Exercise 4

Indledning:
Formålet med denne øvelse er at opnå en erfaring med referencer og statiske medlemsvariable.

Igennem denne øvelse skal du gøre dig nogle overvejelser, der har bund i teorien omkring disse emner. Det er meget vigtigt at du gør dig disse overvejelser og forstår hvad det rigtige svar er, og ikke mindst hvorfor svaret er rigtigt. hvis du blot prøver dig frem indtil tingene virker, har du *ikke* løst opgaven – så har du blot skrevet et program, du ikke ved hvorfor det virker!

Exercise 4.1: Klassen `Person`

I denne øvelse skal du implementere klassen `Person`. Et objekt af denne type skal indeholde oplysninger om personens navn og CPR-nummer.

Person
- firstName_: string - middleName_: string - lastName_: string - socialSecNumber_: string - numberOfPersons_: static int
+ setFirstName(const string &): void + setMiddleName(const string &): void + setLastName(const string &): void + getData(string &, string &, string &, string &): void + getNumberOfPersons( ): static int + printPartially( ): void + printAll( ): void + checkForSameName(const Person &): bool

Variablen `getNumberOfPersons` er `static`, og skal til enhver tid indeholde antallet af erklærede instanser af `Person`-klassen.

Ud over de metoder, der er angivet ovenfor skal `Person` have *to* constructors:

```
Person( const string &sSN, const string &fN, const string &lN );
Person( const string &sSN, const string &fN, const string &mN,
    const string &lN );
```

Default-værdierne for default constructoren er

- fornavn: ”N”
- mellemnavn: ”” (tom streng)
- efternavn: ”N”
- personnummer: ”000000-0000”

Den *ene* af de to constructors skal også være default-constructor (dvs. have default argumenter). Det er *kun* den ene der *kan* være det.

a) Diskutér med en medstuderende, hvilken af de to constructors der skal være default-constructor og hvorfor?

   Bemærk: Det er vigtigt, at du ***forstår*** hvorfor kun den ene constructor kan være default constructoren og ikke den anden. Hint: Overvej hvordan du kan erklære objekter, når du har en constructor med default argumenter.

b) Klassen `Person` har også brug for en destructor. Overvej hvorfor det er tilfældet og hvad destructoren skal gøre.

Beskrivelsen af nogle af klassens metoder ses herunder.

```
void printPartially( void );
```

Parametre: ingen
Returværdi: ingen
Beskrivelse: Metoden skal udskrive oplysningerne således:
Navn**:** Peter K. Hansen
Født**:** 290265

```
void printAll( void );
```

Parametre: ingen
Returværdi: ingen
Beskrivelse: Metoden skal udskrive oplysningerne således:
Navn**:** Hansen, Peter Kastrup
Personnr.**:** 290265-0123

```
bool checkForSameName( const Person & );
```

Parametre: Reference til den person der skal sammenlignes med
Returværdi: true hvis de to personer har samme navn
false ellers
Beskrivelse: Metoden skal sammenligne navnene for de to personer

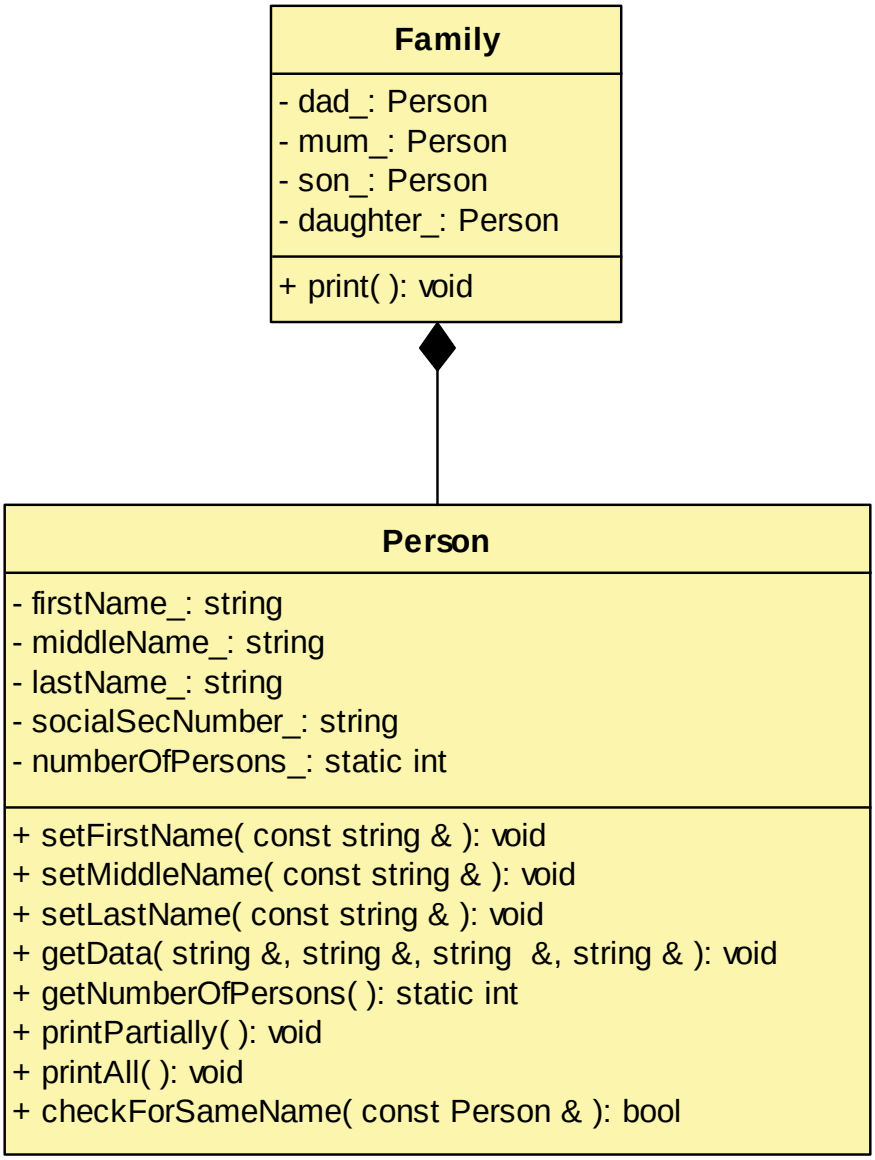
c) Implementér klassen Person. Overvej herunder hvilke metoder der skal erklæres `const` og hvorfor.

d) Sammen med denne øvelse ligger filen test_Person.cpp med et testprogram til klassen Person. Test din klasse med dette program.
Hvis det ikke virker, så er der fejl i ***din*** klasse ☺. Ret den så din klasse virker sammen med testprogrammet **OG** forstå dine fejl.

Exercise 4.2: Klassen `Family`

I denne øvelse skal du implementere klassen `Family`. Klassen gør – som du kan se af klassediagrammet herunder - via komposition brug af klassen `Person`. Som du også kan se af diagrammet har hver familie 4 medlemmer (vi kan ikke gøre det bedre lige nu, men det kommer senere ☺).

a) Implementer klassen `Family`.



Ekstra udfordringer:

Exercise 4.3:

I klassen `Person` er der to constructors. Prøv om du kan slå dem sammen til EN. Der skal en lille snedighed til ☺

Exercise 4.4: (Svær)

I klassen `Family` er der 4 `Person` objekter. Ville det være bedre, at ændre dette til et array med 4 pladser?

Prøv at implementere denne ændring og få det hele til at virke igen.

Hints:
1. Arrayet skal være af typen `Person` og størrelsen skal være en global konstant.
2. Constructoren skal modtage et `const` array. Derfor skal du selvfølgelig også oprette de 4 `Person` objekter i et array i dit testprogram. Et eksempel på det kan du se herunder.

```
Person personArray[4] = { Person(”010108-0001”, ”Peter”, ”Hansen”),
                          Person(”020108-0002”, ”Ib”, ”Hald”, ”Bo” )
                          Person(), Person(”030108-0003”) };
```

Exercise 4.5: (Sværere)

Du skal vide noget om dynamisk hukommelse og om at oprette et array med `new` (læs om det hvis du ikke kender til det i forvejen).

Har alle familier 4 medlemmer? Nej vel ☺

Det ville derfor være meget bedre at have et dynamisk array, således at arrayets størrelse kan være forskellig i forskellige `Family` objekter.

Prøv at implementere denne ændring.

Hints.
1. I stedet for et `Person` array som member i `Family` klassen skal du have en `Person` pointer.
2. Constructoren skal modtage et array og en størrelse på arrayet (antal familiemedlemmer).
3. Arrayet oprettes så i constructoren vha. af `new`.
4. Husk en destructor med `delete`.