Ohjelma: Kilpailu

Valmistelutoimenpiteet

Kilpailuun liittyy osallistujia, joista kukin suorittaa kilpailusuorituksen. Kilpailusuoritus voi olla lajista riippuen pistemäärä, käytetty aika tms. Kilpailun osallistuja voi olla henkilö, eläin, kone, joukkue tms.

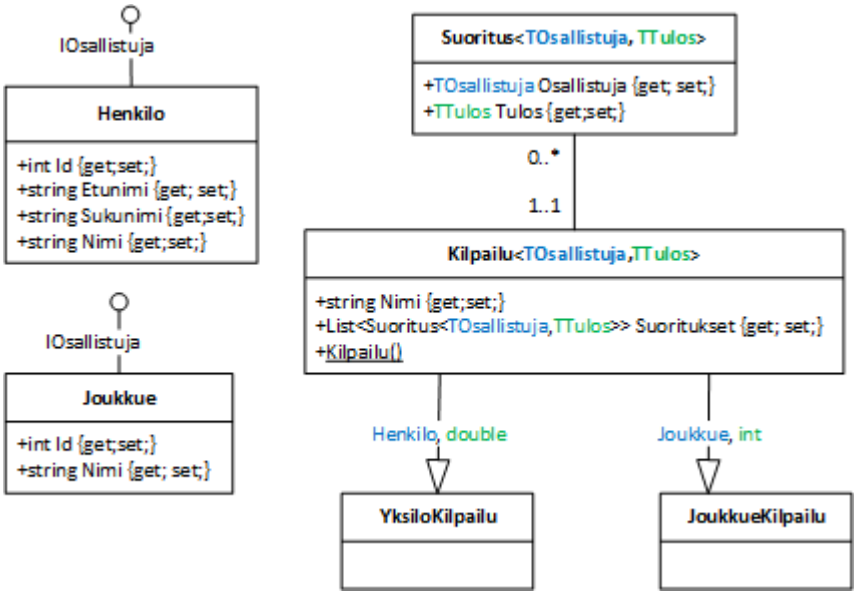
Tehtävä

Toteuta komentokehoteohjelma nimeltä Tentti03.

Lisää projektiin tiedoston Program.cs lisäksi seuraavat tiedostot:

- IOsallistuja.cs (rajapinta IOsallistuja),
- Henkilo.cs (luokka Henkilo),
- Joukkue.cs (luokka Joukkue),
- Suoritus.cs (luokka Suoritus),
- Kilpailu.cs (luokka Kilpailu),
- JoukkueKilpailu.cs (luokka JoukkueKilpailu),
- YksiloKilpailu.cs (luokka YksiloKilpailu) ja
- Sovellus.cs (luokka Sovellus).

Alla on kuva toteuttavista luokista (lukuun ottamatta Sovellus- ja Program-luokkaa) UML-luokkakaaviona [-tästä voi olla apua kokonaisuuden hahmottamiseen, joskaan ei ole välttämätön tehtävän tekemisen kannalta]:



IOsallistuja-rajapinta sisältää pakotteen merkkijonotyyppisestä ominaisuudesta Nimi, jolla on vähintään get-metodi ja kokonaislukutyyppisestä ominaisuudesta Id, jolla on vähintään get-metodi.

Henkilo-luokka toteuttaa rajapinnan IOsallistuja sisältäen seuraavat elementit:

- automaattiset ominaisuudet Id, Etunimi ja Sukunimi
- ominaisuus Nimi, jonka get-metodi palauttaa ominaisuuksien Sukunimi ja Etunimi arvot välilyönnillä eroteltuna ja jonka set-metodissa sijoitetaan ominaisuuteen sijoitettavasta arvosta

(value) saatavat merkkijonot ominaisuuksiin Sukunimi ja Etunimi [ks. vihje]. Jos sijoitettava arvo ei ole muodoltaan oikea (eli ei sisällä kahta välilyönnillä eroteltua merkkijonoa), heitetään poikkeus virhetekstillä ”Henkilön nimi on oltava muodossa sukunimi etunimi.”.

Joukkue-luokka toteuttaa rajapinnan IOsallistuja sisältäen seuraavat elementit:

- automaattiset ominaisuudet Id ja Nimi

Suoritus-luokka on geneerinen sisältäen kaksi tyyppiparametria, joista ensiksi mainittu määrittää osallistujan tyypin ja jälkimmäinen osallistujan tuloksen tyypin. Luokka sisältää seuraavat elementit:

- automaattiset ominaisuudet Osallistuja ja Tulos, jotka tyypitetään em. tyyppiparametreilla

Kilpailu-luokka on geneerinen sisältäen kaksi tyyppiparametria, joita käytetään Suoritus-luokan tyypittämiseen (eli ovat vastaavat kuin Suoritus-luokassa). Luokka sisältää seuraavat elementit:

- automaattiset ominaisuudet Nimi ja Suoritukset, joista jälkimmäinen tyypitetään luokan tyyppiparametrein määritellyn Suoritus-tyyppisten olioiden listaksi.
    - Esimerkiksi jos käytetään tyyppiparametrien niminä TOsallistuja ja TTulos, määritellään tällöin ominaisuuden Suoritukset tyypiksi `List<Suoritus<TOsallistuja, TTulos>>`.
- konstruktori, jossa ominaisuuden Suoritukset arvoksi sijoitetaan uusi listaolio.
    - Esimerkiksi jos käytetään tyyppiparametrien niminä TOsallistuja ja TTulos, tehdään listaolio seuraavasti: `new List<Suoritus<TOsallistuja, TTulos>>()`.

JoukkueKilpailu-luokka periytetään luokasta Kilpailu käyttäen tyyppiparametreille arvoja Joukkue ja int. Luokkaan ei määritellä tässä mitään lisäelementtejä.

YksilöKilpailu-luokka periytetään luokasta Kilpailu käyttäen tyyppiparametreille arvoja Henkilo ja double. Luokkaan ei määritellä tässä mitään lisäelementtejä.

Sovellus-luokka on staattinen ja se sisältää seuraavat elementit:

- staattisen System.Random-tyyppinen muuttuja, johon sijoitetaan ko. tyypin olio. Tätä oliota käytetään satunnaislukujen tuottamiseen.
- staattinen metodi TeeYksiloKilpailu, jolla on kaksi double-tyyppistä parametria minimi- ja maksimipistemäärän välittämiseen ja jonka toiminta on seuraava
    - kysytään yksilökilpailun nimi ja tehdään sitä käyttäen uusi YksiloKilpailu-olio.
    - kysytään toistuvasti osallistujan nimeä (ks. esimerkkisuoritus – tyhjä merkkijono lopettaa syöttämisen). Jokaista osallistujan nimeä käyttäen tehdään seuraava:
        - lisätään YksiloKilpailu-olion Suoritukset-ominaisuuteen uusi Suoritus-olio (tyyppiä Suoritus<Henkilo,double>), jonka Osallistuja-ominaisuuteen luodaan edellä saadun nimen avulla Henkilo-olio ja Tulos-ominaisuuteen arvotaan desimaaliluku väliltä minimitulos – maksimitulos (ks. vihje).
    - tulostetaan keltaisella värillä edellä lisättyjen (ja arvottujen) suoritusten tulokset (ks. esimerkkisuoritus). Tulokset on pyöristetty kahteen desimaaliin. Osallistujien tulostusjärjestyksellä ei ole tässä väliä.
- staattinen metodi TeeJoukkueKilpailu, jolla on int-tyyppinen parametri maksimipistemäärän välittämiseen ja jonka toiminta on seuraava:

    - kysytään joukkuekilpailun nimi, jota käyttäen tehdään JoukkueKilpailu-olio.
    - kysytään toistuvasti osallistujan nimeä (ks. esimerkkisuoritus – tyhjä merkkijono lopettaa). Lisätään aina JoukkueKilpailu-olion Suoritukset-ominaisuuteen Suoritus-olio (tyyppiä Suoritus<Joukkue, int>), jonka Osallistuja-ominaisuuteen luodaan edellä saadun nimen avulla Joukkue-olio ja Tulos-ominaisuuteen arvotaan kokonaisluku väliltä 0 – maksimipistemäärä.
    - tulostetaan keltaisella värillä kaikkien kilpailuun osallistujien nimet ja tulospistemäärät (ks. esimerkkisuoritus). Osallistujien tulostusjärjestyksellä ei ole tässä väliä.
- staattiset metodit Desimaaliluku, DesimaalilukuPakottaen, Kokonaisluku ja KokonaislukuPakottaen, joiden koodi on annettu valmiiksi tehtävän lopussa.
- staattinen metodi Aja (- muista varustaa public-määreellä), jonka toiminta on seuraava:
    - kysytään käyttäjältä toistuvasti, haluaako tämä tehdä yksilö- vai joukkuekilpailun (ks. esimerkkisuoritus).
    - Jos tehdään yksilökilpailu, pyydetään yksilökilpailun minimi- ja maksimitulokset sekä kutsutaan metodia TeeYksiloKilpailu.
    - Jos tehdään joukkuekilpailu, pyydetään joukkuekilpailun maksimitulos sekä kutsutaan metodia TeeJoukkueKilpailu.

Program-luokka sisältää Main-metodin, jossa vikasietoisesti kutsutaan staattista metodia Sovellus.Aja(). Jos sen suorituksessa nostetaan poikkeus, käsitellään se tulostamalla tekstirivit

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

Esimerkkisuoritus (- huomaa, että tulokset on saatu arpomalla, joten eri suorituskerroilla tulee erilaiset tulokset):

```
Tehdäänkö yksilö- (y) vai joukkuekilpailu (j), tyhjä lopettaa: y
Anna yksilökilpailun minimitulos:  7,9
Anna yksilökilpailun maksimitulos:  21,5
Anna yksilökilpailun nimi: Juoksu 100 m
Anna osallistujan nimi muodossa "sukunimi etunimi" (tyhjä lopettaa): Mainio Matti
Anna osallistujan nimi muodossa "sukunimi etunimi" (tyhjä lopettaa): Juonio Jussi
Anna osallistujan nimi muodossa "sukunimi etunimi" (tyhjä lopettaa): Pulla Pauli
Anna osallistujan nimi muodossa "sukunimi etunimi" (tyhjä lopettaa):

Kilpailun Juoksu 100 m tulokset:
Mainio Matti, 11,84
Juonio Jussi, 14,11
Pulla Pauli, 10,85

Tehdäänkö yksilö- (y) vai joukkuekilpailu (j), tyhjä lopettaa: j
Anna joukkuekilpailun maksimipistemäärä:  120
Anna joukkuekilpailun nimi: SAMK-turnaus
Anna osallistujan nimi (tyhjä lopettaa): TK19
Anna osallistujan nimi (tyhjä lopettaa): TK20
Anna osallistujan nimi (tyhjä lopettaa): AE20
Anna osallistujan nimi (tyhjä lopettaa):

Kilpailun SAMK-turnaus tulokset:
TK19, 35 pistettä
TK20, 20 pistettä
AE20, 29 pistettä

Tehdäänkö yksilö- (y) vai joukkuekilpailu (j), tyhjä lopettaa:
```

VIHJEITÄ TEKEMISEEN:

- Merkkijonon jakaminen osamerkkijonoihin (merkkijonotaulukoksi) tehdään metodilla Split. Saadusta merkkijonotaulukosta voidaan viitata alkioihin indeksinumeroilla 0, 1, … (- jokainen alkio on yksi alkuperäisen merkkijonon osamerkkijonoista).
- Random-olion metodilla NextDouble saadaan satunnainen desimaaliluku väliltä 0-1. Tätä voi käyttää satunnaisen desimaaliluvun tuottamiseen halutulle välille. Esimerkiksi väliltä 5.0 – 8.0 saadaan satunnaisluku seuraavasti: 5.0 + NextDouble() * (8.0 – 5.0).

KOPIOITAVAA KOODIA:

Luokan Sovellus staattiset metodit Desimaaliluku, DesimaalilukuPakottaen, Kokonaisluku ja KokonaislukuPakottaen (- huomaa kopioidessasi miinusmerkkien ongelmallisuus -> kirjoita Visual Studiossa koodin miinusmerkit uudelleen):

```
static double Desimaaliluku(string kehote, int tarkkuus = -1)
{
    double paluu;
    Write($"{kehote} ");
    if (!double.TryParse(ReadLine(), out paluu))
    {
        throw new ApplicationException("Syöte ei ole kelvollinen desimaaliluku.");
    }

    return tarkkuus >= 0 ? Math.Round(paluu, tarkkuus) : paluu;
}
static double DesimaalilukuPakottaen(string kehote, int tarkkuus = -1)
{
    do
    {
        try
        {
            return Desimaaliluku(kehote, tarkkuus);
        }
        catch (Exception e)
        {
            WriteLine(e.Message);
        }
    } while (true);
}
static int Kokonaisluku(string kehote, int min = int.MinValue, int max = int.MaxValue)
{
    int paluu;
    Write($"{kehote} ");
    if (!int.TryParse(ReadLine(), out paluu) || (paluu < min || paluu > max))
    {
        throw new ApplicationException("Syöte ei ole kelvollinen kokonaisluku." +
            (min > int.MinValue ? ($" Minimi on {min}.") : "") +
            (max < int.MaxValue ? ($" Maksimi on {max}.") : ""));
    }

    return paluu;
}
static int KokonaislukuPakottaen(string kehote, int min = int.MinValue, int max =
int.MaxValue)
{
    do
    {
        try
        {
            return Kokonaisluku(kehote, min, max);
        }
        catch (Exception e)
        {
            WriteLine(e.Message);
        }
    } while (true);
}
```