Beadandó 3: Törtszámok (Határidő: 09.21 23:59)

# Követelmények

## Tudnivalók

* JAVA programozási nyelvben kell írni a beadandót
* A beadandót meg kell védeni majd órán
* Minden részfeladat egy plusz jegy
* Részben megoldott feladatért jár részpont, ha az előtte lévők készen vannak
* Részfeladatokat sorrendben kell megoldani, mert épülnek egymásra
* Fordítási- vagy futási hiba => 1-es jegy
* **Minden szint megoldását (egész projektet) külön mentsd le és küldd el! (Így könnyebb javítani)**

## Egyéb elvárások/törekvések

* Konvenciók követése
* Optimalizáció
* Erőforrásokat helyes kezelése
* Kódredundancia kerülése, függvények és eljárások használata
* Részfeladatok és nem egyértelmű függvények és eljárások működésének pár szavas leírása felettük komment formájában
* Kivételkezelés

## Jó tanácsok

* Minden új funkciót tesztelj, hogy az esetleges hiba keletkezésekor könnyebb dolgod legyen

# Feladat leírás

## Bevezetés

Ez a beadandó az alapvető objektum-orientált gondolkodást teszi próbára, pl.: adattagok, metódusok (ezeknek osztályszintű változata), konstruktorok, stb használatát.

Feladatod a ***Fraction.java*** osztály megvalósítása és az evvel való programozási feladatok megoldása (5-ösért végén). Ennek az osztálynak rendelkeznie kell az alapműveletekkel (+, -, \*, /, ^), egyszerűsítéssel és bővítéssel és még egyéb funkciókkal. Ezeket a részletes feladatleírásban találod. Rendelkezésedre áll egy segédosztály ***MathUtils.java*** néven. Ebben olyan függvényeket találsz, amik hasznosak lehetnek. Ezeket használhatod a megoldásodban

## *MathUtils*

|  |  |
| --- | --- |
| **GCD**(int, int) : / **GCD**(int…) : | Legnagyobb közös osztó két vagy több számra |
| **LCM**(int, int) : / **LCM**(int…) : | Legkisebb közös többszörös két vagy több számra |
| **isPrime**(int) : | Igaz, ha a paraméterben adott szám prím |

## Mappaszerkezet

A projektednek az alábbi mappaszerkezetet (***package szerkezetet***) kell követnie:

* src // forrás mappa
  + main
    - model
      * ***Fraction.java***
      * ***MathUtil.java*** // ezt a fájlt neked kell ide behúzni
    - ***Main.java*** // további feladatok
  + resources
    - *input.txt*
    - *output.txt*
  + test// erre majd nekem lesz szükségem
    - model

# Feladatok

Alapvetően adattagok privát-, metódusok és konstruktorok publikus láthatósággal rendelkeznek, ha nincs megadva más.

## 2-es szint

Ezen a szinten a törtszámok alapvető funkcióinak megvalósítása a feladat.

### **Adattagok**

A törtszámokat számlálójával és a nevezőjével fogjuk reprezentálni, ezen felül eltároljuk, hogy egész szám-e.

|  |  |  |
| --- | --- | --- |
| **Név** | **Típus** | **Leírás** |
| numerator | int | Tört számlálója |
| denominator | int | Tört nevezője |
| isInteger | boolean | Igaz, ha a tört egy egész szám |

### **Konstruktorok**

Az osztálynak több konstruktora lesz, ezeket későbbi szinteknél ki kell egészíteni. Ha egy törtszám valójában egész szám, akkor az ***isInteger*** adattag legyen igaz.

|  |  |
| --- | --- |
| **Típus** | **Feladat** |
| Kétparaméteres konstruktor | Tört helyes inicializálása számlálójával és nevezőjével |
| Egyparaméteres konstruktor | Tört inicializálása egész számként |
| Nullaparaméteres konstruktor | 0-val egyenlő tört inicializálása |
| Másoló konstruktor | Tört inicializálása egy másik alapján |

### **Metódusok**

Alapműveletek, hatványozás és egyéb metódusok implementálása. Ezeknek kiegészítése későbbi szinteken.

|  |  |
| --- | --- |
| **Név(Paraméterek) : Visszatérési érték** | **Feladat** |
| add(Fraction) : Fraction | Az eredeti tört összeadása a paraméterben megadott törttel, az eredmény mentése az objektumba[[1]](#footnote-1) és az eredeti objektummal való visszatérés[[2]](#footnote-2) |
| sub(Fraction) : Fraction | … kivonása … |
| mul(Fraction) : Fraction | … szorzása … |
| div(Fraction) : Fraction | … osztása … |
| pow(int) : Fraction | … hatványozása a paraméterben megadott kitevővel, … |
| opposite() : Fraction | Tört másolatának ellentettjével való visszatérés |
| reciprocal() : Fraction | Tört másolatának reciprokával való visszatérés, ezenfelül dobjon ***ArithmeticException*** kivételt, ha nem létezik a tört reciproka. |

|  |  |  |
| --- | --- | --- |
| **Alapfüggvények** | **Feladat** | **Egyéb kikötések** |
| toString | * Ha a tört egész szám, akkor egész formátumban íródjon ki * Egyébként „számláló/nevező” alakban | - |
| Getterek | Adattaggal való visszatérés | Csak a számláló és a nevező adattagjára legyen getter |
| Setterek | Adattag beállítása | Csak a számláló és a nevező adattagjára legyen setter,  Setterek láthatósága legyen privát,  Nevező hibás beállítása esetén dobjon a függvény ***IllegalArgumentException*** kivételt |
| equals | Két törtszám egyenlőségének vizsgálata | - |
| hashCode | Objektum hashCode-jával való visszatérés | - |

## 3-mas szint

Az eddigiek során a törtszámokat redundánsan tároltuk/számoltunk velük, itt ezeket kell kijavítani. Ennél a feladatrésznél a törtek egyszerűsítését/bővítését kell megvalósítani és az eddigi funkciókat kiegészíteni. Most már arra is ügyelni kell, hogy ha egy törtszám egész számmá egyszerűsíthető, akkor az ***isInteger*** adattag igaz legyen.

### **Metódusok**

|  |  |  |
| --- | --- | --- |
| **Név(Paraméterek) : Visszatérési érték** | **Feladat** | **Egyéb kikötések** |
| simplify() : void | Az eredeti tört egyszerűsítése | Privát láthatóság |
| expand(int) : void | Az eredeti tört bővítése a paraméterben megadott egésszel |
| getSimplestForm() : Fraction | A tört egyszerűsített másolatával való visszatérés | - |
| getExpandedForm(int) : Fraction | A paraméterben megadott egésszel való bővített tört másolatával való visszatérés |

### **Eddigiek kiegészítése**

* A konstruktorokat írd át úgy, hogy automatikusan egyszerűsítse a törteket
  + pl.: *Fraction f = new Fraction(2, 4);* //1/2
* A műveleteket írd át úgy, hogy végeredményül a tört legegyszerűbb alakját mentse le, és evvel térjen vissza.
* equals metódus átírása, hogy egyszerűsített és nem egyszerűsített törtek összehasonlításakor is helyesen működjön.

### **Újabb Konstruktorok**

Hozz létre két új konstruktort az alábbiak alapján.

|  |  |  |
| --- | --- | --- |
| **Típus** | **Feladat** | **Paraméterek** |
| Háromparaméteres konstruktor | Tört helyes inicializálása számlálójával és nevezőjével,  Ezenfelül, ha a ***boolean*** paraméter igaz, akkor az egyszerűsített változat kerüljön mentésre, egyébként a paraméterben megadottak | (int, int, boolean) |
| Kétparaméteres konstruktor | Másoló konstruktorhoz hasonló működés,  Annyi kiegészítéssel, hogy ha a ***boolean*** paraméter igaz, akkor az egyszerűsített változat kerüljön mentésre, egyébként a paraméterben megadottak | (Fraction, boolean) |

## 4-es szint

### **Osztályszintű (statikus) elemek**

* Egészítsd ki az osztály egy publikus osztályszintű konstans adattaggal, ami a -t fogja reprezentálni, ahol
* Valósítsd meg a műveletek osztályszintű variánsait, ahol a paraméterekben megadott törtszámok másolataival végződnek ezek a műveletek (eredeti számokat nem változtatva) és visszatérnek az eredménnyel. Az +, -, \*, / műveleteknek paramétereiben tetszőleges mennyiségű törtszám megadható legyen[[3]](#footnote-3).
* A hatványozás az első paraméterben megadott törtszám másolatának a második paraméterben megadott hatványával térjen vissza. Megint csak ügyelve arra, hogy az eredeti törtszám értéke ne változzon.

## 5-ös szint

### **Metódusok**

|  |  |  |
| --- | --- | --- |
| **Név(Paraméterek) : Visszatérési érték** | **Feladat** | **Egyéb kikötések** |
| evaluate() : double | Törtszám tizedestört alakjával való visszatérés | - |
| parseDouble(Double) : Fraction | Tizedestört átváltása törtszámmá, evvel való visszatérés,  Ha paraméterül ***Math.PI***-t kapja a függvény, akkor az osztály értékével térjen vissza. | Osztályszintű metódus |
| parseString(String) : Fraction | String átalakítása törtszámmá, evvel való visszatérés,  Ha nem végezhető el az átalakítás, akkor a dobjon ***IllegalArgumentException*** kivételt a függvény |

### **Rendezés megvalósítása**

A ***Comparable*** interface megfelelő függvényének felülírásával implementáld a törtszámokra való rendezést.

### **Fraction osztállyal való feladatok (*Main.java*)**

Az input fájlban lévő tört és tizedestört számokkal végezd el a műveleteket, és az eredményt írd bele az output fájlba tört és tizedestört alakban.

* Feltesszük, hogy az input fájl nem üres és helyesen van kitöltve (ld. *input.txt*)
* Tegyük fel, hogy minden művelet zárójelezve van, azaz ((3/4 + 12/42) \* 3/2) …

1. „Eredmény mentése az objektumba”: ***a.add(b)*** esetén, az összeadás eredménye mentése az ***a*** objektumba. [↑](#footnote-ref-1)
2. „Eredeti objektummal való visszatérés”: Az előbbi összeadás esetén ***a*** objektummal térjen vissza a függvény, ne csak az eredmény másolatával. [↑](#footnote-ref-2)
3. Például az add(Fraction… fractions) esetében függvény törzsében a franctions egy tömböt reprezentál. Ennek segítségével lehet elérni, hogy egy függvény tetszőleges mennyiségű elemet kapjon paraméterül. Ebben az esetben add(fraction1, fraction2, fraction3, …) ⬄ fraction1 + fraction2 + fraction3 + … eredménnyel fog visszatérni. [↑](#footnote-ref-3)