**1**

Класс Vector2D - двумерный вектор. Атрибуты - два вещественных числа (координаты). Далее (здесь и в последующих подобных задачах) указываю методы с типом возвращамых значений, а в скобках пишу только типы параметров. get- и set- методы создавать по необходимости (тоже здесь и далее).   
• Vector2D() - конструктор для нулевого вектора;  
• Vector2D(double, double) - конструктор вектора с координатами; в конструкторах устраняйте дублирование кода;  
• Vector2D add(Vector2D) - сложение вектора с другим вектором, результат возвращается как новый объект.  
• void add2(Vector2D) - сложение вектора с другим вектором, результат сохраняется в том, у кого был вызван этот метод;  
• Vector2D sub(Vector2D) - вычитание из вектора другого вектора, результат возвращается как новый объект;  
• void sub2(Vector2D) - вычитание из вектора другого вектора, результат сохраняется в том векторе, у кого был вызван этот метод;  
• Vector2D mult(double) - умножение вектора на вещественное число, результат возвращается как новый объект;  
• void mult2(double) - умножение вектора на вещественное число, результат сохраняется в векторе;  
  
• String toString() - строковое представление вектора;  
• double length() - длина вектора;  
• double scalarProduct(Vector2D) - скалярное произведение вектора на другой вектор;  
• double cos(Vector2D) - косинус угла между этим и другим вектором;  
• boolean equals(Vector2D) - сравнить вектор с другим вектором;

**2**  
Класс RationalFraction - рациональная дробь. Атрибуты - два целых числа (числитель и знаменатель). Методы:  
• RationalFraction() - конструктор для дроби, равной нулю;  
• RationalFraction(int, int) - конструктор дроби со значениями числителя и знаменателя; в конструкторах устраняйте дублирование кода;  
• void reduce() - сокращение дроби;  
• RationalFraction add(RationalFraction) - сложение дроби с другой дробью, результат возвращается как новый объект (не забудьте сократить)  
• void add2(RationalFraction) - сложение дроби с другой дробью, результат сохраняется в том, у кого был вызван этот метод (не забудьте сократить);  
• RationalFraction sub(RationalFraction) - вычитание из дроби другой дроби, результат возвращается как новый объект (не забудьте сократить);  
• void sub2(RationalFraction) - вычитание из дроби другой дроби, результат сохраняется в том, у кого был вызван этот метод (не забудьте сократить);  
• RationalFraction mult(RationalFraction) - умножение дроби на другую дробь, результат возвращается как новый объект (сократить)  
• void mult2(RationalFraction) - умножение дроби на другую дробь, результат сохраняется;  
• RationalFraction div(RationalFraction) - деление дроби на другую дробь, результат возвращается как новый объект (сократить)  
• void div2(RationalFraction) - деление дроби на другую дробь, результат сохраняется; больше не буду писать "возвращается" или "сохраняется", думаю, уже и так понятно.  
• String toString() - строковое представление дроби (например, -2/3);  
• double value() - десятичное значение дроби;  
• boolean equals(RationalFraction) - сравнить дробь с другой дробью   
• (не забывайте, что 2/4 и 1/2 - одна и та же дробь)  
• int numberPart() - целая часть дроби;

**3**  
Если вы не в теме, прочитайте: https://ru.wikipedia.org/wiki/Комплексное\_число  
Создать класс ComplexNumber - комплексное число. Атрибуты - действительная и мнимая части (два числа). Методы:  
• ComplexNumber() - конструктор для нулевого комплексного числа;  
• ComplexNumber(double, double) - конструктор комплексного числа с заданными значениями вещественной и мнимой части; в конструкторах устраняйте дублирование кода;  
• ComplexNumber add(ComplexNumber) - сложение комплексного числа с другим комплексным числом;  
• void add2(ComplexNumber) - сложение комплексного числа с другим комплексным числом;  
• CompexNumber sub(ComplexNumber) - вычитание из комплексного числа другого комплексного числа;  
• void sub2(ComplexNumber) - вычитание из комплексного числа другого комплексного числа;  
• ComplexNumber multNumber(double) - умножение комплексного числа на вещественное число;  
• void multNumber2(double) - умножение комплексного числа на вещественное число;  
• ComplexNumber mult(ComplexNumber) - умножение комплексного числа на другое комплексное число;  
• void mult2(ComplexNumber) - умножение комплексного числа на другое комплексное число;  
• ComplexNumber div(ComplexNumber) - деление на другое комплексное число;  
• void div2(ComplexNumber) - деление на другое комплексное число;  
• double length() - модуль комплексного числа;   
• String toString() - строковое представление комплексного числа. Только без всяких "2 \* i + - 3". Проверяйте знаки, чтобы было красиво: 2 \* i - 3.  
• double arg() - аргумент комплексного числа (может понадобиться тригонометрическое представление (читайте ссылку) и арктангенс Math.atan);  
• ComplexNumber pow(double) - возвести в степень по Формуле Муавра (иные способы запрещены). Внимание - разрешено использование Math.pow для возведения аргумента в степень (т.к. оба аргумента double), также вам понадобятся Math.cos, Math.sin.  
• boolean equals(ComplexNumber) - сравнить комплексное число с другим комплексным числом;

**4**  
Создать класс Matrix2x2 - двумерная матрица из вещественных чисел. Аргументы - содержимое матрицы (лучше, разумеется, хранить двумерным массивом, а то замучаетесь). Методы:  
• Matrix2x2() - конструктор для нулевой матрицы;  
• Matrix2x2(double) - конструктор для матрицы, у которой каждый элемент равен поданному числу;  
• Matrix2x2(double [][]) - конструктор для матрицы, содержимое подается на вход в виде массива;  
• Matrix2x2(double, double, double, double) - глупый конструктор, но пусть он будет. Сами знаете, что он делает. В конструкторах устраняйте дублирование кода;  
• Matrix2x2 add(Matrix2x2) - сложение матрицы с другой;  
• void add2(Matrix2x2) - сложение матрицы с другой;  
• Matrix2x2 sub(Matrix2x2) - вычитание из матрицы другой матрицы;  
• void sub2(Matrix2x2) - вычитание из матрицы другой матрицы;  
• Matrix2x2 multNumber(double) - умножение матрицы на вещественное число;  
• void multNumber2(double) - умножение матрицы на вещественное число;  
• Matrix2x2 mult(Matrix2x2) - умножение матрицы на другую матрицу;  
• void mult2(Matrix2x2) - умножение матрицы на другую матрицу;  
• double det() - определитель матрицы;  
• void transpon() - транспонировать матрицу;  
• Matrix2x2 inverseMatrix() - возвратить обратную матрицу для заданной. Если это невозможно, вывести сообщение об ошибке и вернуть нулевую матрицу (кто вдруг знает исключения, может их использовать).  
• Matrix2x2 equivalentDiagonal() - возвратить эквивалентную диагональную матрицу;  
• Vector2D multVector(Vector2D) - умножить матрицу на двумерный вектор (считая его столбцом) и возвратить получившийся столбец в виде вектора.  
  
*А теперь комбинируем! ^^   
В дальнейших задачах все методы, которые делают операции (сложение, умножение и т.п.) всегда возвращают значения, void среди них нет. Все вспомогательные операции над компонентами должны опираться на операции, написанные в классах 46-49.*  
**5**  
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 RationalFraсtion);  
• 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);  
  
**6**  
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;  
  
**7**  
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(RationalFration) - конструктор для матрицы, у которой каждый элемент равен поданному числу;  
• RationalMatrix2x2(RationalFraсtion, RationalFraсtion, RationalFraсtion, RationalFraсtion) - конструктор на 4 дробях. В конструкторах устраняйте дублирование кода;  
• RationalMatrix2x2 add(RationalMatrix2x2) - сложение матрицы с другой;  
• RationalMatrix2x2 mult(RationalMatrix2x2) - умножение матрицы на другую матрицу;  
• RationalFraсtion det() - определитель матрицы;  
• RationalVector2D multVector(RationalVector2D) - умножить матрицу на двумерный вектор (считая его столбцом) и возвратить получившийся столбец в виде вектора.  
  
**8**  
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 возвратить получившийся столбец в виде вектора.  
  
**9**  
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.  
  
**10**  
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;

**11**  
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 возвратить получившийся столбец в виде вектора.