```
/**
 * @file oblig2.cpp
 * @date 2023-02-19
 *
 * Program som holder oversikt over filmer og bøker til utlån.
 * Programmet kan registrere nye bøker eller filmer, leie ut disse,
 * levere de inn igjen, skrive ut alle registrerte bøker/filmer,
 * og alle utleide bøker/filmer. Programmet holder oversikt over antall
 * eksemplarer av en gitt bok/film og hvor mange av dem som er lånt ut.
 *
 */
#include <iostream>                 //  cout, cin
#include <string>                   //  string-klassen
#include <vector>                   //  vector-klassen
#include "LesData2.h"               //  Verktøykasse for lesing av diverse
 data
using namespace std;


const int MAKSANTALL = 5;         ///<   Maks antall/beholdning av hver bok/film.


/**
 *   2x enum'er med utlånsgjenstand-sjangre  og  bok-formater.
 */
enum Sjanger {  Barn, Drama, Fantasy, Spenning  };

enum Format  {  Pocket, Innbundet, Lydbok  };


/**
 * Låneinfo med navn på person som låner bok/film, lånedato og returfrist.
 */
class LaaneInfo {
private:
    string laanersNavn;
    int    laaneDato;   //  På formen:  AAAAMMDD
    int    returFrist;  //  På formen:  AAAAMMDD

public:
    LaaneInfo(string nvn)      {  laanersNavn = nvn;  }
    string hentNavn()   const  {  return laanersNavn; }
    void   settData(const int lD, const int rF)
                               {  laaneDato = lD;  returFrist = rF;  }
    void   skrivData()  const;
};


/**
 * Baseklassen 'Gjenstand', blant annet inneholdende vector m/utlånene.
 */
class UtlaansGjenstand {
protected:
    string  tittel;
    int     antallEksemplarer;
```

```
    vector <LaaneInfo*> utleideEksemplarer;
    Sjanger sjanger;

public:
    UtlaansGjenstand()  {  }
    UtlaansGjenstand(const string & t) { tittel = t; }
    ~UtlaansGjenstand();
    void   endreAntallEks();
    bool   erTilgjengelig() const {
               return ((antallEksemplarer - utleideEksemplarer.size()) >
                0); }
    string hentTittel() const { return tittel; }
    void   laanUt();
    void   lesData();
    void   leverInn();
    bool   sjekkDato(const int dag, const int mnd, const int aar);
    void   skrivData() const;
    void   skrivTittelOgAntall() const;
};


/**
 *  Subklassen 'Film' med spilletid og aldersgrense.
 */
class Film : public UtlaansGjenstand {

private:
    int timer;
    int minutter;
    int aldersGrense;

public:
    Film() {  }
    Film(const string & tittel) : UtlaansGjenstand(tittel) {  }
    void lesData();
    void sjekkAlderOgEvtLaanUt();
    void skrivData()   const;
    void skrivUtleid() const ;
};


/**
 *  Subklassen 'Bok' med antall sider og formatet på bok.
 */
class Bok : public UtlaansGjenstand {
    private:
        int    antallSider;
        Format bokFormat;

    public:
        Bok()  {  }
        Bok(const string & navn) : UtlaansGjenstand(navn) {  }
        void lesData();
        void skrivData()   const;
        void skrivUtleid() const;
```

```
};


void endreAntEksemplarer();
UtlaansGjenstand* finnEnGjenstand(const string tittel);
void fjernAllokertData();
void laanUtGjenstand();
void nyGjenstand();
void innleverGjenstand();
void skrivAlle();
void skrivMeny();
void skrivUtleide();


vector <Film*> gFilmene;     ///<  Vector med ALLE filmene i kartoteket.
vector <Bok*> gBokene;       ///<  Vector med ALLE bøkene  i kartoteket.

/**
 * Hovedprogram
 */
int main()  {

    char valg;

    skrivMeny();
    valg = lesChar("\n\nKommando");
    while (valg != 'Q') {
    switch (valg) {
      case 'N': nyGjenstand();          break;
      case 'U': laanUtGjenstand();      break;
      case 'I': innleverGjenstand();    break;
      case 'E': endreAntEksemplarer();  break;
      case 'A': skrivAlle();            break;
      case 'S': skrivUtleide();         break;
      default:  skrivMeny();            break;
    }
    valg = lesChar("\n\nKommando");
    }

    fjernAllokertData();
    return 0;
}


//************************  LAANEINFO:  **********************

/**
 * Funksjon som skriver ut alle datamedlemmene til klassen LaaneInfo.
 */
void LaaneInfo::skrivData()  const  {
    cout << "Navn på låner: " << laanersNavn
    << "\nDato for utlån: " << laaneDato
    << "\nReturfrist: " << returFrist << "\n\n";
}
```

```
//************************  UTLAANSGJENSTAND:  **********************

/**
 * Deconstructorfunksjon som sletter alle utleide eksemplarer fra memory.
 */
UtlaansGjenstand::~UtlaansGjenstand() {
    for (int i = 0; i < utleideEksemplarer.size(); i++) {
    delete utleideEksemplarer[i];
    }
    utleideEksemplarer.clear();
}

/**
 * Funksjon som endrer antall eksemplarer av en utlånsgjenstand.
 */
void UtlaansGjenstand::endreAntallEks() {
    int valg;   // Variabel som brukes til brukerinput to ganger.
    valg = lesInt("Ønsker du å\n1. Legge til eksemplarer\n2. Fjerne
     eksemplarer?\n", 1, 2);

    cout << "Antall eksemplarer: " << antallEksemplarer << "\n";
    if (valg == 1) {
        if (antallEksemplarer >= MAKSANTALL) {
            cout << "Kan ikke legge til flere eksemplarer!\n";
        } else {
            valg = lesInt("Velg antall eksemplarer å legge til", 0,
             MAKSANTALL-antallEksemplarer);
            antallEksemplarer += valg;
        }
    } else {
        valg = lesInt("Velg antall eksemplarer å fjerne", 0,
         antallEksemplarer-utleideEksemplarer.size());
        antallEksemplarer -= valg;
    }
}

/**
 * En funksjon som sjekker om det er flere eksempler å låne ut.
 * Om det er det leser den inn navn på låneren, lånedato og returdato,
 * og sjekker datoenes gyldighet, og konverterer til AAAAMMDD.
 * I tillegg sørger den for at returdato er senere en lånedato.
 * Deretter oppretter den et låneobjekt.
 *
 * @see sjekkDato(...)
 */
void UtlaansGjenstand::laanUt(){
    string buffer, dato;
    int laaneDato, returDato, dag, mnd, aar;
    bool datoGyldig, returGyldig;

    if (!erTilgjengelig())
        cout << "Ingen flere eksemplarer å leie ut.\n";
    else {
        do {
```

```
            cout << "Lånerens navn: "; getline(cin, buffer);
            cout << "Dato for utlån:\n";
            dag = lesInt("Dag", 1, 31);
            mnd = lesInt("Måned", 1, 12);
            aar = lesInt("År", 2022, 2030);

            // Sjekker at lånedato er gyldig og konverterer til riktig
             format
            datoGyldig = sjekkDato(dag, mnd, aar);
            dato = to_string(aar)
                 + ((mnd < 10) ? ("0" + to_string(mnd)) : to_string(mnd))
                 + ((dag < 10) ? ("0" + to_string(dag)) : to_string(dag));
            laaneDato =  stoi(dato);

            cout << "Returfrist:\n";
            dag = lesInt("Dag", 1, 31);
            mnd = lesInt("Måned", 1, 12);
            aar = lesInt("År", 2022, 2030);

            // Sjekker at returdato er gyldig og konverterer til riktig
             format
            returGyldig = sjekkDato(dag, mnd, aar);
            dato = to_string(aar)
                 + ((mnd < 10) ? ("0" + to_string(mnd)) : to_string(mnd))
                 + ((dag < 10) ? ("0" + to_string(dag)) : to_string(dag));
            returDato = stoi(dato);
            if (laaneDato > returDato)
                cout << "Returdatoen må være etter lånedatoen!\n";

        } while ((!datoGyldig) || (!returGyldig) || (!(laaneDato <
         returDato)));

        // Oppretter låneobjektet
        LaaneInfo* nyGjenstand = new LaaneInfo(buffer);
        nyGjenstand->settData(laaneDato, returDato);
        utleideEksemplarer.push_back(nyGjenstand);
        cout << "Utlån registrert.\n";
    }
}

/**
 * Funksjon som leser inn data for UtlaansGjenstand-klassen.
 */
void UtlaansGjenstand::lesData(){
    int valg;

    cout << "Tittel: "; getline(cin, tittel);
    antallEksemplarer = lesInt("Antall eksemplarer", 0, MAKSANTALL);

    valg = lesInt("Velg sjanger:\n1. Barn\n2. Drama\n3. Fantasy\n4.
     Spenning\n", 1, 4);
    switch (valg) {
    case 1: sjanger = Barn;     break;
    case 2: sjanger = Drama;    break;
    case 3: sjanger = Fantasy;  break;
```

```
    case 4: sjanger = Spenning; break;
    }

}

/**
 * Funksjon som leser inn en låners navn og sjekker om vedkommende har gjenstander lånt
 * i sitt navn. Om det er tilfellet så vil alle gjenstander i hans navn slettes.
 * Funksjonen gir egne meldinger om ingen eksemplarer er utlånt, eller om ingen eksemplarer
 * er utlånt i det gitte navnet.
 */
void UtlaansGjenstand::leverInn() {
    string buffer;
    bool funn = false;

    if (utleideEksemplarer.size() < 1)
        cout << "Ingen eksemplarer er utlånt.\n";
    else {
        cout << "Navn på låner: "; getline(cin, buffer);
        for (int i = 0; i < utleideEksemplarer.size(); i++) {
          if (utleideEksemplarer[i]->hentNavn() == buffer) {
            delete utleideEksemplarer[i];
            utleideEksemplarer[i] = nullptr;  // Setter den til nullptr til
             den peker til ny verdi
            // Sørger for å sette alle elementer en plass nedover
            for (int j = i; j < utleideEksemplarer.size()-1; j++) {
              utleideEksemplarer[j] = utleideEksemplarer[j+1];
              utleideEksemplarer[j+1] = nullptr;  // Setter den til nullptr
               til den peker til ny verdi
            }
            utleideEksemplarer.pop_back();
            funn = true;
          }
        }
        if (funn == false) {
          cout << "Fant ingen eksemplarer lånt i " + buffer + " sitt
           navn.\n";
        } else {
            cout << "Innlevering registrert.\n";
        }
    }
}

/**
 * En funksjon som sjekker om en dato er innenfor gyldige intervaller.
 *
 * @param dag - Dagen som sjekkes.
 * @param mnd - Måneden som sjekkes.
 * @param aar - Året som sjekkes.
 *
 * @return true/false - avhengig av om datoen som sjekkes er innenfor gyldig intervall.
 */
bool UtlaansGjenstand::sjekkDato(int dag, int mnd, int aar){
    if ((dag >= 1) && (dag <= 31)
    && (mnd >= 1) && (mnd <= 31)
```

```
    && (aar >= 2022) && (aar <= 2030))
    return true;
    else
    return false;
}

/**
 * Funksjon som skriver ut tittelen, sjangeren, og totalt antall tilgjengelige
 * eksemplarer til utlån for en gitt gjenstand.
 */
void UtlaansGjenstand::skrivData() const {
    cout << "\nTittel: " << tittel << "\n";
    cout << "Sjanger: ";
    switch (sjanger) {
    case Barn:      cout << "Barn";       break;
    case Drama:     cout << "Drama";      break;
    case Fantasy:   cout << "Fantasy";    break;
    case Spenning:  cout << "Spenning";   break;
    }
    cout << "\nTotalt antall eksemplarer: "
       << antallEksemplarer << "\n";
}

/**
 * Funksjon som skriver ut tittelen og totalt antall tilgjengelige eksemplarer
 * til utlån for en gitt gjenstand.
 */
void UtlaansGjenstand::skrivTittelOgAntall() const {
    cout << "Tittel: " << tittel << "\n";
    cout << "Totalt antall eksemplarer: "
         << antallEksemplarer << "\n";
}


//************************  FILM:  *********************************

/**
 * Funksjon som leser inn datamedlemmene til en film-klasse, i tillegg til datamedlemmene
 * til baseklassen.
 *
 * @see UtlaansGjenstand::lesData()
 */
void Film::lesData() {
    UtlaansGjenstand::lesData();
    cout << "Varighet:\n";
    timer             = lesInt("Timer", 0, 10);
    minutter          = lesInt("Minutter", 0, 59);
    aldersGrense      = lesInt("Aldersgrense", 0, 20);
}

/**
 * Funksjon som leser inn brukerens alder og sjekker om denne alderen tilfredsstiller aldersgrensen.
 *
 * @see UtlaansGjenstand::laanUt()
 */
```

```
void Film::sjekkAlderOgEvtLaanUt()  {
    int alder;

    alder = lesInt("Velg din alder", 0, 120);
    if (alder >= aldersGrense) {
    laanUt();
    } else {
    cout << "Beklager, ikke gammel nok!\n";
    }

}

/**
 * Funksjon som skriver ut alle datamedlemmene til et film-objekt, i tillegg
 * til baseklassens datamedlemmer.
 *
 * @see UtlaansGjenstand::skrivData()
 */
void Film::skrivData() const {
    UtlaansGjenstand::skrivData();
    cout << "Varighet: " << timer << "t " << minutter << "m\n"
         << "Aldersgrense: " << aldersGrense;
    }

/**
 * Funksjon som skriver ut data om en film-klasse om en gitt tittel er utleid. I tillegg skriver funksjonen
 * ut datamedlemmene til alle utlånte eksemplarer.
 *
 * @see finnEnGjenstand(...)
 * @see UtlaansGjenstand::skrivData()
 */
void Film::skrivUtleid() const {
    if (utleideEksemplarer.size() > 0) {
        skrivData();

        cout << "\n\nInfo om alle utlån av \"" << tittel << "\":\n";
        for (int i = 0; i < utleideEksemplarer.size(); i++) {
            utleideEksemplarer[i]->skrivData();
        }
    }
}


//*************************  BOK:  *********************************

/**
 * Funksjon som leser inn alle datamedlemmene til en bok-klasse, i tillegg til baseklassens
 * datamedlemmer
 *
 * @see UtlaansGjenstand::lesData()
 */
void Bok::lesData() {
    int valg;

    UtlaansGjenstand::lesData();
```

```
    antallSider = lesInt("Antall sider", 1, 1500);

    valg = lesInt("Velg format:\n1. Pocket\n2. Innbundet\n3. Lydbok\n", 1,
     3);
    switch (valg) {
        case 1: bokFormat = Pocket;     break;
        case 2: bokFormat = Innbundet;  break;
        case 3: bokFormat = Lydbok;     break;
    }
}

/**
 * Funksjon som skriver ut alle datamedlemmene til en bok-klasse, i tillegg til baseklassens
 * datamedlemmer.
 *
 * @see UtlaansGjenstand::skrivData()
 */
void Bok::skrivData() const {
    UtlaansGjenstand::skrivData();
    cout << "Antall sider: " << antallSider << "\n";

    cout << "Format: ";
    switch(bokFormat) {
        case Pocket:    cout << "Pocket\n";    break;
        case Innbundet: cout << "Innbundet\n"; break;
        case Lydbok:    cout << "Lydbok\n";    break;
    }
}


void Bok::skrivUtleid() const {
    if (utleideEksemplarer.size() > 0) {
        skrivData();

        cout << "\n\nInfo om alle utlån av \"" << tittel << "\":\n";
        for (int i = 0; i < utleideEksemplarer.size(); i++) {
            utleideEksemplarer[i]->skrivData();
        }
    }
}


//**********************  FUNKSJONER:  *******************************

/**
 * Funksjon som endrer antall eksemplarer av et gitt utleieobjekt.
 *
 * @see finnEnGjenstand(...)
 * @see UtlaansGjenstand::endreAntallEks()
 */
void endreAntEksemplarer() {
    string tittel;
    UtlaansGjenstand* gjenstand = nullptr;

    cout << "Tittel: "; getline(cin, tittel);
```

```
    gjenstand = finnEnGjenstand(tittel);

    if (gjenstand != nullptr) {
        gjenstand->endreAntallEks();
    } else {
        cout << "Finnes ingen gjenstand med denne tittelen.\n";
    }
}


/**
 *  Returnerer (om mulig) gjenstand med hel/delvis match ift tittel.
 *
 *  @param      tittel -  Tittel på bok/film det søkes etter
 *  @return     Peker til aktuell gjenstand, evt nullptr
 *  @see        UtlaansGjenstand::hentTittel()
 */
UtlaansGjenstand* finnEnGjenstand(string tittel){
  int funn = 0;
  UtlaansGjenstand* gjenstand = nullptr;

  for (const auto & val: gFilmene)
      if (!val->hentTittel().compare(0, tittel.size(), tittel))  {
         gjenstand = dynamic_cast <UtlaansGjenstand*> (val);
         funn++;
      }

  for (const auto & val: gBokene)
      if (!val->hentTittel().compare(0, tittel.size(), tittel))  {
         gjenstand = dynamic_cast <UtlaansGjenstand*> (val);
         funn++;
      }

   return ((funn == 1) ? gjenstand : nullptr);
}

/**
 * Funksjon som fjerner all tidligere allokert minne for filmer og bøker.
 */
void fjernAllokertData() {
    for (int i = 0; i < gFilmene.size(); i++) {
        delete gFilmene[i];
    }
    gFilmene.clear();

    for (int i = 0; i < gBokene.size(); i++) {
        delete gBokene[i];
    }
    gBokene.clear();
}

/**
 * Funksjon som leser inn en tittel og sjekker om denne eksisterer, om det eksisterer
 * et eksemplar av denne vil den bli forsøkt lånt ut.
 *
```

```
 * @see finnEnGjenstand(...)
 * @see UtlaansGjenstand::laanUt()
 * @see Film::sjekkAlderOgEvtLaanUt()
 */
void laanUtGjenstand(){
    string tittel;
    UtlaansGjenstand* gjenstand = nullptr;
    Film* filmen = nullptr;
    bool erFilm = false;

    cout << "Tittel: "; getline(cin, tittel);
    gjenstand = finnEnGjenstand(tittel);

    // Undersøker om det blir forsøkt lånt en film mtp. aldersgrense
    for (int i = 0; i < gFilmene.size(); i++) {
        if (tittel == gFilmene[i]->hentTittel()) {
            erFilm = true;
            filmen = gFilmene[i];
        }
    }

    // Ulike funksjoner kalles for bok og film
    if (gjenstand != nullptr && erFilm == false) {
        gjenstand->laanUt();
    } else if (gjenstand != nullptr && erFilm == true) {
        filmen->sjekkAlderOgEvtLaanUt();
    } else {
        cout << "Finner ingen gjenstand med tittel \"" << tittel << "\"\n";
    }
}

/**
 * Funksjon som leser inn tittel og sjekker om denne tittelen allerede eksisterer.
 * Om ikke spørres brukeren om film eller bok ønskes lagt til, deretter opprettes ønsket
 * objekt og det legges til i relevant vektor.
 *
 * @see finnGjenstand(...)
 * @see Film::lesData()
 * @see Bok::lesData()
 */
void nyGjenstand() {
    string tittel;
    int valg;
    UtlaansGjenstand* gjenstand = nullptr;
    Film* nyFilm = nullptr;
    Bok*  nyBok  = nullptr;

    // Sjekk om tittelen allerede finnes
    cout << "\nUndersøker om tittelen allerede finnes\n\n";
    cout << "Tittel: "; getline(cin, tittel);
    gjenstand = finnEnGjenstand(tittel);

    // Oppretter nytt objekt
    if (gjenstand != nullptr) {
        cout << "Det finnes allerede en gjenstand med denne tittelen!\n";
```

```
    } else {
        valg = lesInt("Legge til (1) Film eller (2) Bok", 1, 2);
        if (valg == 1) {
            nyFilm = new Film(tittel);
            nyFilm->lesData();
            gFilmene.push_back(nyFilm);
        } else if (valg == 2) {
            nyBok = new Bok(tittel);
            nyBok->lesData();
            gBokene.push_back(nyBok);
        }
    }
}

/**
 * Funksjon som leser inn en tittel og undersøker om denne tittelen eksisterer,
 * om den eksisterer registreres et eksemplar som levert tilbake.
 *
 * @see finnEnGjenstand(...)
 * @see leverInn()
 */
void innleverGjenstand(){
    string tittel;
    UtlaansGjenstand* gjenstand = nullptr;

    // Sjekker om tittelen finnes
    cout << "Tittel: "; getline(cin, tittel);
    gjenstand = finnEnGjenstand(tittel);

    if (gjenstand == nullptr) {
        cout << "Finner ikke gjenstanden.\n";
    } else {
        gjenstand->leverInn();
    }

}

/**
 * Funksjon som skriver ut alle filmer- og bøkers tittel og antall eksemplarer.
 *
 * @see UtlaansGjenstand::skrivTittelOgAntall()
 */
void skrivAlle() {
    int antall = 0;

    cout << "\nFilmer:\n";
    for (int i = 0; i < gFilmene.size(); i++) {
        gFilmene[i]->skrivTittelOgAntall();
        antall++;
    }
    if (antall < 1)
        cout << "Det finnes ingen filmer registrert.\n";

    antall = 0;
    cout << "\nBøker:\n";
```

```
    for (int i = 0; i < gBokene.size(); i++) {
        gBokene[i]->skrivTittelOgAntall();
        antall++;
    }
    if (antall < 1)
        cout << "Det finnes ingen bøker registrert.\n";

}


/**
 *   Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny()  {
  cout << "\nFlgende kommandoer er tilgjengelige:\n"
       << "\n   N  - Legg til ny Bok/Film"
       << "\n   U  - Laan ut"
       << "\n   I  - Lever inn"
       << "\n   E  - Endre antall Bok/Film"
       << "\n   A  - Skriv ut alle gjenstander i bibliotek"
       << "\n   S  - Skriv ut alle utleide gjenstander"
       << "\n   Q  - Quit / avslutt";
}

/**
 * Funksjon som skriver ut alle utleide eksemplarer av filmer og bøker.
 *
 * @see Film::skrivUtleid()
 * @see Bok::skrivUtleid()
 */
void skrivUtleide(){
    int antall = 0;
    for (int i = 0; i < gFilmene.size(); i++) {
        gFilmene[i]->skrivUtleid();
        antall++;
    }
    if (antall < 1)
        cout << "Det finnes ingen filmer registrert.\n";

    antall = 0;
    for (int i = 0; i < gBokene.size(); i++) {
        gBokene[i]->skrivUtleid();
        antall++;
    }
    if (antall < 1)
        cout << "Det finnes ingen bøker registrert.\n";
}
```