Java objektumok

Készítette: Goldschmidt Balázs, BME IIT, 2019.

1 Complex osztály letöltése

a) Hozzunk létre egy új Java projektet *complex* néven!
b) Hozzunk létre egy *complex* nevű csomagot a projektben!
c) Töltsük le a *Complex* osztály forráskódját a tárgy honlapjáról, és a fájlt másoljuk a projekt *src/complex* nevű mappájába!
d) Fordítsuk le és futtassuk a programot!

2 Complex osztály bővítése

Bővítsük az előző feladat *Complex* osztályát az alábbi publikus, példányszintű metódusokkal!

a) public String toString() : módosítsuk úgy a komplex szám kiíratását, hogy ha a képzetes rész negatív, akkor ne +, hanem - jel álljon a valós és a képzetes rész között.

Komplex szám	Régi kiíratás	Új kiíratás
3 + 4i	3.0+4.0i	3.0+4.0i
1 + 0i	1.0+0.0i	1.0+0.0i
5 - 3i	5.0+-3.0i	5.0-3.0i

b) `public Complex(double d)` : olyan konstruktor, amelyikben csak a valós részt adjuk meg, a képzetes rész értéke 0.0.
Próba: hozzuk létre az alábbi komplex számokat, számoljuk ki az összegüket!

Komplex számok	Eredmény
1; 2; 3;	6.0+0.0i
5; 10;	15.0+0.0i

c) `public Complex mult(Complex x)` : visszaadja a szám és *x* komplex szorzatát.
A szorzat számítása, ha az egyik szám *a+bi* a másik pedig *c+di* alakú:

$$(a+bi)(c+di) = (ac-bd)+(ad+bc)i$$

Komplex számok	Eredmény
1+2i; 2-3i;	8.0+1.0i
2+0i; 0+5i;	0.0+10.0i

3 Időt ábrázoló osztály készítése

Hozzunk létre egy időt (óra, perc) ábrázoló osztályt (*Time*)! A feladat értelme, hogy bemutasson olyan osztályt, ahol a belső reprezentáció eltér a külső megjelenéstől.

a) Az osztály kezdetben az alábbi jellemzőkkel rendelkezzen:

Attribútumok (privátak):

- *min*: egész érték, az éjfél óta eltelt percek száma

Konstruktorok (publikusak):

- 2 paraméter: *hh:mm*-t hozza létre, ahol *hh* és *mm* paraméter (min = 60*hh+mm)

Metódusok (publikusak):

- getter metódusok (*getMins*, *getHours*): visszaadják az óra és a perc értékét. Pl ha *min*=72, akkor *getMins()* a 12, *getHours()* az 1 értékkel tér vissza.
- *toString()*: visszaadja a String reprezentációt HH:mm formában, 24 órás jelleggel: pl. 08:05; 13:00; stb.

Próbáljuk ki néhány időpont létrehozásával és kiíratásával az elkészült osztályt!

b) Bővítsük az osztályt az alábbi módon:

Konstruktorok (publikusak):

- 0 paraméter: *00:00*-t hozza létre
- 1 paraméter: *hh:00*-t hozza létre, ahol *hh* a paraméter (min = 60*hh)

Metódusok (publikusak):

- *getMinsOnly*: visszaadja *min* attribútum értékét
- alapműveletek: összeadás (*add*), kivonás (*sub*). Az eredmény a *this* objektum és a paraméter összege, különbsége. Pl 2:10+3:14 eredménye 5:24; 2:10-1:34 eredménye 1:36. Negatív értéknél 24 órás maradékkal számoljunk: 2:10-4:10 eredménye 22:00.

Próbáljuk ki a műveleteket a korábban létrehozott objektumok felhasználásával. Ellenőrizzük a helyes működést!

c) Bővítsük tovább az osztályt az alábbi módon:

Metódusok (publikusak):

- összeadás órára és percre (*addMins*, *addHours*): paraméternek megfelelő számú percet vagy órát ad az értékhez. 24 órát átlépve 00:00-ról kezdünk.
- összehasonlító metódusok: this és a paraméter közül melyik a kisebb, nagyobb, vagy egyenlők-e?
  - *boolean greaterThan(Time t)*: *this* nagyobb-e mint *t*?
  - *boolean lessThan(Time t)*: *this* kisebb-e mint *t*?
  - *int compareTo(Time t)*: klasszikus összehasonlító metódus. Értéke negatív, ha *this* kisebb; pozitív, ha *t* kisebb; 0, ha azonosak.

Próbáljuk ki a műveleteket a korábban létrehozott objektumok felhasználásával. Ellenőrizzük a helyes működést!

4 Törteket ábrázoló osztály készítése

Hozzon létre egy új osztályt *Fraction* (tört) néven! Az osztály az alábbi jellemzőkkel rendelkezzen:

Attribútumok (privátak):

- *num*: egész érték, a tört számlálója (numerator)
- *den*: egész érték, a tört nevezője (denominator)

  (A tört értéke mindig *num/den* lesz.)

Konstruktorok (publikusak):

- 0 paraméter: *0/1*-et hozza létre
- 1 paraméter: *p/1*-et hozza létre, ahol *p* a paraméter
- 2 paraméter: *p/q*-t hozza létre, ahol *p* és *q* paraméter

Metódusok (publikusak):

- getter metódusok.Visszaadják *num* illetve *den* értékét.
- *doubleValue*: nincs paramétere, visszatérési értéke a tört értéke double-ként
- *reciprocal*: visszatérési érték egy új tört, ami az eredeti reciproka (számláló és nevező felcserélve.
- alapműveletek: összeadás (*add*), kivonás (*sub*), szorzás (*mult*), osztás (*div*). Nem szükséges az eredmény egyszerűsítése. Az eredmény a this objektum és a paraméter összege, különbsége, stb.
- *szorgalmi feladat*: implementáljunk egy privát metódust, ami két integer paraméternek megadja a legnagyobb közös osztóját! (Pl. használjuk az euklideszi algoritmust.) Ennek segítségével készítsünk törtet egyszerűsítő függvényt (*simplify*).
- *szorgalmi feladat*: implementáljunk összehasonlító metódust: this és a paraméter közül melyik a kisebb, nagyobb, vagy egyenlők-e?
  - *boolean greaterThan(Fraction f)*: *this* nagyobb-e mint *f*?
  - *boolean lessThan(Fraction f)*: *this* kisebb-e mint *f*?
  - *int compareTo(Fraction f)*: klasszikus összehasonlító metódus. Értéke negatív, ha *this* kisebb; pozitív, ha *f* kisebb; 0, ha azonosak.

Próbáljuk ki az osztályt és a metódusait!