Zadanie 9 *19, 20 stycznia 2017 r.*

KURS JĘZYKA JAVA

ZBIÓR LICZB WYMIERNYCH

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

Zdefiniuj klasę `RationalNumber` dziedziczącą po klasie `Number` i implementującą interfejs `Comparable<RationalNumber>`, przeznaczoną do reprezentowania liczb wymiernych. Obiekty tej klasy mają być niemodyfikowalne — pola na licznik i mianownik typu `BigInteger` można więc zadeklarować jako publiczne i ustalone.

W obiekcie typu `RationalNumber` mianownik jest zawsze dodatni oraz największy wspólny dzielnik dla licznika i mianownika ma zawsze wynosić 1 (mianownik w liczbie wymiernej będzie równy 1 zawsze wtedy, gdy mamy przechowujemy liczbę całkowitą).

Liczbę wymierną inicjalizuj za pomocą dwóch liczb: licznika i mianownika. Powinna też istnieć możliwość zainicjalizowania liczby wymiernej napisem typu `String` reprezentującym taką liczbę zapisaną dziesiętnie (domyślnie), dwójkowo albo szesnastkowo. Metoda `toString` też powinna umieć wypisać liczbę wymierną w jednym z wymienionych systemów liczbowych. Sam system liczbowy (dziesiętny, dwójkowy i szesnastkowy) zdefiniuj jako wyliczenie `enum`.

W klasie `RationalNumber` zdefiniuj metody odpowiadające podstawowym operacjom arytmetycznym (dodawanie, odejmowanie, mnożenie, dzielenie, modulo, zmiana znaku, wartość bezwzględna, zaokrąglenie do najbliższej liczby całkowitej w górę i w dół, itp).

Zadanie 2.

Zdefiniuj zbiór liczb `NumberSet` jako kolekcję generyczną, gdzie przez liczbę będziemy rozumieć dowolną klasę dziedzicząca po `Number`. Kolekcję tą zaimplementuj na tablicy liczb. Sama kolekcja ma korzystać z tablicy dynamicznej o zmiennej pojemności, adoptującej się automatycznie do liczby elementów. Kolekcja ma umieszczać elementy zapełniając teblicę od początku; przy usuwaniu elementów przenoś z końca tablicy elementy w miejsce usuniętych.

Kolekcja `NumberSet` ma implementować dość bogaty interfejs `Collection` (zawiera on około 20 różnych metod) — możesz więc dziedziczyć po klasie `AbstractCollection`. Implementując `NumberSet` pamiętaj, aby w zbiorze nie znalazły się dwa elementy o takiej samej wartości.

W kolekcji dla zbioru liczb zdefiniuj też iterator do przechodzenia po wszystkich zgromadzonych w zbiorze liczbach, tak aby można było użyć konstrukcji:

```
for (T wartosc : zbior) {
    // używaj w pętli zmiennej 'wartosc' typu generycznego T
    // pochodzącej z kolekcji 'zbior'
}
```

Iterator ten musi być zdefiniowany, ponieważ interfejs `Collection` rozszerza interfejs `Iterable`. Iterator możesz zdefiniować jako klasę zagnieżdżoną implementującą interfejs `Iterator`.

Przetestuj swoją kolekcję dla danych typu `RationalNumber`.

Uwaga.

W kolekcji `NumberSet` nie wolno korzystać z żadnych kolekcji standardowych za wyjątkiem `AbstractCollection`.