Dart-spelet

I denna uppgift ska du skapa en dart-simulator och en förenklad version av spelet 501 (fast med 301 poäng).

Uppgiften har fokus på *objektorienterad* programmering.

Notera att vi struntar i att hantera fält för dubbla och trippla poäng vilket skulle göra den svårare versionen alldeles för komplex.

För betyget A och C är det dock viktigt att alla fält och listor är satta som privata. Om man siktar mot C så *kan* man sätta listan som publik men med ”avdrag” och därmed måste all annan kodning vara utmärkt.

För betyget A finns också ett tillägg längre ner.

Följande är spelförloppet:

1. Man börjar med att lägga till alla spelare som ska vara med genom att ange deras namn.
2. Spelarna kastar i tur och ordning. När det är en spelares tur att kasta står hans namn på skärmen.
3. Spelaren kastar 3 pilar som kan ge 0-20 poäng per pil. (max 60p per runda) Varje kast läggs manuellt in i datorn förutom om Dator spelar.
4. Därefter är det nästa spelares tur.
5. Den spelare vars totala summa på alla kast är över 301 poäng först vinner. (Vi bryr oss inte om att man måste gå jämt ut – se dock nedan för *svårare* version.)
6. När en spelare har vunnit ska hans namn visas på skärmen varefter alla hans pil-kast skrivs ut för hur han kastade. Varje serie av 3 pil-kast ska visas så att man förstår hur varje omgång har kastats.
7. Ni kan själva välja om rundan spelas klart när någon uppnått maxpoängen.

Klasser och beskrivning

De klasser som bör ha konstruktor och ToString-metoder ska ha detta.

Följande klasser ska (minst) finnas med:

Klassen Game

Klassen *game* har en lista med player-objekt. Denna klass har också en metod för att lägga till nya spelare (AddPlayer) och har en sträng som in-parameter som sätter namnet på spelaren.

Tillägg: Metoden *AddPlayer* behövs egentligen inte men man kan ha med den för tydlighets skull. Annars kan man skapa nya objekt av klassen spelare direkt i PlayGame-metoden.

I denna metod finns också metoden PlayGame där programmet körs och där man välkomnas till programmet och börjar skriva in spelare.

Se kodexempel på nästa sida:

```
/*
 * Created by SharpDevelop.
 * User: Oscar
 * Date: 2015-05-04
 * Time: 07:08
 *
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */
using System;

namespace dart
{
    class Program
    {
        public static void Main(string[] args)
        {
            //Här skapas ett objekt av klassen Game enbart i syfte att köra igång spelet
            //Denna klass hade också kunnat vara statisk men det är mer förvirrande
            Game My_game = new Game();
            My_game.Playgame();
        }
    }

    class Game
    {
        //inkapslade variabler och lista
        public void Playgame()
        {
            Console.WriteLine("Welcome to the awesome Dart game");
            //Här körs själva spelet
        }

        public void AddPlayer(string name)
        {
            //code
        }
    }
```

Klassen Player

Klassen *player* har en variabel för namn och en lista för *turns*. Här kan du använda en *property* (egenskap) för namnet om du vill kunna nå detta utanför klassen. Eller om du använder ToString-metoden för detta ändamål.

Vidare har denna klass följande metoder:

1. Calculatepoints. Denna metod loopar genom hela listan för objekten *turns* i listan för att få fram den totalla summan poäng som en spelare hittills har fått. Denna metod bör användas för att i slutändan kontrollera om poängen nått upp till 301.
2. Add_turn. Denna metod lägger till en pilomgång i listan som finns i denna klass. Denna metod tar emot tre stycken heltal som motsvarar poängen för respektive pil. Om det är en spelare så får man skriva in poängen medan de slumpas för datorn

Du kommer också troligtvis skapa en metod som heter Print_Turns eller något liknande som skriver ut alla omgångars poäng på ett lämpligt sätt. Denna metod behövs troligtvis om du jobbar med privata listor.

Klassen Turns

Denna klass innehåller inkapslade (privata) variabler/fält och metoder kopplat till en pilomgång. De fält som ska finnas är då tre stycken heltal som motsvarar tre stycken pil-kast.

Denna klass ska ha följande specifika metoder:

1. Get_Score. Denna metod returnerar den sammanlagda poängen för det objektet (det vill säga en pilomgång).

Utöver dessa klasser kan man utveckla programmet ytterligare:

Bygga ut programmet (främst betyg A)

Det går att lägga till en fjärde klass som kan vara statisk om man vill och som symboliserar darttavlan. Det bästa är dock troligtvis att man skapar objekt även av denna klass och att varje spelare ”äger” en egen darttavla.

Denna klass innehåller en vektor på 20 tal där tal lagras medsols och som symboliserar dart-tavlans poäng-markeringar.

Denna vektor börjar på talet 20,1,18 osv.

Denna klass innehåller också en *överlagrad* metod som antingen tar emot ett heltal (spelaren siktar) eller inget tal alls (spelaren kastar på måfå och ett slumptal mellan 1-20 returneras).

I samband med att spelaren kastar sin pil kan spelaren välja att kasta på ”måfå” eller sikta.

I metoden används sannolikhet för att returnera den poäng som man faktiskt får.

Följande är ett *förslag* på poängfördelning om man siktar:

1. Det är 60% sannolikt att du träffar den poäng du siktar mot
2. Det är 15% sannolikt att du träffar poängen som är placerad innan den poäng du siktar mot
3. Det är 15% sannolikt att du träffar poängen som är placerad efter den poäng du siktar mot.
4. Det är 5% sannolikt att du träffar en slumpmässig siffra
5. Det är 5% sannolikt att du helt missar tavlan.

Hanteringen av detta ska ske med egen kod och inga ”hjälp”-metoder som finns för objektet *array.*

Vidare får ni fundera över hur ni vill hantera om spelaren kastar på måfå och om det ska vara helt slumpat som i grunduppgiften eller istället en utarbetad tabell med möjligheter som att missa tavlan helt etc

Spelreglerna kommer också förändras i och med detta då man nu också kan gå *över* 301 poäng. Om man blir ”tjock” så nollställs poängen för den aktuella omgången och turen går över till den andra spelaren. Det vill säga, spelaren kommer stå på samma poäng som han gjorde efter *föregående* omgång.