21 (Black Jack)

Introduktion

- Spelet går ut på att med två eller flera kort försöka uppnå summan 21 eller komma så nära som möjligt
- I detta program kommer det finnas en spelare och en bank som spelar mot varandra
- Spelaren vinner om den får högre summa än banken men bara om summan är högst 21

Version 1.0

- Den första versionen kommer att ha en hel del förenklingar, men vi kommer förbättra den succesivt i senare versioner
- (1) Spelaren får en fråga om den vill ha ett (till) kort
  - Om Ja:
    - Spelaren får ett ”kort” dvs. ett heltal mellan 1-13 slumpas fram och presenteras för spelaren.
    - Summan av alla eventuella tidigare kort presenteras.
    - Om summan understiger 21 upprepas frågan (1).
    - Om summan är lika med 21 räknas bankens resultat ut (se punkt 2)
    - Om summan överstiger 21 räknas resultatet ut (se punkt 2)
  - Om Nej:
    - Bankens resultat räknas ut (se punkt 2)
- (2) När spelaren är nöjd eller har fått 21 så ska bankens resultat räknas ut:
  - Slumpa fram en siffra mellan 1-13
    - Om summan av tidigare siffror är lägre en 17, slumpa fram en till
    - Om summan av tidigare siffror är högre eller lika med 17 blir summa datorns resultat
- (3) Nu är både spelaren och bankens resultat klara
  - Om en av spelarna har fått över 21 förlorar den
  - Om båda spelarna har fått över 21 vinner banken
  - Den som har högst summa vinner
  - Vid lika summa vinner banken

Lösningsförslag: https://gist.github.com/jonatanhallenberg/ebb30eb8a6221a9142df0d9015d98582

Version 1.0 – Screenshot, exempel

```
Vill du ha ett till kort? (Y, N)
Y
Du fick 3. Totalt har du nu 3


Vill du ha ett till kort? (Y, N)
Y
Du fick 9. Totalt har du nu 12


Vill du ha ett till kort? (Y, N)
Y
Du fick 5. Totalt har du nu 17


Vill du ha ett till kort? (Y, N)
N


Du fick 17. Datorn fick 20.
Du förlorade!
```

Version 1.0 – Tips

- För att slumpa fram ett tal mellan 1 och 13
  - Deklara följande före Main:
    - `static Random rnd = new Random();`
  - Varje gång du ska slumpa fram ett nytt tal:
    - `int newCard = rnd.Next(1, 13);`

Version 1.0 – Om du har tid över

- När spelet är slut, fråga om man vill spela igen
  - Om Ja: Starta om spelet
  - Om Nej: Stäng applikationen
- Om du använt Console.ReadLine för att ta in svar på Ja (Y) eller Nej (N), ersätt det med Console.ReadKey
- Om spelaren blir tjock (för över 21) behöver bankens resultat inte räknas ut och presenteras

Version 2.0 - array

- Nu ska vi göra kortleken lite mer verklighetstrogen. När vi slumpar fram ett nr mellan 1-13 kan det ju finnas fler än 4 av varje siffra.
- Skapa en int-array med 52 positioner som innehåller kortlekens alla kort.
- För att dra första kortet, slumpa fram ett heltal mellan 0-51 och ta kortet som finns på det indexet i arrayen.
- Ta sedan bort kortet som finns på den positionen i arrayen så att det inte kan komma en gång till.
- När nästa kort ska dras, slumpa fram ett heltal mellan 0-**50** och ta kortet som finns på det indexet i arrayen.

Lösningsförslag:

https://gist.github.com/jonatanhallenberg/8382e633f6137a75e203ee955f76e0db

Version 3.0 - lista

- I Version 2.0 använde vi en int-array för att hålla reda på kortleken. Nu när vi har lärt oss listor så vill vi ersätta den med en lista.
- Ersätt int-arrayen med en lista som innehåller 52 heltal som motsvarar alla kort i kortleken.
- Anpassa koden som ändrade arrayen till att ändra listan på motsvarande sätt.

Lösningsförslag:

https://gist.github.com/jonatanhallenberg/001eaeebb00cc3e611909aeff87a1b13

Version 4.0 - metoder

- Tidigare har all kod funnits i Main-metoden. Nu ska vi dela upp koden i olika metoder.
- Förslag på metoder:
  - static bool AskYesOrNoQuestion(string question)
    - Man anropar metoden med en fråga t.ex. ”Vill du ha ett nytt kort? (Y eller N)”
    - Metoden returnerar true eller false beroende på om spelaren skrivit in ’y’ eller ’n’
  - static int CalculateBankResult()
    - Räkar ut bankens (datorns) resultat i spelet och returnerar det som int