmeter.

*Læringsmål oppgave 9.*

*Her skal du vise at du skjønner hvordan en datastruktur med en*

*array der elementene refererer starten av lenkede lister av objekter ser ut.*

*På tre arrayplasser må arrayaelementet peke på første objekt i en liste av to eller tre rom.*

*To etasjer er tomme dvs. det er null-verdier i arrayen.*

*Det er ok om tegningen ikke viser hvilke egenskaper som evt. arves fra superklasser (som i lf)*

**Oppgave 10.** 1 poeng

Deklarer klassen Hotell med konstruktør og arrayen forsteRomEtasje , men ennå ingen

metoder.

*Læringsmål oppgave 10.*

*Her skal du vise at du kan programmere en klasse med en array og*

*en konstruktør. Arrayen må opprettes av konstruktøren med riktig lenge som gis av*

*parameter til konstruktøren.*

Du skal nå skrive en iterator over alle rommene i hotellet. Iteratoren må kunne håndtere det

tilfellet at det finnes etasjer der det ikke er noen rom.

**Oppgave 11.** 14 poeng

Skriv en iterator over alle rommene i klassen Hotell.

*Læringsmål oppgave 11.*

*Her skal du vise at du kan skrive en iterator. Klassen Hotel må*

*implementere Iterabale<Rom> og det må finnes en metode som heter iterator() som*

*returnerer en referanse til et objekt av en klasse som implementerer Iterator. Denne klassen*

*inneholder next() og hasNext() og må programmeres på riktig måte. Antagelig er det best å*

*ha en konstruktør til å initialisere variablene i denne klassen.*

Du skal nå skrive en metode med signaturen int[] ledigeRom() i klassen Hotell som

finner antall ledige rom fordelt på antall sengeplasser i hvert rom. Det betyr at antall ledige

rom med 1 sengeplass skal legges i resultat-arrayen på indeksplass 0, antall ledige rom med

2 sengeplasser skal legges i resultatarrayen på indeksplass 1, osv. opp til at antall ledige rom

med MAX\_ANT\_SENGEPLASSER sengeplasser skal legges inn i resultat-arrayen på indeks

MAX\_ANT\_SENGEPLASSER-1.

**Oppgave 12.** 8 poeng

Bruk iteratoren du skrev i oppgave 11 til å skrive metoden ledigeRom().

*Læringsmål oppgave 12.*

*Her skal du vise at du kan bruke iteratoren du selv har skrevet. Det*

*gjøres mest elegant med en for-each-løkke, men eksplisitte kall på hasNext() og next() kan*

*også godtas. Metoden må opprette en array av riktig lenge og øke verdien på riktig plass i*

*arrayen med én hver gang metoden finner et ledig rom.*

## Programmering med tråder

Bedriften du har søkt jobb i, har mange hoteller spredt rundt i hele verden. Noen

ganger trenger administrasjonen av hotellkjeden å få rapporter i løpet av svært kort

tid, og det er derfor viktig at den som tilsettes, kan programmere med tråder. I denne

oppgaven skal du lage en rapport om hvor mange ledige rom av hver størrelse (dvs antall

sengeplasser) hele hotellkjeden har på tvers av alle hotellene i kjeden.

Anta at vi har en klasse Hotellkjede med konstantene ANTALL\_HOTELLER og

KJEDE\_MAKS\_ANTALL\_SENGEPLASSER i tillegg til en array kalt alleHoteller med alle

hotellene. Du skal ikke skrive denne klassen. I resten av oppgaven kan du anta at arrayen

alleHoteller er fylt opp med hoteller.

Til oppgaven trenger vi en trådklasse. Hver tråd jobber på hvert sitt hotell og trenger

derfor en referanse til hotellet det jobber på. Når en tråd kjøres, skal den hente ut hvor

mange ledige rom det er i dette hotellet fordelt på antall sengeplasser per rom. Det er jo

nettopp det metoden du programmerte i oppgave 12 gjør; derfor skal tråden din kalle

denne metoden.

For å samle og summere opp verdiene fra de ulike trådene skal du lage en monitor. Trådene

skal sende inn sine resultater til metoden void rapporterLedigeRom(int[] ledige)

i monitoren. (Derfor trenger trådene en referanse til en monitor.) Monitoren skal også ha

en metode int[] hentLedigeRom() som henter ut antallet ledige rom fordelt på antall

sengeplasser i hvert rom etter at alle trådene er ferdig.

**Oppgave 13.** 6 poeng

Skriv monitoren slik den er beskrevet over.

*Læringsmål oppgave 13.*

*Her skal du vise at du kan skrive en meget enkel monitor. Det må*

*deklareres en lås og innholdet i metoden rapporterLedigeRom() må beskyttes av denne*

*tråden. Monitoren må også inneholde en array av lengde*

*KJEDE\_MAKS\_ANTALL\_SENGEPLASSER og verdiene i arrayen som er parameter må*

*kopieres inn i denne arrayen. Å bruke original Java med synchronized er også mulig.*

**Oppgave 14.** 6 poeng

Skriv trådklassen.

*Læringsmål oppgave 14.*

*Her skal du vise at du kan skrive en meget enkel klasse som implementerer interface-et Runnable med metoden run(). Klassen må ha en konstruktør som angir hvilket hotell tråden skal jobbe på, og run-metoden skal bare kalle ledigeRommetoden fra oppgave 12 og så kalle monitoren med resultatet. Hvis metoden skrivUtLedigeRomMedTrader() venter vha. en barriere, må den være med i konstruktøren og telles ned på slutten av run()-metoden. Barrieren kan f.eks. vær en CountDownLatch. Subklassing av klassen Thread er også mulig.*

**Oppgave 15.** 6 poeng

Skriv en metode med signaturen void skrivUtLedigeRomMedTrader() i klassen Hotellkjede. Metoden skal opprette monitoren og opprette og starte alle trådene. Når alle trådene er ferdige, skal metoden hente ut resultatet fra monitoren og skrive ut dette resultatet i kommandovinduet.

*Læringsmål oppgave 15.*

*Her skal du vise at du kan opprette et monitor-objekt, opprette og starte mange tråder, vent på at trådene er terminert og skrive ut resultatet. Aktuelle parametre til trådenes konstruktører må være riktig. Venting kan f.eks. gjøres med en egen barriere eller vha. join().*

# Løsningsforslag

import java.util.concurrent.\*;

import java.util.concurrent.locks.\*;

import java.util.Iterator;

abstract class Rom {

    final int ROM\_NR, ROM\_KVM, ETASJE, ANT\_SENGEPLASSER;

    boolean erLedig = true;

    Rom nesteRom = null;

    Rom nesteRomEtasje = null;

    Rom (int nr, int kvm, int etasje, int antSeng) {

    ROM\_NR = nr;  ROM\_KVM = kvm;

    ETASJE = etasje;  ANT\_SENGEPLASSER = antSeng;

    }

    // Hjelpemetode for å finne egnet rom:

    boolean erEgnet (int antSeng, boolean kjokken) {

    if (! erLedig) return false;

    if (antSeng != ANT\_SENGEPLASSER) return false;

    if (kjokken) return false;

    return true;

    }

    @Override

    public String toString () {

    return ROM\_NR + " (" + ETASJE + " etasje) med " + ROM\_KVM + " m2 og " +

        ANT\_SENGEPLASSER + " sengeplasser [" +

        (erLedig ? "ledig" : "opptatt") + "]";

    }

}

class EnkeltRom extends Rom {

    EnkeltRom (int nr, int kvm, int etasje, int antSeng) {

    super(nr, kvm, etasje, antSeng);

    }

    @Override

    public String toString () {

    return "Enkeltrom " + super.toString();

    }

}

class VanligRom extends Rom {

    VanligRom (int nr, int kvm, int etasje, int antSeng) {

    super(nr, kvm, etasje, antSeng);

    }

    @Override

    public String toString () {

    return "Vanlig rom " + super.toString();

    }

}

class Suite extends Rom {

    Suite (int nr, int kvm, int etasje, int antSeng) {

    super(nr, kvm, etasje, antSeng);

    }

    @Override

    public String toString () {

    return "Suite " + super.toString();

    }

}

interface Kjokken {

    int hentKjokkenStorrelse ();

}

class VanligRomMedKjokken extends VanligRom implements Kjokken {

    final int KJOKKEN\_KVM;

    VanligRomMedKjokken (int nr, int kvm, int etasje, int antSeng, int kjKvm) {

    super(nr, kvm, etasje, antSeng);

    KJOKKEN\_KVM = kjKvm;

    }

    @Override

    public int hentKjokkenStorrelse () {

    return KJOKKEN\_KVM;

    }

    // Hjelpemetode for å finne egnet rom:

    @Override

    boolean erEgnet (int antSeng, boolean kjokken) {

    if (! erLedig) return false;

    if (antSeng != ANT\_SENGEPLASSER) return false;

    return true;

    }

    @Override

    public String toString() {

    return super.toString() + " med " + KJOKKEN\_KVM + " m2 kjokken";

    }

}

class SuiteMedKjokken extends Suite implements Kjokken {

    final int KJOKKEN\_KVM;

    SuiteMedKjokken (int nr, int kvm, int etasje, int antSeng, int kjKvm) {

    super(nr, kvm, etasje, antSeng);

    KJOKKEN\_KVM = kjKvm;

    }

    @Override

    public int hentKjokkenStorrelse () {

    return KJOKKEN\_KVM;

    }

    // Hjelpemetode for å finne egnet rom:

    @Override

    boolean erEgnet (int antSeng, boolean kjokken) {

    if (! erLedig) return false;

    if (antSeng != ANT\_SENGEPLASSER) return false;

    return true;

    }

    @Override

    public String toString() {

    return super.toString() + " med " + KJOKKEN\_KVM + " m2 kjokken";

    }

}

class Hotell implements Iterable<Rom> {

    final int MAX\_ANT\_SENGEPLASSER = 8;

    final int ANTALL\_ETASJER;

    Rom forsteRom = null;

    Reservasjon forsteR = null, sisteR = null;

    Rom[] forsteRomEtasje;

    Hotell (int antEt) {

    ANTALL\_ETASJER = antEt;

    forsteRomEtasje = new Rom[ANTALL\_ETASJER+1];

    }

    // Hjelpemetode: Finn en gjests reservasjon:

    Reservasjon finnRes (String navn) {

    Reservasjon p = forsteR;

    while (p != null) {

        if (p.gjest.navn.equals(navn))

        return p;

        p = p.nesteR;

    }

    return null;

    }

    void tildelRom (String navn) {

    // Finn reservasjonen:

    Reservasjon res = finnRes(navn);

    if (res == null) throw new IngenReservasjon(navn);

    taUtRes(res);

    // Finn egnet rom:

    Rom rom = null;

    for (int sx = res.trengerSenger;  sx <= MAX\_ANT\_SENGEPLASSER;  ++sx) {

        rom = finnRom(sx, res.onskerKjokken);

        if (rom != null) break;

    }

    if (rom == null)

        throw new IntetLedigRom(navn, res.trengerSenger);

    // Tildel rommet:

    rom.erLedig = false;

    res.gjest.rom = rom;

    System.out.println(navn + " har fatt tildelt " + rom);

    }

    Rom finnRom (int antSeng, boolean kjokken) {

    Rom romP = forsteRom;

    while (romP != null) {

        if (romP.erEgnet(antSeng,kjokken))

        return romP;

        romP = romP.nesteRom;

    }

    return null;

    }

    void taUtRes (Reservasjon r) {

    if (r == forsteR && forsteR == sisteR) {

        forsteR = sisteR = null;

    } else if (r == forsteR) {

        forsteR = forsteR.nesteR;

        forsteR.forrigeR = null;

    } else if (r == sisteR) {

        sisteR = sisteR.forrigeR;

        sisteR.nesteR = null;

    } else {

        r.forrigeR.nesteR = r.nesteR;

        r.nesteR.forrigeR = r.forrigeR;

    }

    r.forrigeR = r.nesteR = null;

    }

    class RomIterator implements Iterator<Rom> {

    int denneEtasjen;

    Rom detteRommet;

    RomIterator () {

        denneEtasjen = 0;

        detteRommet = forsteRomEtasje[0];

        finnEtRom();

    }

    void finnEtRom () {

        while (denneEtasjen<=ANTALL\_ETASJER && detteRommet==null) {

        ++denneEtasjen;

        if (denneEtasjen <= ANTALL\_ETASJER)

            detteRommet = forsteRomEtasje[denneEtasjen];

        }

    }

    @Override

    public boolean hasNext () {

        return denneEtasjen <= ANTALL\_ETASJER;

    }

    @Override

    public Rom next () {

        Rom svar = detteRommet;

        detteRommet = detteRommet.nesteRomEtasje;

        finnEtRom();

        return svar;

    }

    }

    public RomIterator iterator () {

    return new RomIterator();

    }

    int[] ledigeRom () {

    int[] ledige = new int[MAX\_ANT\_SENGEPLASSER];

    for (Rom r: this)

        if (r.erLedig) ++ledige[r.ANT\_SENGEPLASSER-1];

    return ledige;

    }

    // Testprogram (ikke bedt om i oppgaven):

    public static void main (String[] arg) {

    Hotell h = new Hotell(4);

    Rom rom1 = new EnkeltRom(1, 15, 0, 1);

    Rom rom2 = new VanligRom(21, 20, 2, 2);

    Rom rom3 = new VanligRomMedKjokken(22, 25, 2, 3, 6);

    Rom rom4 = new SuiteMedKjokken(31, 40, 3, 6, 10);

    h.forsteRom = rom1;  rom1.nesteRom = rom2;

    rom2.nesteRom = rom3; rom3.nesteRom = rom4;

    h.forsteRomEtasje[0] = rom1;

    h.forsteRomEtasje[2] = rom2;  rom2.nesteRomEtasje = rom3;

    h.forsteRomEtasje[3] = rom4;

    System.out.println("Rommene er;");

    for (Rom r: h)

        System.out.println("  " + r);

    System.out.println();

    Gjest g1 = new Gjest();  g1.navn = "Dag";

    Gjest g2 = new Gjest();  g2.navn = "Siri";

    Gjest g3 = new Gjest();  g3.navn = "Stein";

    Reservasjon res1 = new Reservasjon();

    res1.gjest = g1;  res1.trengerSenger = 5;  res1.onskerKjokken = false;

    Reservasjon res2 = new Reservasjon();

    res2.gjest = g2;  res2.trengerSenger = 2;  res2.onskerKjokken = true;

    Reservasjon res3 = new Reservasjon();

    res3.gjest = g3;  res3.trengerSenger = 2;  res3.onskerKjokken = true;

    h.forsteR = res1;  h.sisteR = res3;

    res1.nesteR = res2;  res2.nesteR = res3;

    res2.forrigeR = res1;  res3.forrigeR = res2;

    Hotellkjede kjede = new Hotellkjede();

    kjede.alleHoteller[0] = h;

    kjede.alleHoteller[1] = new Hotell(2);

    kjede.skrivUtLedigeRomMedTrader();

    System.out.println();

    String[] navneliste = { "Dag", "Anne", "Stein", "Siri" };

    for (String s: navneliste) {

        try {

        h.tildelRom(s);

        } catch (IngenReservasjon e) {

        System.out.println(e);

        } catch (IntetLedigRom e) {

        System.out.println(e);

        }

    }

    System.out.println();

    kjede.skrivUtLedigeRomMedTrader();

    }

}

class Hotellkjede {

    final int ANTALL\_HOTELLER = 2;

    final int KJEDE\_MAKS\_ANTALL\_SENGEPLASSER = 10;

    Hotell[] alleHoteller = new Hotell[ANTALL\_HOTELLER];

    void skrivUtLedigeRomMedTrader () {

    Monitor mon = new Monitor(KJEDE\_MAKS\_ANTALL\_SENGEPLASSER);

    CountDownLatch teller = new CountDownLatch(ANTALL\_HOTELLER);

    for (Hotell h: alleHoteller)

        new Thread(new RomTeller(h,mon,teller)).start();

    try {

        teller.await();

    } catch (InterruptedException e) {}

    System.out.println("Totalt antall ledige rom:");

    int[] ledige = mon.hentLedigeRom();

    for (int i = 0;  i < ledige.length;  ++i)

        System.out.printf("%4d sengeplasser: %6d\n", i+1, ledige[i]);

    }

}

class RomTeller implements Runnable {

    Hotell hotell;

    Monitor mon;

    CountDownLatch teller;

    RomTeller (Hotell h, Monitor m, CountDownLatch c) {

    hotell = h;  mon = m;  teller = c;

    }

    @Override

    public void run () {

    mon.rapporterLedigeRom(hotell.ledigeRom());

    teller.countDown();

    }

}

class Monitor {

    Lock las = new ReentrantLock();

    int[] sumLedige;

    Monitor (int maxSeng) {

    sumLedige = new int[maxSeng];

    }

    void rapporterLedigeRom(int[] ledige) {

    las.lock();

    // Trenger ikke try-finally siden ingen trådoperasjoner i denne løkken.

    for (int i = 0;  i < ledige.length;  ++i)

        sumLedige[i] += ledige[i];

    las.unlock();

    }

    int[] hentLedigeRom() {

    // Trenger ikke beskyttes siden alle produsenter er ferdige.

    return sumLedige;

    }

}

class IngenReservasjon extends RuntimeException {

    IngenReservasjon (String navn) {

    super(navn + " har ingen reservasjon");

    }

}

class IntetLedigRom extends RuntimeException {

    IntetLedigRom (String navn, int antSeng) {

    super("Intet ledig rom med " + antSeng + " sengeplasser til " + navn);

    }

}

class Gjest {

    String navn;

    Rom rom = null;

}

class Reservasjon {

    Gjest gjest;

    int trengerSenger;

    boolean onskerKjokken;

    Reservasjon nesteR, forrigeR;

}