Øving 4
TDT4102

Frist: 22.2.2013

Mål for denne øvingen:

- Lære å bruke «Enumerations»
- Lære å bruke Strukturer (structs)
- Lære å implementere og bruke klasser

Generelle krav:

- bruk de eksakte navn og spesifikasjoner som er gitt i oppgaven
- det er valgfritt om du vil bruke en IDE (Visual Studio, Xcode), men koden må være enkel å lese, kompilere og kjøre

Anbefalt lesestoff:

- Kapittel 2.2, 6, 7 & 9.3 Absolute C++ (Walter Savitch)

Merk: Du skal i denne øvingen, ved å bruke «enumerations», strukturer og klasser, definere dine egne typer. Når du gir navn til typene dine, skal du bruke store/små bokstaver lignende når du gir navn til variabler, men med forskjellen at første bokstaven i navnet skal være stor. Dette er kun en konvensjon for at det skal være lettere å lese koden.

Typenavnene dine skal derfor være på formen

```
MyType
```

mens variabler, som du sikkert husker, ser ut som

```
myVariable
```

og til slutt konstanter

```
MYCONSTANT
```

Del 1: Enumeration (15 poeng)

Tabeller er nyttige for å lagre forskjellige typer data, og kan brukes for å lagre lignende data i et sammenhengende minneområde. I den første deloppgaven, skal du lage og bruke en tabell med telefonnumre
I denne delen av øvingen skal du lage en «enumeration» (enum) og bruke den i en «switch»

a) Lag enumeration'en *Beatle* som kan ha de mulige verdiene JOHN, PAUL, GEORGE and RINGO.

   Hint: Verdiene i en enumeration er konstante og skal derfor skrives med store bokstaver

b) Skriv funksjonen *isAlive*() som tar en *Beatle* definert i a) som argument og returnerer en bool som sier om personen er i live. (Paul og Ringo er i live). Bruk en «switch» til å sjekke argumentet mot enumeration'en.

   Hint: En variabel som er av en enumeration type er fremdeles en variabel, og skal derfor begynne med liten bokstav.

Del 2: Strukturer (15 poeng)

I denne delen skal du lage og bruke strukturer.

a) Lag strukturen (struct) *Musician* som har følgende variabler:

   name, en streng (string) med musikerens navn. *weight*, et heltall som gir musikerens vekt. *height*, et heltall som gir musikerens høyde *beardLength*, et heltall som gir musikerens skjegglengde

b) Lag strukturen (struct) *Band* som har følgende variabler:

   name, en streng med gruppas navn. *rockBand*, en bool som sier om gruppe er et rockeband eller ikke *releases*, et heltall som gir antall plater gruppa har sluppet *musicians*, en tabell (array) av «Musicians» (strukturen fra **a)**) som inneholder de fire gruppemedlemmene. Størrelsen på tabellen er fast, siden alle skikkelige «bands» har fire medlemmer.

c) Skriv funksjonen *totalBeardLength*() som tar en gruppe, i form av strukturen definert i b), som argument. Funksjonen skal regne ut den totale skjegglengden til alle medlemmene i gruppa, **men bare hvis gruppa er et rockeband**. Hvis gruppa ikke er et rockeband, et skjegglengden uinteressant for oss, og funksjonen skal returnere -1 Test begge strukturene og funksjonen din fra main().

Del 3: Bruk av string klassen (10 poeng)

I denne deloppgaven skal du bruke en eksisterende klasse, `string`.

Det er viktig å merke seg at det i C++ er to ulike typer «strenger», c-string og standardklassen string. i denne øvingen skal du bruke den sistnevnte, så du må huske å ha med `#include <string>` i koden din og bruke `std` navnerommet (namespace).

Merk: Det er en feil i læreboka om strenger. Den funksjonen som boka kaller *remove*() heter egentlig *erase*()

a) Skriv funksjonen *nameFixer*(). Funksjonen skal ta en streng (string) som argument og returnere en streng Funksjonen skal konvertere et navn på formen FORNAVN ETTERNAVN til ETTERNAVN, FORNAVN.

   Eksempel på korrekt oppførsel:

```
Input: Nikola Tesla
Output: Tesla, Nikola
```

Bruk de forskjellige funksjonene til string-klassen for å implementere dette.

Del 4: Kortklasse (30 poeng)

I denne deloppgaven skal du lage en klasse Card med grunnleggende funksjoner. Denne klassen skal du benytte videre i del 5.

a) Lag klassen *Card*. Denne klassen skal inneholde følgende *private* medlemsvariabler:

value, et heltall som representerer verdien til kortet (1-13). *suit*, en streng som representerer fargen til kortet (S, H, C eller D).

b) Skriv medlemsfunksjonen *getValue*(). Funksjonen skal være *public* og returnere verdien til kortet som et heltall.

c) Skriv medlemsfunksjonen *getSuit*(). Funksjonen skal være *public* og returnere fargen til kortet som en streng (string).

d) Skriv medlemsfunksjonen *setValue*().Funksjonen skal være *public* og ta et heltall som argument. Funksjonen skal sjekke om det gitte heltallet er en gyldig kortverdi (1-13) og oppdatere klassens medlemsvariabel *value* hvis det er gyldig. Funksjonen skal ikke returnere noe.

e) Skriv medlemsfunksjonen *setSuit*(). Funksjonen skal være *public* og ta en streng som argument. Funksjonen skal sjekke om den gitte strengen er en gyldig kortfarge (S, H, C eller D) og oppdatere klassens medlemsvariabel *suit* hvis den er gyldig. Funksjonen skal ikke returnere noe.

f) Skriv medlemsfunksjonen *getFace*(). Funksjonen skal være *public* og returnere fargen og verdien til kortet som en streng.

```
Eksempel på returnert streng: "S5".
```

Å konvertere et heltall til en streng er, dessverre, ikke rett frem. Du kan her bruke en «stringstream» (husk å skrive `#include <sstream>` i koden din) på følgende måte:

```
int number = 42;
string numberAsString;
stringstream ss;
ss << number;
numberAsString = ss.str();
```

Del 5: Kortstokklasse (30 poeng)

I denne deloppgaven skal du implementere klassen CardDeck som bruker klassen Card for å representere en kortstokk. Du skal deretter implementere en enkel funksjonalitet for klassen.

a) Lag klassen *CardDeck*. Klassen skal inneholde følgende *private* medlemsvariabel:

cards, en tabell (array) med 52 Card objekter, som representerer en kortstokk. Variabelen skal være *private*, slik at ingen onde, utenforstående krefter kan endre på kortstokken din.

b) Skriv medlemsfunksjonen *buildDeck*(). Denne funksjonen tar ingen argumenter og returnerer ingenting, men den bygger kortstokken. Funksjonen skal fylle *cards*-tabellen med alle kortene med verdi 1 - 13 i alle 4 farger (S, D, H, C).

c) Skriv medlemsfunksjonen *shuffle*(). Denne funksjonen skal stokke *cards*-tabellen. Den tar ingen argumenter og returnerer ingenting, men den endrer rekkefølgen på kortene i *cards*-tabellen.

Hint: Du trenger ikke lage nye tabeller for å implementere denne funksjonen. Du kan enkelt la to og to kort bytte plass i tabellen.

d) Lag en konstruktør for klassen. Konstruktøren skal kalle *buildDeck*()-funksjonen.

e) Lag en ekstra konstruktør for klassen. Denne konstruktøren skal ta den boolske *shouldShuffle* som argument. Konstruktøren skal kalle *buildDeck*()-funksjonen. Hvis det gitte argumentet *shouldShuffle* er true, skal konstruktøren også kalle *shuffle*()-funksjonen.

f) Skriv medlemsfunksjonen *dealHand*(). Denne funksjonen skal dele ut en hånd med kort hvis det er nok kort gjenstående i stokken.

For å implementere denne funksjonen, må du legge til en ny *privat* medlemsvariabel, *numberDealt*, som er et heltall som representerer hvor mange kort som er delt ut fra stokken allerede. *numberDealt* skal settes til null av funksjonene *buildDeck*() og *shuffle*(). Funksjonen skal ta et heltall som representerer hvor mange kort vi ønsker å dele ut og en tabell (array) av typen *Card* som argument. Kortene som deles ut skal lagres i denne tabellen. Heltallsargumentet skal kontrolleres mot *numberDealt* og antall kort i stokken (52) for å se om det er nok kort igjen i stokken. Hvis det ikke er nok kort igjen i stokken, skal funksjonen ikke gjøre noe. Hvis det er nok kort, skal funksjonen trekke ønsket antall kort fra tabellen og lagre dem i tabellen som er gitt som argument.

g) Test klassene dine fra del 4 og 5 i *main*()-funksjonen. Lag en kortstokk og prøv å trekke gjentatte hender fra stokken og skriv ut disse til skjermen med hjelp av *getFace*()-funksjonen til hvert kort.