Oppgaver, 16.1.17 – PG4400 C++ progr.

Oppgave 1

a) Lag en klassedeklarasjon (.h-fil) for klassen `PlayerCharacter`. Følgende regler gjelder:
   - Skal legges i en egen headerfil.
   - En `PlayerCharacter` har navn og alder som medlemsdata. Alder kan ikke ha negativ verdi.
   - Klassen skal ha en ikke-parametrisk konstruktør (altså en konstrøktør uten parametere ...) og en parametrisk konstruktør til å initialisere medlemsdata for klassen.
   - Deklarer også en destruktør for klassen. (Siden vi foreløpig ikke gjør bruk av dynamisk minne i `PlayerCharacter` klassen, har destruktøren ikke noen reell oppgave ennå.) Til debuggingsformål skal destruktøren skrive ut spillernavnet for objektet som blir "destruert"/ slettet.
   - Klassen skal ha medlemsfunksjoner for å hente og sette verdier for medlemsdataene. For medlemsfunksjoner som ikke endrer medlemsdata, skal du lage const-funksjoner.
   - Deklarer også en medlemsfunksjon som returnerer dataene i et `PlayerCharacter`-objekt.

b) Lag en implementasjonsfil (.cpp-fil) som definerer klassefunksjonene og som ivaretar regler forbundet med medlemsdata.

c) Lag et testprogram som oppretter to objekter av `PlayerCharacter` klassen ved å bruke både ikke-parametrisk konstruktør og parametrisk konstruktør. `PlayerCharacter` skal *ikke* opprettes vha new, og havner dermed på runtime stack'en. Gjør endringer på medlemsdata til instansene og vis informasjon om dem. Følge med på hva som skjer når programmet er ferdig: objektene slettes og dataminnet gjenkalles.

d) Utvid testprogrammet over ved å deklarere en PlayerCharacter-peker og opprett et PlayerCharacter-objekt i det dynamiske minnet (heap'en) – altså ved bruk av `new`. Gjør kall til medlemsfunksjoner (som over).

e) Sørg for å slette det dynamisk opprettede objektet (med `delete`) før programmet er slutt. (Du vet at det er slettet om teksten fra destruktøren blir skrevet ut.)

Oppgave 2

Skriv klassen `Die` (terning). Lag filene `Die.cpp` og `Die.h`. Den skal inneholde en `init()` funksjon som seeder random generatoren (se forelesning 1), og så skal den ha en funksjon som gir to funksjonsparametere hver et tilfeldig tall i området [1, 6]. Implementer funksjonen to ganger: en gang ved å bruke pass-by-reference og en gang ved å bruke pekere. Funksjonsprototypene skal da se ut som følger:

```
void rollDie(int& die1,  int& die2);
void rollDie(int* die1, int* die2);
```

Med tanke på god objektorientering bør du enten lage en privat hjelpefunksjon som gjør selve terningrullingen og som kalles av funksjonene over, eller implementere terningrullingen direkte i en av funksjonene over, og la den andre funksjonen være en wrapper for denne. (Med andre ord, *ikke* implementer den samme terningrulle logikken i to forskjellige funksjoner i klassen din!)

Oppgave 3 – Craps (spill)

Etter at du har implementert og testet funksjonene, kan du skrive et lite ”craps”-liknende program som lar brukeren vedde på utfallet av terningkastene. Dersom det samlede antall øyne blir 7 eller 11, vinner spilleren det dobbelte av beløpet som ble satset. Spillet pågår inntil spilleren velger å slutte eller går tom for penger. (Se wikipedia.org og søk etter ”craps” for å finne detaljerte regler for spillet dersom du vil utvide reglene).

En kjøring av programmet kan f.eks. vise følgende utskrift:

```
Hvor mye vil du spille med? 1000
Du har 1000 kroner.
Hvor mye satser du? 300
Terning 1 viser 4, terning 2 viser 5
Du tapte...
Vil du spille mer? (j, n) j
Du har 700 kroner.
Hvor mye satser du? 300
Terning 1 viser 4, terning 2 viser 2
Du tapte...
Vil du spille mer? (j, n) j
Du har 400 kroner.
Hvor mye satser du? 200
Terning 1 viser 1, terning 2 viser 2
Du tapte...
Vil du spille mer? (j, n) j
Du har 200 kroner.
Hvor mye satser du? 100
Terning 1 viser 1, terning 2 viser 3
Du tapte...
Vil du spille mer? (j, n) j
Du har 100 kroner.
Hvor mye satser du? 100
Terning 1 viser 2, terning 2 viser 2
Du tapte...
Du har ikke mer penger!
Game over!
Trykk en tast for å fortsette...
```