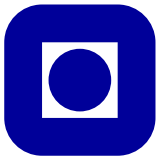

Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2015

Øving 4

Frist: 2015-02-13

Mål for denne øvinga:

- 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 oppgava
- Det er valgfritt om du vil bruke en IDE (Visual Studio, Xcode), men koden må være enkel å lese, kompilere og kjøre.
- Lag et bibliotek for hver klasse, et biblotek består av en header-fil og en implementasjons-fil.

Anbefalt lesestoff:

- Kapittel 2.2, 6, 7 & 9.3 Absolute C++ (Walter Savitch)
- It's Learning notater

1 Enumeration (10%)

I denne oppgava skal du lage og bruke enumerations.

a) Lag enumeration-en `Suit`.

`Suit` kan ha verdiene CLUBS, DIAMONDS, HEARTS, SPADES.

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

b) Lag enumeration-en `Rank`.

`Rank` kan ha verdiene TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING, ACE.

c) Skriv funksjonen `suitToString()`.

`suitToString` tar en `Suit` definert i a) som argument og returnerer en `string` som representerer verdien. 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.`

d) Skriv funksjonen `rankToString()`.

`rankToString` tar inn en `Rank` definert i b) som argument og returnerer en Streng som representerer verdien. Bruk en `switch` til å sjekke argumentet mot enumeration'en.

2 Struktur (10%)

I denne delen skal du lage og bruke strukturer.

a) Lag strukturen (`struct`) `CardStruct`.

Strukturen har følgende variabler:

- `Suit s`, en enumeration av typen `Suit`
- `Rank r`, en enumeration av typen `Rank`

b) Skriv funksjonen `toString()`.

`toString`() tar et kort (CardStruct) i form av strukturen definert i a), som argument. Denne funksjonen skal returnere en `String` som er en tekstlig representasjon av kortet.

For eksempel:

"Ace of Spades"

c) Skriv funksjonen `toStringShort()`.

Du kan i denne oppgava få bruk for følgende funksjon:

```
#include <sstream>
string intToString(int number){
        stringstream ss;
        ss << number;
        return ss.str();
}
```

`toStringShort`() tar et kort (CardStruct) i form av strukturen definert i a) som argument. Denne funksjonen skal returnere en `String` som er en tekstlig representasjon av kortet, til forskjell fra forrige oppvave er vi her ute etter en kort representasjon. Formen som skal skrives ut er <farge (en bokstav)><verdi som tall>. For eksempel "S14" som er et spar ess.

Hint: Du kan hente ut en del av en tekststreng med substr.

Hint: Enumerations er bare heltall konstanter, prøv å skrive ut en Rank variabel.

Test begge strukturene og funksjonene dine fra main().

3 Kortklasse (20%)

I denne deloppgava skal du lage en klasse `Card` med grunnleggende funksjoner. Denne klassen kommer til å bli brukt videre i øvinga.

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

- `Suit s`, en `Suit` enumeration som definert i oppgave 1.
- `Rank r`, en `Rank` enumeration som definert i oppgave 1.
- `bool invalid`, en boolsk variabel som sier hvorvidt klassen (kortet) har blitt satt opp eller ikke.

b) Skriv medlemsfunksjonen `initialize`(Suit s, Rank r).

Funksjonen skal være `public`, ta inn en `Suit` og en `Rank` og ikke returnere noe.

Denne funksjonen setter `Suit s` og `Rank r`. I tillegg settes `invalid` til `usann`.

c) Skriv medlemsfunksjonen `getSuit`().

Funksjonen skal være `public`, tar ikke inn noe men returner `Suit` for kortet.

d) Skriv medlemsfunksjonen `getRank`().

Funksjonen skal være `public`, tar ikke inn noe men returner `Rank` for kortet

e) Skriv medlemsfunksjonen `toString`().

Funksjonen skal være `public` og ikke ta inn noe.

Denne funksjonen returnerer en `Streng` representasjon av kortet. Dersom kortet har `invalid` sann skal "Invalid kort" returneres.

Hint: Kanskje du kan gjenbruke noe kode du har skrevet fra før?

f) Skriv medlemsfunksjonen `toStringShort`().

Funksjonen skal være `public` og ikke ta inn noe.

Denne funksjonen returnerer en kort `Streng` representasjon av kortet. Dersom kortet har `invalid` sann skal "Invalid kort" returneres.

g) Skriv konstruktøren `Card`().

Denne konstruktøren tar ikke inn noe.

Det gjør at vi ikke kan sette opp kortets `Suit` og `Rank` riktig, isteden skal `invalid` settes til `sann`.

h) Skriv konstruktøren `Card`(Suit s, Rank r).

Denne konstruktøren tar inn `Suit` og `Rank`.

Bruk gjerne funksjoner du har laget tidligere for å forenkle denne og sette opp kortet riktig.

4 Kortstokklasse (20%)

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

a) Lag klassen `CardDeck`.

Klassen skal inneholde følgende `private` medlemsvariabler:

- `cards`, en tabell (array) med 52 Card-objekter, som representerer en kortstokk.
- `currentCardIndex`, et heltall som brukes til å holde styr på hvor mange kort som er delt ut.

b) Lag konstruktøren `CardDeck`().

Denne konstruktøren tar ikke inn noe.

Konstruktøren må sørge for at kortstokken blir satt opp riktig, det vil si at hvert kort må settes opp med rett farge og verdi. I tillegg må `currentCardIndex` settes til 0, siden ingen kort har blitt delt ut ennå.

c) Lag medlemsfunksjonen `swap`().

Denne funksjonen tar inn to indekser til `cards` tabellen og bytter om på kortene som finnes på disse to posisjonene. Funksjonen skal være `private`.

d) Lag medlemsfunksjonen `print`().

Denne funksjonen skriver ut kortstokken.

Hvert kort skrives ut med den lange `String` representasjonen.

e) Lag medlemsfunksjonen `printShort`().

Denne funksjonen skriver ut kortstokken.

Hvert kort skrives ut med den korte `String` representasjonen.

f) Lag medlemsfunksjonen `shuffle`().

Denne funksjonen stokker kortstokken, bruk gjerne `swap`() når du implementerer denne.

g) Lag medlemsfunksjonen `drawCard`().

Funksjonen `drawCard`() trekker et vilkårlig kort fra kortstokken. Du må kunne garantere at samme kort ikke kan trekkes flere ganger.

5 BlackJack (40%)

I denne oppgava skal du lage en klasse `BlackJack`. Denne klassen implementerer spillet BlackJack.

a) Lag klassen `BlackJack`.

Klassen skal inneholde følgende `private` medlemsvariabel:

- `deck`, en variabel av typen `CardDeck`.

b) Lag medlemsfunksjonen `getCardValue()`.

Denne tar inn en peker til et kort og returnerer verdien kortet har i BlackJack som et heltall. Ess returneres som -1.

c) Lag medlemsfunksjonen `isAce()`.

Denne tar inn en peker til et kort og returnerer boolsk sann dersom kortet er et ess.

d) Lag medlemsfunksjonen `getPlayerCardValue()`.

Denne tar inn en peker til et kort og returnerer verdien kortet har for spilleren i BlackJack. Dersom kortet er et ess skal spillere selv få velge om kortet regnes som 1 eller 11.

e) Lag medlemsfunksjonen `getDealerCardValue()`.

Denne tar inn en peker til et kort og verdien til dealerens hånd. Funksjonen returnerer verdien til kortet gitt følgende regler:

- Hvis kortet er et ess skal verdien til kortet regnes som 11 med mindre dette gjør at dealerens hånd går over 21. I så fall settes verdien av kortet til 1.
- I alle andre tilfeller returneres kortets verdi som normalt.

f) Lag medlemsfunksjonen `askPlayerDrawCard()`.

Denne funksjonen tar ikke inn noe, men spør spilleren om han / hun ønsker et nytt kort og returner sann dersom spilleren vil ha et nytt kort.

g) Lag medlemsfunksjonen `playGame()`.

Denne funksjonen lar brukeren spille BlackJack mot en dealer. For regler i BlackJack se følgende nettside: http://no.wikipedia.org/wiki/Blackjack