Działania na ułamkach — budujemy klasę Fraction

Rozwiązując systematycznie kolejne zadania, będziemy tworzyć własną klasę `Fraction` (ułamki) oraz programy ukazujące możliwości tej klasy.

Zadanie 1

Zbuduj klasę `Fraction`, która będzie zawierać dwa prywatne pola reprezentujące licznik i mianownik ułamka. W klasie tej umieść konstruktor z dwoma parametrami, który będzie budować ułamek na podstawie dwóch liczb całkowitych (licznika i mianownika), oraz publiczną metodę `toString()`, zwracającą ułamek w postaci łańcucha znaków, np. `"4/13"` (licznik, kreska ułamkowa `/` i mianownik). Napisz program pokazujący działanie konstruktora i zdefiniowanej metody.

Zadanie 2

Dodaj do klasy `Fraction` konstruktor bezparametrowy budujący ułamek odpowiadający liczbie *0* oraz konstruktor z jednym parametrem całkowitym `m` budujący ułamek `m/1`. Napisz program pokazujący działanie tych konstruktorów.

Zadanie 3

Utwórz w klasie `Fraction` konstruktor kopiujący i napisz program pokazujący działanie tego konstruktora.

Zadanie 4

Zauważmy, że obiekty `new Fraction(-3, 4)` i `new Fraction(3, -4)` reprezentują ten sam ułamek $-\frac{3}{4}$. Po zamianie obiektów na łańcuchy znaków (metodą `toString()`) otrzymamy odpowiednio `"-3/4"` i `"3/-4"`. Podobnie wyglądałaby sytuacja dla obiektów `new Fraction(2, 5)` i `new Fraction(-2, -5)` reprezentujących ułamek `2/5`. Zapis ułamka w postaci `"3/-4"` lub `"-2/-5"` nie wygląda korzystnie (lepszy będzie zapis `"-3/4"` lub `"2/5"`). Można przyjąć, że będziemy zapamiętywali zawsze dodatni mianownik, a znak licznika zadecyduje o znaku ułamka. Dodaj do klasy `Fraction` prywatną metodę, która wywołana wewnątrz konstruktora skoryguje licznik i mianownik ułamka zgodnie z przyjętą umową. Napisz program pokazujący skutki działania tej metody.

> Metodę nazwij `correction()` (poprawka). Będzie ona przydatna podczas wyznaczania odwrotności ujemnego ułamka lub dzielenia przez taki ułamek.

Zadanie 5

Zauważmy, że obiekty `new Fraction(3, 4)` i `new Fraction(15, 20)` reprezentują ten sam ułamek *3/4*. Dodaj do klasy `Fraction` publiczne metody służące do skracania ułamka (przez największy wspólny dzielnik licznika i mianownika lub inną podaną wartość) oraz publiczną metodę pozwalającą na rozszerzanie ułamka. Napisz program pokazujący działanie tych metod.

> Metodę skracającą ułamek nazwij `reduce()` (ang. *reducing fraction* — skracać ułamek). Do realizacji tej metody niezbędna będzie prywatna metoda obliczająca najmniejszy wspólny dzielnik (ang. *Greatest Common Factor* — *GCF*). Proponujemy w tym przypadku użycie polskiego skrótu `nwd`. Metodę rozszerzającą ułamek nazwij `equivalent()` (ang. *equivalent fraction* — ułamek).

Zadanie 6

W klasie `Fraction` utwórz metody zwracające nowy obiekt `Fraction`, będący iloczynem ułamka reprezentowanego przez ten obiekt i inny obiekt lub liczbę całkowitą. Napisz program pokazujący działanie tych metod.

> Dzięki możliwości przeciążania nazw metod obie metody mogą mieć tę samą nazwę `mult()` (ang. *multiplication* — mnożenie).

Zadanie 7

W klasie `Fraction` utwórz metody statyczne (o nazwie `product()`, ang. *product* — iloczyn) zwracające obiekt `Fraction`, będący iloczynem dwóch ułamków, ułamka i liczby całkowitej lub dwóch liczb całkowitych. Napisz program demonstrujący działanie tych metod.

Zadanie 8

W klasie `Fraction` utwórz metodę zwracającą nowy obiekt `Fraction`, reprezentujący ułamek odwrotny do ułamka zawartego w obiekcie wywołującym tę metodę. Utwórz metodę statyczną o podobnej funkcjonalności. Napisz program pokazujący działanie tych metod. Wykorzystaj odwrotność do obliczenia ilorazu dwóch ułamków.

> Metodom nadaj nazwę `multInv()`, pochodzącą od określenia ang. *multiplicative inverse* — odwrotność (liczby).

Zadanie 9

W klasie `Fraction` utwórz metody zwracające nowy obiekt `Fraction`, będący ilorazem ułamka reprezentowanego przez ten obiekt i inny obiekt lub liczbę całkowitą.

> Metodom nadaj nazwę `div()` (ang. *division* — dzielenie).

Zadanie 10

W klasie `Fraction` utwórz metody statyczne (o nazwie `quot()`, ang. *quotient* — iloraz) zwracające obiekt `Fraction`, będący ilorazem dwóch ułamków, ułamka i liczby całkowitej lub dwóch liczb całkowitych. Napisz program demonstrujący działanie tych metod.

Zadanie 11

W klasie `Fraction` utwórz metody (o nazwie `add()`, ang. *addition* — dodawanie) zwracające nowy obiekt `Fraction`, będący sumą ułamka reprezentowanego przez ten obiekt i inny obiekt lub liczbę całkowitą. Napisz program pokazujący działanie tych metod.

> Podczas dodawania ułamków o różnych mianownikach niezbędne jest sprowadzenie ułamków do wspólnego mianownika.
>
> Wspólnym mianownikiem może być iloczyn mianowników, czyli dodawanie możemy wykonać według wzoru $\frac{a}{b}+\frac{c}{d}=\frac{ad}{bd}+\frac{bc}{bd}=\frac{ad+bc}{bd}$. Korzystniej będzie jednak obliczyć najmniejszą wspólną wielokrotność (*NWW(b, d)*) mianowników i tę liczbę przyjąć jako wspólny mianownik (zob. rozwiązania zadań 15.7 lub 15.9). Prywatną metodę `nww()` dodaj do klasy `Fraction`.

Zadanie 12

W klasie `Fraction` utwórz metody statyczne (o nazwie `sum()`, ang. *sum* — suma) zwracające obiekt `Fraction`, będący sumą dwóch ułamków lub ułamka i liczby całkowitej. Napisz program demonstrujący działanie tych metod.

Zadanie 13

W klasie `Fraction` utwórz metodę zwracającą nowy obiekt `Fraction`, reprezentujący ułamek przeciwny do ułamka zawartego w obiekcie wywołującym tę metodę. Utwórz metodę statyczną o podobnej funkcjonalności. Napisz program pokazujący działanie tych metod. Wykorzystaj ułamek przeciwny do obliczenia różnicy dwóch ułamków.

> Metodom nadaj nazwę `addInv()`, pochodzącą od określenia ang. *additive inverse* — liczba przeciwna.

Zadanie 14

W klasie `Fraction` utwórz metody (o nazwie `sub()`, ang. *subtraction* — odejmowanie) zwracające nowy obiekt `Fraction`, będący różnicą ułamka reprezentowanego przez ten obiekt i inny obiekt lub liczbę całkowitą. Napisz program pokazujący działanie tych metod.

> Odejmowanie możemy wykonać według wzoru $\frac{a}{b}-\frac{c}{d}=\frac{ad}{bd}-\frac{bc}{bd}=\frac{ad-bc}{bd}$ lub stosując jako wspólny mianownik *NWW(b, d)*. Proponujemy uprościć sprawę i zastąpić odejmowanie dodawaniem liczby przeciwnej: $\frac{a}{b}-\frac{c}{d}=\frac{a}{b}+\frac{-c}{d}$.

Zadanie 15

W klasie `Fraction` utwórz metody statyczne (o nazwie `diff()`, ang. *difference* — różnica) zwracające obiekt `Fraction`, będący różnicą dwóch ułamków lub ułamka i liczby całkowitej. Napisz program demonstrujący działanie tych metod.

Zadanie 16

Dodaj do klasy `Fraction` metody (`getNum()` i `getDen()`) pozwalające na odczytanie wartości prywatnych pól obiektu (licznika i mianownika ułamka reprezentowanego przez obiekt). Napisz program demonstrujący działanie tych metod.

Zadanie 17

Dopuszcza się zmiany wartości prywatnych pól obiektu (przy użyciu metod) w celu zmiany jego wartości bez tworzenia nowej instancji obiektu. Zdefiniuj w klasie `Fraction` metody `setNum(int)`, `setDen(int)` i `setFrac(int, int)`, które będą zmieniać wartość licznika, mianownika lub jednocześnie licznika i mianownika ułamka (obiektu). Napisz program demonstrujący działanie tych metod.