```
/**
 * Obligatorisk Oppgave 3
 * Program som holder oversikt over et hotells rom, hvorav rom er booket eller ikke.
 * Programmet benytter seg av en hotellrom baseklasse, med to subklasser enkelt- og
 * dobbeltrom. Programmet leser inn bookede hotellrom fra filen ('HOTEL.txt') når
 * programmet starter, og lagrer bookede rom på samme fil når programmet avsluttes.
 *
 * .txt fil-extension er benyttet fordi programemt er skrevet og kjører på MacOS.
 *
 *
 *  @file Oblig3.cpp
 *  @author  av den innledende/startende koden:
 *           Malin Foss,  Markus Olsvik,  Øystein Qvigstad  &  FrodeH, NTNU
 *
 *  @author Steffen S. Martinsen
 */


#include <iostream>
#include <fstream>
#include <string>
#include <list>
#include <map>
#include "LesData2.h"
using namespace std;


const int MAXDOGN             =   14,    ///<  Max.antall døgn mulig å booke.
          LAVESTEROMNR        =  100,    ///<  Hotellets laveste romnummer.
          HOYESTEROMNR        =  999,    ///<  Hotellets høyeste romnummer.
          PRISENKELTROM       = 1000,    ///<  Døgnpris for enkeltrom.
          PRISDOBBELTROM      = 1500,    ///<  Døgnpris for dobbeltrom.
          PRISALLINCLUSIVE    = 1200,    ///<  Døgnpris for 2 stk all inclus.
          PRISFILMPAKKE       =   50,    ///<  Døgnpris for filmpakke.
          PRISFROKOST         =  170,    ///<  Frokostpris - KUN enkeltrom.
          STUDENTRABATTPROSENT =   40;   ///<  Studentrabatt i prosent
                                         //        (gjelder KUN enkeltrom).

/**
 *  Operasjonene som utføres på et rom (ser kundataene eller sjekker ut).
 */
enum romOperasjon  {  seData,  sjekkUt  };


/**
 *  Baseklassen 'Hotellrom' (med gjestens navn og antall bookede døgn).
 */
class Hotellrom {
    protected:
        string navn;                        //  Bookers/gjestens navn.
        int antallDager;                    //  Oppholdets varighet.

    public:
        Hotellrom() {  }
```

```
        Hotellrom(ifstream & inn);
        virtual ~Hotellrom()  {  }
        string hentNavn() const  {  return navn;  }
        virtual void lesData();
        virtual void skrivData() const = 0;  // Pure virtual - dvs. subklasser
        virtual void skrivHoveddata() const = 0; // MÅ lage disse funksjonene.
        virtual void skrivTilFil(ofstream & ut, int romNr) const;
};


/**
 *  Avledet klasse 'Enkeltrom' (med om har frokost og evt studentrabatt).
 */
class Enkeltrom : public Hotellrom {
    private:
        bool frokost,                    //  Bare enkeltrom har studentrabatt!
             studentRabatt;              //  Bare enkeltrom har frokost!

    public:
        Enkeltrom()  {  }
        Enkeltrom(ifstream & inn);
        virtual ~Enkeltrom()  {  }
        virtual void lesData();
        virtual void skrivData() const;
        virtual void skrivHoveddata() const;
        virtual void skrivTilFil(ofstream & ut, int romNr) const;
};


/**
 *  Avledet klasse 'Dobbeltrom' (med om har All Inclusive og/eller filmpakke).
 */
class Dobbeltrom : public Hotellrom {
    private:
        bool allInclusive,           //  Bare dobbeltrom har 'All inclusive'!
             filmpakke;              //  Bare dobbeltrom har filmpakke!

    public:
        Dobbeltrom()  {  }
        Dobbeltrom(ifstream & inn);
        virtual ~Dobbeltrom()  {  }
        virtual void lesData();
        virtual void skrivData() const;
        virtual void skrivHoveddata() const;
        virtual void skrivTilFil(ofstream & ut, int romNr) const;
};


void bookRom();
void lesFraFil();
void rom(const romOperasjon sDsU);
void skrivAlleRommene();
void skrivBookedeRomnumre();
void skrivMeny();
void skrivTilFil();
```

```
map <int, Hotellrom*> gHotellRommene;      ///<  Alle hotellrommene.


/**
 *  Hovedprogram.
 */
int main() {
    char menyvalg;

    lesFraFil();

    skrivMeny();
    menyvalg = lesChar("\nMenyvalg");

    while (menyvalg != 'Q')  {
        switch (menyvalg) {
            case 'S':  skrivAlleRommene(); break;
            case 'B':  bookRom();          break;
            case 'F':  rom(seData);        break;
            case 'U':  rom(sjekkUt);       break;
            default:   skrivMeny();        break;
        }
        menyvalg = lesChar("\nMenyvalg");
    }

   skrivTilFil();

   return 0;
}


/**
 * Leser inn alle baseklassens datamedlemmer fra fil.
 *
 * @param   inn - Filobjektet data leses inn fra
 */
Hotellrom::Hotellrom(ifstream & inn) {
    getline(inn, navn);
    inn >> antallDager;
    inn.ignore();
}


/**
 * Funksjon som leser inn alle datamedlemmene til baseklassen.
 */
void Hotellrom::lesData() {
    cout << "Gjestens navn: ";
    getline(cin, navn);
    antallDager = lesInt("Antall dager", 1, 31);   // Setter grense på ≈1 mnd. per
booking.
}
```

```
/**
 * Funksjon som skriver alle klassens datamedlemmer til fil.
 *
 * @param   ut - Filobjektet datamedlemmene skrives til.
 */
void Hotellrom::skrivTilFil(ofstream & ut, int romNr) const {
    ut << " " << romNr << " " << navn << "\n" << antallDager << "\n";
}


/**
 *  Leser inn alle subklassens og baseklassens datamedlemmer.
 *
 * @param   inn - Filobjektet datamedlemmene leses fra.
 */
Enkeltrom::Enkeltrom(ifstream & inn) : Hotellrom(inn) {
    inn >> frokost >> studentRabatt;
    inn.ignore();
}


/**
 * Funksjon som leser inn alle subklassen og subklassens datamedlemmer.
 *
 * @see     Hotellrom::lesData();
 */
void Enkeltrom::lesData() {
    Hotellrom::lesData();
    cout << "1 = Ja\n0 = Nei\n";
    frokost         = lesInt("Frokost", 0, 1);
    studentRabatt   = lesInt("Studentrabbat", 0, 1);
}


/**
 * Funksjon som skriver ut samlet sum for et enkeltrom avhengig av om gjesten
 * har bestilt frokost og har studentrabatt.
 *
 * @see     Hotellrom::hentNavn()
 */
void Enkeltrom::skrivData() const {
    float sum = 0;
    cout << "Gjestens navn: " << Hotellrom::hentNavn() << ".\n"
         << "Enkeltrom\t- " << antallDager << " Dager\t"
         << PRISENKELTROM * antallDager << ",-\n";
         sum += PRISENKELTROM * antallDager;

        if (frokost) {
            cout << "Frokost\t\t- " << antallDager << " dager\t"
            << PRISFROKOST * antallDager << ",-\n";
            sum += PRISFROKOST * antallDager;
        }

        if (studentRabatt) {
```

```
            cout << "Studentrabatt:\t- " << antallDager << " dager\t-"
            << (sum*0.4) << ",-.\n";
            sum = sum - (sum*0.4);
        }

        cout << "Sum: \t\t\t\t" << fixed << setprecision(2) << sum << ",-\n";
}


/**
 * Funksjon som sjekker om frokost og studentrabatt gjelder for et rom, skriver så ut
 * det om det er true.
 */
void Enkeltrom::skrivHoveddata() const {
                        cout << "Enkeltrom\n";
    if (frokost)        cout << "Frokost\n";
    if (studentRabatt)  cout << "Studentrabatt\n";
}


/**
 * Funksjon som skriver alle baseklassen og klassens datamedlemmer til fil.
 *
 * @param   ut - Filobjektet datamedlemmene skrives til.
 * @see     Hotellrom::skrivTilFil(...)
 */
void Enkeltrom::skrivTilFil(ofstream & ut, int romNr) const {
    ut << 'E';              // Subklasse av typen 'E'
    Hotellrom::skrivTilFil(ut, romNr);
    ut  << frokost << " " << studentRabatt << "\n";
}


/**
 * Leser inn alle subklassens og baseklassens datamedlemmer.
 *
 * @param   inn - Filobjektet datamedlemmene leses fra.
 */
Dobbeltrom::Dobbeltrom(ifstream & inn) : Hotellrom(inn) {
    inn >> allInclusive >> filmpakke;
    inn.ignore();
}


/**
 * Funksjon som leser inn alle subklassen og subklassens datamedlemmer.
 *
 * @see     Hotellrom::lesData();
 */
void Dobbeltrom::lesData() {
    Hotellrom::lesData();
    cout << "1 = Ja\n0 = Nei\n";
    allInclusive    = lesInt("All inclusive", 0, 1);
    filmpakke       = lesInt("Filmpakke", 0, 1);
}
```

```
/**
 * Funksjon som skriver ut samlet sum for et dobbeltrom avhengig av om gjesten
 * har bestilt all inclusive og/eller filmpakke.
 *
 * @see     Hotellrom::hentNavn()
 */
void Dobbeltrom::skrivData() const {
    float sum = 0;
    cout << "Gjestens navn: " << Hotellrom::hentNavn() << ".\n"
         << "Dobbeltrom\t- " << antallDager << " dager:\t"
         << PRISDOBBELTROM * antallDager << ",-\n";
         sum += PRISDOBBELTROM * antallDager;
        if (allInclusive) {
            cout << "All inclusive\t- " << antallDager << " dager:\t"
                 << PRISALLINCLUSIVE * antallDager << ",-\n";
            sum += allInclusive * antallDager;
        }
        if (filmpakke) {
            cout << "Filmpakke\t- " << antallDager << " dager:\t"
                 << PRISFILMPAKKE * antallDager << ",-.\n";
            sum += filmpakke * antallDager;
        }
        cout << "Sum:\t\t\t\t" << fixed << setprecision(2) << sum << ",-\n";
}


/**
 * Funksjon som sjekker om et rom inneholder all inclusive og filmpakke,
 * skriver ut dette om det er true.
 */
void Dobbeltrom::skrivHoveddata() const {
                        cout << "Dobbeltrom\n";
    if (allInclusive)   cout << "All Inclusive\n";
    if (filmpakke)      cout << "Filmpakke\n";
}


/**
 * Funksjon som skriver alle baseklassen og klassens datamedlemmer til fil.
 *
 * @param   ut - Filobjektet datamedlemmene skrives til.
 * @see         Hotellrom::skrivTilFil(...)
 */
void Dobbeltrom::skrivTilFil(ofstream & ut, int romNr) const {
    ut << "D";                          // Subklasse av typen 'D'
    Hotellrom::skrivTilFil(ut, romNr);
    ut << allInclusive << " " << filmpakke << "\n";
}


/**
 * Funksjon som leser inn alle hotellrom fra fil og legger de inn i hotelromlisten.
 */
```

```
void lesFraFil() {
    ifstream innfil("HOTEL.txt");  // .txt format pga MacOS
    char romType;
    int romNr;

    if (innfil) {
        cout << "Leser inn fra 'HOTEL.txt'\n";
        innfil >> romType >> romNr;
        innfil.ignore();
        while (!innfil.eof()) {
            switch(romType) {
                case 'E': gHotellRommene[romNr] = new Enkeltrom(innfil);    break;
                case 'D': gHotellRommene[romNr] = new Dobbeltrom(innfil);   break;
                default: cout << "Finner ikke romtypen.\n";
            }
            innfil >> romType >> romNr;
            innfil.ignore();
        }
        innfil.close();
    } else {
        cout << "Finner ikke filen 'HOTEL.txt'\n";
    }
}


/**
 * Skriver et roms datamedlemmer til en gitt fil.
 *
 * @see     virtual Hotellrom::skrivTilFil(...)
 */
void skrivTilFil() {
    ofstream utfil("HOTEL.txt");        // .txt grunnet MacOS
    cout << "Skriver til filen 'HOTEL.txt'.\n";
    for (const auto & val : gHotellRommene) {
        val.second->skrivTilFil(utfil, val.first);
    }
    utfil.close();
}


/**
 * Funksjon som skriver ut menyen til brukeren.
 */
void skrivMeny() {
    cout << "Vennligst velg et alternativ: \n"
         << "S - Skriv alle rommene\n"
         << "B - Book et rom\n"
         << "F - Se data om et rom\n"
         << "U - Sjekk ut\n\n"
         << "Q - Avslutt\n";
}


/**
 * Funksjon som skriver ut romnumrene til bookede rom.
```

```
 */
void skrivBookedeRomnumre() {
    cout << "Alle bookede hotellrom: \n";
    if (gHotellRommene.size() > 0) {
        for (const auto & val : gHotellRommene) {
            cout << val.first << "\n";
        }
    } else {
        cout << "Finnes ingen bookede hotellrom.\n";
    }
}


/**
 * Funksjon som skriver ut alle rom som er booket. Skriver ut navnet på romgjesten,
 * samt om det er enkeltrom/dobbeltrom, og hvilke tjenester/fordeler som knyttes til
rommet.
 *
 * @see     Hotellrom::hentNavn()
 * @see     virtual Hotellrom::skrivHoveddata()
 */
void skrivAlleRommene() {
    if (gHotellRommene.size() > 0) {
        for (const auto & val : gHotellRommene) {
            cout << "\nNavn: " << val.second->hentNavn() << "\n";
            val.second->skrivHoveddata();
        }
    } else {
        cout << "Finnes ingen bookede rom.\n";
    }
}


/**
 * Funksjon som booker nytt rom. Spør etter øsnket rom og undersøker om rommet er
opptatt.
 * Funksjonen skiller mellom booking av enkeltrom og dobbeltrom.
 *
 * @see     skrivBookedeRomnumre()
 * @see     virtual Hotellrom::lesData()
 */
void bookRom() {
    int onsketRom;
    char romType;
    bool opptatt = false;
    Hotellrom* hotellrom = nullptr;
    skrivBookedeRomnumre();

    // Sjekker om ønsket rom er opptatt.
    onsketRom = lesInt("Les inn ønsket rom", LAVESTEROMNR, HOYESTEROMNR);
    for (const auto & val : gHotellRommene) {
        if (onsketRom == val.first) {
            opptatt = true;
        }
    }
```

```
    if (!opptatt) {
        do {     // Sørger for at enten 'E eller 'D' skrives inn.
            romType = lesChar("Romtype ((E)nkel/(D)obbel)");
        } while ((romType != 'E') && (romType != 'D'));

        switch (romType) {
            case 'E': hotellrom = new Enkeltrom;    break;
            case 'D': hotellrom = new Dobbeltrom;   break;
            default: cout << "Finnes ingen rom med denne betegnelsen.\n";
        }
        hotellrom->lesData();
        gHotellRommene[onsketRom] = hotellrom;
    } else {
        cout << onsketRom << " er allerede booket.\n";
    }
}


/**
 * Funksjon som tar inn et ønsket rom og sjekker om dette rommet er booket.
 * Er det booket vil funksjonen skrive ut fakture til dette rommet.
 * Om gjesten sjekker ut fra rommet vil rommet fjernes fra listen av bookede rom.
 *
 * @see     skrivBookedeRomnumre()
 * @see     virtual Hotellrom::skrivData()
 */
void rom(const romOperasjon sDsU) {
    int onsketRom;
    bool finnes = false;
    Hotellrom* hotellrommet = nullptr;
    skrivBookedeRomnumre();

    // Leser inn og sjekker om rommet finnes i listen.
    onsketRom = lesInt("Ønsket rom", LAVESTEROMNR, HOYESTEROMNR);
    for (const auto & val : gHotellRommene) {
        if (onsketRom == val.first) {
            finnes = true;
            hotellrommet = val.second;
        }
    }

    if (finnes) {
        hotellrommet->skrivData();
        if (sDsU == sjekkUt) {
            delete hotellrommet;
            gHotellRommene.erase(onsketRom);
        }
    } else {
        cout << "Finner ikke rom " << onsketRom << ".\n";
    }

}
```