

Task #1

Класс `RationalVector2D` - двумерный вектор, компоненты которого являются рациональными дробями (т.е. объектами класса `RationalFraction`). Это и есть атрибуты класса. Методы:

- `RationalVector2D()` - конструктор для нулевого вектора (компоненты должны быть равны нулевым рациональным дробям);
- `RationalVector2D(RationalFraction, RationalFraction)` - конструктор вектора с координатами; в конструкторах устраняйте дублирование кода;
- `RationalVector2D add(RationalVector2D)` - сложение вектора с другим вектором;
- `String toString()` - строковое представление вектора (использует строковое представление `RationalFraction`);
- `double length()` - длина вектора;
- `RationalFraction scalarProduct(RationalVector2D)` - скалярное произведение вектора на другой вектор;
- `boolean equals(RationalVector2D)` - сравнить вектор с другим вектором (опираться на `equals` у `RationalFraction`);

Task #2

Класс `ComplexVector2D` - двумерный вектор, компоненты которого являются комплексными числами (т.е. объектами класса `ComplexNumber`). Это и есть атрибуты класса. Уже не буду писать, что надо в операциях опираться на методы класса `ComplexNumber`. Методы:

- `ComplexVector2D()` - конструктор для нулевого вектора (компоненты должны быть равны нулевым комплексным числам);
- `ComplexVector2D(ComplexNumber, ComplexNumber)` - конструктор вектора с координатами; в конструкторах устраняйте дублирование кода;
- `ComplexVector2D add(ComplexVector2D)` - сложение вектора с другим вектором;
- `String toString()` - строковое представление вектора.
- `ComplexNumber scalarProduct(ComplexVector2D)` - скалярное произведение вектора на другой вектор;
- `boolean equals(ComplexVector2D)` - сравнить вектор с другим вектором;

Task #3

Создать класс `RationalMatrix2x2` - двумерная матрица из `RationalFraction`. Аргументы - содержимое матрицы (лучше разумеется хранить двумерным массивом, а то замучаетесь). Методы:

- `RationalMatrix2x2()` - конструктор для нулевой матрицы;
- `RationalMatrix2x2(RationalFraction)` - конструктор для матрицы, у которой каждый элемент равен поданному числу;
- `RationalMatrix2x2(RationalFraction, RationalFraction, RationalFraction, RationalFraction)` - конструктор на 4 дробях. В конструкторах устраняйте дублирование кода;
- `RationalMatrix2x2 add(RationalMatrix2x2)` - сложение матрицы с другой;
- `RationalMatrix2x2 mult(RationalMatrix2x2)` - умножение матрицы на другую матрицу;
- `RationalFraction det()` - определитель матрицы;
- `RationalVector2D multVector(RationalVector2D)` - умножить матрицу на двумерный вектор (считая его столбцом) и вернуть получившийся столбец в виде вектора.

Task #4

Создать класс `ComplexMatrix2x2` - двумерная матрица из `ComplexNumber`. Аргументы - содержимое матрицы (лучше разумеется хранить двумерным массивом, а то замучаетесь). Методы:

- `ComplexMatrix2x2()` - конструктор для нулевой матрицы;
- `ComplexMatrix2x2(ComplexNumber)` - конструктор для матрицы, у которой каждый элемент равен поданному числу;
- `ComplexMatrix2x2(ComplexNumber, ComplexNumber, ComplexNumber, ComplexNumber)` - конструктор на 4 дробях, в конструкторах устраняйте дублирование кода;
- `ComplexMatrix2x2 add(ComplexMatrix2x2)` - сложение матрицы с другой;
- `ComplexMatrix2x2 mult(ComplexMatrix2x2)` - умножение матрицы на другую матрицу;
- `ComplexNumber det()` - определитель матрицы;
- `ComplexVector2D multVector(ComplexVector2D)` - умножить матрицу на двумерный комплекснозначный вектор (считая его столбцом) и вернуть получившийся столбец в виде вектора.

Task #5

Создать класс `RationalComplexNumber` - комплексное число, компонентами которого являются рациональные дроби. Атрибуты - `RationalFraction`). Методы:

- `RationalComplexNumber()` - конструктор для нулевого комплексного числа;
- `RationalComplexNumber(RationalFraction, RationalFraction)` - конструктор комплексного числа с заданными значениями вещественной и мнимой части; в конструкторах устраняйте дублирование кода;
- `RationalComplexNumber add(RationalComplexNumber)` сложение с другим таким числом;
- `RationalComplexNumber sub(RationalComplexNumber)` - вычитание;
- `RationalComplexNumber mult(RationalComplexNumber)` - умножение;
- `String toString()` - строковое представление такого комплексного числа. Только без всяких " $2/3 * i + -3/5$ ". Проверяйте знаки, чтобы было красиво: $2/3 * i - 3/5$.

Task #6

Класс `RationalComplexVector2D` - двумерный вектор, компоненты которого являются объектами класса `RationalComplexNumber`. Это и есть атрибуты класса. Уже не буду писать, что надо в операциях опираться на методы класса `RationalComplexNumber`. Методы:

- `RationalComplexVector2D()` - конструктор для нулевого вектора (компоненты должны быть равны нулевым рациональным дробям);
- `RationalComplexVector2D(RationalComplexNumber, RationalComplexNumber)` - конструктор вектора с координатами; в конструкторах устраняйте дублирование кода;
- `RationalComplexVector2D add(RationalComplexVector2D)` - сложение вектора с другим вектором;
- `String toString()` - строковое представление такого вектора.
- `RationalComplexNumber scalarProduct(RationalComplexVector2D)` - скалярное произведение вектора на другой вектор;

Task #7

Создать класс `RationalComplexMatrix2x2` - двумерная матрица из `RationalComplexNumber`. Аргументы - содержимое матрицы (лучше, разумеется, хранить двумерным массивом, а то замучаетесь). Методы:

- `RationalComplexMatrix2x2()` - конструктор для нулевой матрицы;
- `RationalComplexMatrix2x2(RationalComplexNumber)` - конструктор для матрицы, у которой каждый элемент равен поданному числу;
- `RationalComplexMatrix2x2(RationalComplexNumber, RationalComplexNumber, RationalComplexNumber, RationalComplexNumber)` - конструктор на 4 дробях, в конструкторах устраняйте дублирование кода;
- `RationalComplexMatrix2x2 add(RationalComplexMatrix2x2)` - сложение матрицы с другой;
- `RationalComplexMatrix2x2 mult(RationalComplexMatrix2x2)` - умножение матрицы на другую матрицу;
- `RationalComplexNumber det()` - определитель матрицы;
- `RationalComplexVector2D`
`multVector(RationalComplexVector2D)` - умножить матрицу на двумерный комплекснозначный рациональный вектор (считая его столбцом) и вернуть получившийся столбец в виде вектора.