Java SE Feladatsor 1
Ciklusok, Tombok, OOP alapok

Szabo Daniel
daniel.szabo99@outlook.com

2021. június 9.

Kivonat

Ez a feladatsor egy sorozat elso resze, amelyben Java programozasi nyelven a kovetkezo temakhoz fuzodo feladatsorok talalhatok: ciklusok, tombok es objektumorientalt programozas alapok. A feladatsor harom allomanybol tevodik ossze: a feladatok leirasa (ez a dokumentum), egy kodvazlat amiben feladatokat kell kidolgozni illetve egysegtesztek. Aki a feladatsor minden feladatat sikeresen teljesiteti, az magabiztosan tudja kezelni a tomboket es ciklusokat egyutt es kulon, azokkal hasznos muveleteket tud vegezni onalloan, fuggvenyeket tud definialni es az objektumorientalt programozasrol alapszintu megertese van.

Tartalomjegyzék

1. Tesztek Futtatasa

Ez a feladatsor tesztekkel lett kiegeszitve, ami lehetove teszi szamodra, hogy azonnal visszajelzest kapj, hogy megfeleloen mukodik-e egy bizonyos feladatra irt megoldasod, illetve a megoldas ertekelojenek is segit egy pillanat alatt ellenorizni, hogy mukodokepes megoldast adtal be. A tesztelesi keretrendszer mukodesi elve miatt a feladatoknak muszaj az OOP elveit hasznalnia, a metodusoknak parametereket befogadnia es visszateresi ertekeket adnia. Erdemes a teszteket minden valtoztatas utan lefuttatni, hogy folyamatosan lasd, esetleg elrontottal-e valamit egy valtoztatasoddal, ami korabban mukodott.

Segitseg: Tesztek futtatasa IntelliJ IDEA-ban, Tesztek futtatasa Eclipse-ben

2. Szamologep

Bevezetes Ebben a feladatban a `Szamologep` osztaly fuggvenyeit fogjuk implementalni. Az egyszeruseg kedveert ebben az osztalyban csak `float` adattipust hasznalunk szamitasokhoz. A feladat 5 reszbol all, minden reszben egy metodus hianyzo logikajat kell potolnod. Egy pelda metodus `osszead()` segitsegkent el lett helyezve a forraskodban.

```
1 public float osszead(float a, float b){
2     return a + b;
3 }
```

Kodreszlet 1. Pelda Metodus

2.1. kivon(a,b)

A `kivon` fuggveny a ket parameterkent kapott szam kulonbseget kell visszaadja, peldaul:

`kivon(4,3)` → `1`

`kivon(-2.5,3)` → `-5.5`

`kivon(0,2.25)` → `-2.25`

2.2. szoroz(a,b)

A `kivon` fuggveny a ket parameterkent kapott szam szorzatat kell visszaadja, peldaul:

`szoroz(4,3)` → `12`

`szoroz(-2.5,3)` → `-7.5`

`szoroz(0,2.25)` → `0`

2.3. oszt(a,b)

Az `oszt` fuggveny a ket parameterkent kapott szam hanyadosat kell visszaadja, peldaul:

`oszt(4, 2)` → `2`

`oszt(-4, 8)` → `-0.5`

`oszt(3, 0)` → `hiba(-∞)`

2.4. hatvany(a,b)

A `hatvany` fuggveny a ket parameterkent kapott szam hatvanyat kell visszaadja, peldaul:

`hatvany(2,3)` → `8`

`hatvany(2,-2)` → `0.25`

`hatvany(0,2.25)` → `0`

2.5. negyzetgyok(a)

A `negyzetgyok` fuggveny a ket parameterkent kapott szam kulonbseget kell visszaadja, peldaul:

`negyzetgyok(4)` → `2`

`negyzetgyok(1)` → `1`

`negyzetgyok(-2)` → `hiba (-1)`

3. Ciklusok

Ebben a feladatban a `Ciklusok` osztaly fuggvenyeit fogjuk implementalni, aminek soran ciklusos algoritmusokat gyakorlunk. Az egyszeruseg kedveert ebben az osztalyban csak `int` adattipust hasznalunk szamitasokhoz. A feladat 4 reszbol all, minden reszben egy metodus hianyzo logikajat kell potolnod. Egy pelda metodus `szamOsszeg10ig()` segitsegkent el lett helyezve a forraskodban. A feladatok megoldasahoz javasolt a `for(int i = 0; i < n; i++){...}` struktura hasznalata, de mas megoldas is megengedett (`while(condition)`, `for(Object o: objects){...}`)

Segitseg: docs.oracle.com - The for statement.

```
1 public int szamOsszeg10ig(){
2     int eredmeny = 0;
3
4     for (int i = 1; i <= 10; i++) {
5         eredmeny = eredmeny + i;
6     }
7
8     return eredmeny;
9 }
```

Kodreszlet 2. Pelda Metodus

3.1. szamOsszeg(n)

A `szamOsszeg` 0 es n kozott (inkluziv) levo osszes egesz szam osszeget adja vissza eredmenyul, peldaul:

`szamOsszeg(-3)` → `-6`

`szamOsszeg(0)` → `0`

`szamOsszeg(4)` → `10`

3.2. legnagyobbPrim(n)

A `legnagyobbPrim` 0 es n kozott (inkluziv) a legnagyobb primszamot adja eredmenyul, peldaul:

`legnagyobbPrim(0)` → `hiba(-1)`

`legnagyobbPrim(2)` → `2`

`legnagyobbPrim(10)` → `7`

3.3. fibonacci(n)

Az `fibonacci` a fibonacci sorozat n-edig elemet adja vissza
(`[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...]`), ahol `fibonacci(0) = 0`, peldaul:
`fibonacci(-1)` → `hiba (-1)`
`fibonacci(1)` → `1`
`fibonacci(7)` → `13`

3.4. legnagyobbKozosOszto(a,b)

A `legnagyobbKozosOszto` fuggveny a ket parameterkent kapott szam legnagyobb kozos osztojat kell visszaadja, peldaul:
`legnagyobbKozosOszto(0, 4)` → `hiba(-1)`
`legnagyobbKozosOszto(3, 7)` → `1`
`legnagyobbKozosOszto(54, 24)` → `6`

Segitseg: Wikipedia - Legnagyobb Kozos Oszto.

4. Tombok

Ebben a feladatban a `Tombok` osztaly fuggvenyeit fogjuk implementalni, aminek soran tombok hasznalatat gyakoroljuk. Az egyszeruseg kedveert ebben az osztalyban csak `int` tipusu tomboket hasznalunk hasznalunk szamitasokhoz, egyedul a 10. feladat hasznal String tipust. A feladat 10 reszbol all, minden reszben egy metodus hianyzo logikajat kell potolnod. Egy pelda metodus `szum()` segitsegkent el lett helyezve a forraskodban. A feladatok megoldasahoz javasolt a `for(int i = 0; i < n; i++){...}` struktura hasznalata, de mas megoldas is megengedett (`while(condition)`, `for(Object o: objects){...}`)

Segitseg: docs.oracle.com - Arrays.

```
1 public int szum(int[] tomb) {
2     int eredmeny = 0;
3
4     for (int i = 1; i <= 10; i++) {
5         eredmeny = eredmeny + tomb[i];
6     }
7
8     return eredmeny;
9 }
```

Kodreszlet 3. Pelda Metodus

4.1. uj10HosszuIntTomb()

A `uj10HosszuIntTomb` egy 10 hosszusagu `int` tombot ad vissza. A tomb minden eleme 0 legyen.

4.2. ujIntTombMegadottSzamokkal(a, b, c, d)

A `ujIntTombMegadottSzamokkal` 4 megadott szammal tolt fel egy 4 hosszusagu tombot, peladul:

`ujIntTombMegadottSzamokkal(2, 3, 8, 3)` → `[2, 3, 8, 3]`

4.3. tombHossz(t)

Az `tombHossz` a parameterkent kapott tomb hosszat adja vissza eredmenyul, peldaul:

`tombHossz([])` → `0`

`tombHossz([9, 23])` → `2`

4.4. legkisebb(t)

A `legnagyobb` fuggveny a ket parameterkent kapott tomb legkisebb erteket adja vissza, peldaul:

`legnagyobb([]) → hiba(-1)`

`legnagyobb([3, 4, 1, 6]) → 1`

4.5. legnagyobb(t)

A `legnagyobb` fuggveny a ket parameterkent kapott tomb legnagyobb erteket adja vissza, peldaul:

`legnagyobb([]) → hiba(-1)`

`legnagyobb([3, 4, 1, 6]) → 6`

4.6. atlag(t)

A `legnagyobb` fuggveny a ket parameterkent kapott tomb ertekeinek atlagat adja vissza `int`-kent, peldaul:

`legnagyobb([]) → 0`

`legnagyobb([2, 0, 5, 7]) → 3`

4.7. keres(t,k)

A `keres` fuggveny a parameterkent kapott tombben a parameterkent kapott int erteket keresi meg, es ha megtalalja akkor a szam tombon beli indexet adja vissza eredmenyul, peldaul:

`keres([1, 3, 6], 3) → 1`

`keres([1, 3, 6], 2) → hiba(-1)`

4.8. szamol(t, k)

A `szamol` fuggveny a parameterkent kapott tombben a parameterkent kapott int erteket keresi meg, es megszamolja, hanyszor fordul elo benne, majd ezt a szamot adja vissza eredmenyul, peldaul:

`szamol([1, 3, 3], 1) → 1`

`szamol([1, 3, 3], 3) → 2`

`szamol([1, 3, 3], 2) → 0`

4.9. tukroz(t)

A `tukroz` a parameterkent kapott tombot atrendezi ugy, hogy az elemek eredeti sorrendjet megforditja. Fontos, hogy ez a fuggveny nem ter vissza eredmennyel, hanem magat a parameterkent kapott tombot modositja, peldaul:

`tukroz([])` → `[]`

`tukroz([1, 2, 3])` → `[3, 2, 1]`

4.10. osszefuz(t)

A `osszefuz` a parameterkent kapott String tomb elemeit egy `String` objektumma fuzi ossze es azt az objektumot adja vissza eredmenyul, peldaul:

`osszefuz(["j", "a", "v", "a"])` → `"java"`

`osszefuz(["hello", "␣", "world!"])` → `"hello␣world!"`

`osszefuz([])` → `""`

5. Versenyauto

Ebben a feladatban a `verseny` csomag osztalyainak (`verseny.Verseny, verseny.Versenyauto, verseny.Versenyzo`) fuggvenyeit fogjuk implementalni, aminek soran az objektum orientalt programozasi szemlelet alapjait gyakoroljuk. A feladat 3 fo reszbol all, minden reszben egy vagy tobb metodus hianyzo logikajat kell potolnod.

Segitseg: docs.oracle.com - Classes, docs.oracle.com - Methods.

5.1. get-set, konstruktor

Implementald a `Versenyauto` es `Versenyzo` osztalyok teljes konstruktorait, get-set metodusait.

5.2. gyorsitas, lassitas

Implementald a `Versenyauto` osztaly gyorsit es `lassit` metodusait. Mindket metodus a versenyauto sebesseget modositjak a parameterkent kapott mennyiseggel, amig el nem eri az auto a maximalis sebesseget (200km/h) vagy meg nem all (0km/h). Ha az auto tomege 1000kg felett van, gyorsitani es lassitani is egyszerre csak 10km/h-al lehet, alatta 15km/h-val. Ha ezeknel az ertekeknel nagyobbat kap a fuggveny, akkor is csak 10 vagy 15km/h-val fogja modositani a sebesseget. Negativ bemenetet egyik fuggveny sem kezel, a sebesseg maradjon valtozatlan.

5.3. toString

A `Versenyauto` osztaly `toString` fuggvenye a versenyauto adatait egy String objektumban osszefoglalja es azt az objektumot adja vissza eredmenyul. A String-nek a kovetkezo formatumot kell kovetnie egy pelda Versenyauto eseteben:

```
1 Nev: Michael Schumacher (1969)
2 Auto: Mercedes (740kg)
```

Kodreszlet 4. Pelda toString kimenet

5.4. verseny

A `Verseny` osztaly `verseny` fuggvenye ket versenyauto peldany kap bemenetkent, a ket autot megversenyezteti es a verseny nyertesenek nevet adja vissza eredmenyul. A verseny nyerteset a kovetkezo keppen dontjuk el: ha az egyik auto tomege 100kg-al vagy tobbel alacsonyabb mint a masik auto tomege, akkor a konnyebbik auto fog nyerni. Ha a kulonbseg a tomegek kozott kevesebb mint 100kg, akkor veletlenszeruen 50-50% esellyel nyerhet barmelyik versenyzo.

6. Megoldas beadasa

Ha minden feladatot megoldottal es az osszes teszt sikeresen lefut, akkor gyozodj meg rola, hogy a kodod tiszta es olvashato, nem tartalmaz felesleges, kommentelt vagy nem mukodo kodot, majd a teljes projektet (tehat nem csak a src mappat vagy csak a .java fajlokat) csomagold be egy .zip kiterjesztesu fajlba, aminek a neve legyen peldaul `DanielSzabo_JavaSE1.zip`. Ezt a fajlt utana vagy el kell kuldened e-mail-ben vagy fel kell feltoltened, errol kulon fogsz pontos utmutatast kapni.