Ohjelma: NoppaPeli

Valmistelutoimenpiteet

NoppaPeli on peli, johon osallistuu kaksi pelaajaa. Peli muodostuu heittokierroksista, joissa kumpikin pelaajista heittää kahta noppaa. Noppien tulokset lasketaan yhteen. Pelikierroksen pistelasku toimii seuraavasti:

- jos toinen pelaajista saa isomman yhteislukeman, hänen pistesaldoaan kasvatetaan yhdellä ja samalla toisen pelaajan pistesaldo nollataan.
- jos molemmat saavat saman tuloksen, ei pistesaldoja muuteta.

Pelin voittaa se pelaaja, joka saa ensin kolme pistettä.

Tehtävä

Toteuta komentokehoteohjelma nimeltä Tentti02. Lisää projektiin tiedoston Program.cs lisäksi seuraavat tiedostot:

- INimi.cs (rajapinta INimi),
- Noppa.cs (luokka Noppa),
- Pelaaja.cs (luokka Pelaaja),
- Peli.cs (luokka Peli) ja
- Sovellus.cs (staattinen luokka Sovellus).

INimi-rajapinta sisältää pakotteen merkkijonotyyppisestä ominaisuudesta Nimi, jolla on vähintään get-metodi.

Noppa-luokka sisältää seuraavat elementit:

- staattinen muuttuja, jonka arvoksi sijoitetaan System.Random-olio.
- automaattiset ominaisuudet Lukema ja HeittoLkm.
- konstruktori, joka alustaa heittojen lukumäärän (ominaisuus HeittoLkm) nollaksi.
- metodi Heita, joka arpoo kokonaisluvun väliltä 1-6 staattisen muuttujan sisältämän System.Random-olion avulla ja sijoittaa arvotun luvun ominaisuuteen Lukema samalla kasvattaen heittojen lukumäärää yhdellä. Metodi palauttaa ominaisuuden Lukema arvon.

Pelaaja-luokka toteuttaa rajapinnan INimi sisältäen seuraavat elementit:

- automaattiset ominaisuudet Nimi, Pisteet ja Noppa (- ja ominaisuuksien tyypit ovat string, int ja Noppa).
- konstruktori, jolla on parametrimuuttuja pelaajan nimen vastaanottamiseen. Toimintana asetetaan pelaajalle nimi (parametrimuuttujasta), nollataan pelaajan pistemäärä ja luodaan pelaajalle noppa (eli asetetaan ominaisuuteen Noppa arvoksi uusi Noppa-olio).

Peli-luokka sisältää seuraavat elementit:

- Pelaaja-tyyppiset automaattiset ominaisuudet Pelaaja1 ja Pelaaja2
- int-tyyppiset automaattiset ominaisuudet KierrosLkm ja VoitonPisteRaja, joista jälkimmäisessä on vain get-metodi.

- Pelaaja-tyyppinen ominaisuus Voittaja, jossa vain get-metodi ja sen toimintana palautetaan jokin seuraavista arvoista:
    - ominaisuuden Pelaaja1 arvo, jos ko. ominaisuuden Pelaaja-olion pistemäärä (ominaisuus Pisteet) on vähintään voittamiseen tarvittava pistemäärä
    - ominaisuuden Pelaaja2 arvo, jos ko. ominaisuuden Pelaaja-olion pistemäärä (ominaisuus Pisteet) on vähintään voittamiseen tarvittava pistemäärä
    - null, jos kumpikaan edellisistä vaihtoehdoista ei toteudu
- konstruktori, jolla on parametrimuuttujat kahden Pelaaja-olion ja voittamiseen tarvittavan pisterajan vastaanottamiseen. Toimintana asetetaan parametrien arvot vastaaviin ominaisuuksiin Pelaaja1, Pelaaja2 ja VoitonPisteRaja sekä nollataan ominaisuuden KierrosLkm arvo.
- yksityinen (private) metodi NoppienHeitto, jolla parametrimuuttuja Pelaaja-olion vastaanottamiseen. Toimintana parametrissa saadun Pelaaja-olion sisältämää Noppa-oliota heitetään kahdesti ja tulostetaan komentokehotteeseen pelaajan heittotulokset (noppien silmäluvut ja yhteenlaskettu luku – ks. malli ohjelman suoritusesimerkistä). Metodi palauttaa arvonaan heittojen yhteenlasketun luvun.
- metodi PelaaKierros, jolla ei ole parametreja ja joka tulostaa tekstin ”Heittokierros X”, missä X on kierroksen numero. Sitten suoritetaan nopan heitot kutsumalla metodia NoppienHeitto antamalla argumentiksi ensin Pelaaja1:n ja sitten Pelaaja2:n arvona olevan Pelaaja-olion. Saatujen tulosten avulla tutkitaan, kumman pelaajan pistemäärää kasvatetaan ja kumman nollataan vai tehdäänkö kumpaakaan (= sama tulos).

Sovellus-luokka on staattinen eli siinä voi olla vain staattisia elementtejä. Luokka sisältää vain yhden metodin:

- staattinen metodi Aja, joka toimii varsinaisena pelin algoritmina. Metodi
    - tulostaa keltaisella värillä tekstin ”Noppa-peli”
    - kysyy käyttäjältä pelaajien nimet ja perustaa niiden avulla Pelaaja-oliot
    - perustaa Peli-olion, johon pelaajiksi edellä tehdyt Pelaaja-oliot ja voittamisen pisterajaksi luku 3.
    - perustaa Noppa-oliot pelaamista varten
    - etenee peliä heittokierros kerrallaan kutsuen Peli-olion metodia PelaaKierros, kunnes pelissä on olemassa voittaja (eli Peli-olion ominaisuudessa Voittaja on muu arvo kuin null).
    - tulostaa keltaisella värillä voittajan nimen ja pelattujen heittokierrosten määrän (ks. malli ohjelman suoritusesimerkistä).

Program-luokka sisältää Main-metodin, jossa vikasietoisesti kutsutaan staattista metodia Sovellus.Aja. Jos sen suorituksessa nostetaan poikkeus, käsitellään se tulostamalla tekstirivit (- alemmassa rivissä tulostuu virheolion sisältämä virheteksti):

```
Ohjelman suoritus päättyi virheeseen.
Virhe: <TÄSSÄ POIKKEUSOLION VIRHETEKSTI>
```

Esimerkkejä (3 kpl) ohjelman suorittamisesta:

```
Noppa-peli
Pelaajan 1 nimi: Matti
Pelaajan 2 nimi: Maija
Heittokierros 1
Matti: 6 + 6 = 12
Maija: 3 + 1 = 4
Heittokierros 2
Matti: 3 + 3 = 6
Maija: 3 + 4 = 7
Heittokierros 3
Matti: 3 + 2 = 5
Maija: 3 + 3 = 6
Heittokierros 4
Matti: 2 + 2 = 4
Maija: 5 + 4 = 9

Pelin voittaja on Maija ja noppia heitettiin 4 kertaa!
```

```
Noppa-peli
Pelaajan 1 nimi: Matti
Pelaajan 2 nimi: Maija
Heittokierros 1
Matti: 3 + 6 = 9
Maija: 2 + 5 = 7
Heittokierros 2
Matti: 3 + 6 = 9
Maija: 1 + 5 = 6
Heittokierros 3
Matti: 6 + 4 = 10
Maija: 4 + 1 = 5

Pelin voittaja on Matti ja noppia heitettiin 3 kertaa!
```

```
Noppa-peli
Pelaajan 1 nimi: Matti
Pelaajan 2 nimi: Maija
Heittokierros 1
Matti: 2 + 5 = 7
Maija: 4 + 4 = 8
Heittokierros 2
Matti: 5 + 1 = 6
Maija: 3 + 3 = 6
Heittokierros 3
Matti: 3 + 3 = 6
Maija: 4 + 1 = 5
Heittokierros 4
Matti: 4 + 2 = 6
Maija: 6 + 3 = 9
Heittokierros 5
Matti: 4 + 4 = 8
Maija: 4 + 3 = 7
Heittokierros 6
Matti: 2 + 5 = 7
Maija: 4 + 1 = 5
Heittokierros 7
Matti: 5 + 2 = 7
Maija: 1 + 4 = 5

Pelin voittaja on Matti ja noppia heitettiin 7 kertaa!
```

VIHJEITÄ TEKEMISEEN:

- Projektiin tarvittavat tiedostot saat Visual Studiossa lisättyä joko Solution Explorer -ikkunassa projektin nimen päällä pikavalikosta Add / New Item… tai valikkoriviltä Project / Add New Item….
- Huomaa, että vaikka kaikki tehtävässä tarvittava voidaan kirjoittaa yhteen tiedostoon, tulee tässä tehtävässä tehdä erilliset tiedostot.
- Huom. Voit myös käyttää toteutuksessa ominaisuuksien asemesta aksessorimetodeita, mutta se vaikuttaa arvioinnissa alentavasti (= toteutus ei ole tällöin täysin tehtävän annon mukainen).