**2018. ИТИС. ИНФОРМАТИКА. ДОМАШНЯЯ РАБОТА.**

**11-801. 1 КУРС 1 СЕМЕСТР**

*Автор: Абрамский М.М., 2018*

***ВНИМАНИЕ!***

***Обычные*** *домашки помечаются обычным номером (01)*

***Штрафные*** *задания помечаются суффиксом* ***x*** *(04x, например)*

*По умолчанию обычные задачи загружаются в репозиторий в папку homework с указанием номера (файла или, если их несколько, в папку с номером задачи).*

*Штрафные задачи надо загружать в папку penalty с нужным номером.*

Каждый файл с кодом нужно подписывать. Вверху файла комментарии по образцу:

**/\*\*  
\* @author Mikhail Abramskiy  
\* 09-53a  
\* Task 01 (для вспомогательного класса указывайте для чего используется, например for Problem Set 1 Task 01)  
\*/**

**2018.09.20. Темы: циклы, ряды**

**Дедлайн - 2018.09.23 (вс), 21:00**

**01**

Для введенного n вывести «трифорс» (пример ниже), n - высота каждого треугольника. Не использовать строки и массивы символов. Например, для n = 3

\*

\*\*\*

\*\*\*\*\*

\* \*

\*\*\* \*\*\*

\*\*\*\*\* \*\*\*\*\*

**02**

Вводятся целые k, m. Вывести целые числа между k и m, которые делятся на 3.

**03**

Вычислить квадратный корень числа x с точностью до 6 знака после запятой. Не использовать Math.

**04**

Для введенного n приблизительно нарисовать символами круг радиуcа n. Не использовать строки и массивы символов. Например, для n = 10:

\*\*\*\*\*\*\*\*\*\*0\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*000000000\*\*\*\*\*\*

\*\*\*\*0000000000000\*\*\*\*

\*\*\*000000000000000\*\*\*

\*\*00000000000000000\*\*

\*\*00000000000000000\*\*

\*0000000000000000000\*

\*0000000000000000000\*

\*0000000000000000000\*

\*0000000000000000000\*

000000000000000000000

\*0000000000000000000\*

\*0000000000000000000\*

\*0000000000000000000\*

\*0000000000000000000\*

\*\*00000000000000000\*\*

\*\*00000000000000000\*\*

\*\*\*000000000000000\*\*\*

\*\*\*\*0000000000000\*\*\*\*

\*\*\*\*\*\*000000000\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*0\*\*\*\*\*\*\*\*\*\*

(выглядит как овал, но только потому, что расстояние между буквами меньше, чем расстояние между строками, а если такое нарисовать, будет именно кружок)

**05**

Найти ряд Маклорена для функции exp(x) и вычислить ее с точностью 1e-9.

**06**

Найти ряд Маклорена для функции ln(x) и вычислить ее с точностью 1e-9.

**07**

Найти ряд Маклорена для функции cos(x) и вычислить ее с точностью 1e-9.

**08**

Найти ряд Маклорена для функции sin(x) и вычислить ее с точностью 1e-9.

**2018.10.05. Тема - кванторы, цифры числа.**

**Дедлайн - 2018.10.07 (вс), 21:00**

**09**

Вводится n чисел. Проверить, что среди них существует ровно два таких числа, что длина (количество цифр) каждого из них равна 3 или 5, а их цифры либо все четные, либо все нечетные.

**10**

Вводится n чисел. Проверить, что существует ровно три числа, в котором цифры идут по убыванию.

**11**

Вводится целое 2 < k < 9, затем вводится целое число n, которое можно интерпретировать как число в k-ичной системе счисления. Сконвертировать n в десятичную систему счисления. Строковые типы не используются.

**12**

Локальным максимумом в последовательности назовем элемент, который больше предыдущего и следующего (если они есть). Проверить, что во входной последовательности существует четный по значению локальный максимум.

**13**

Проверить, что во входной последовательности целых чисел существует ровно два четных по значению локальных максимума.

**Массивы - обработка массива**

**14**

Заполнить массив размера n следующим образом: a[0]=1, a[1]=-3, a[2]=5, a[3]=-7, и т.д. Вывести полученный массив на экран

**15**

По вводимым в args цифрам a0, a1, … ,an-1 построить число a0a1a2…an-1 (не вывести, как оно будет выглядеть, а именно получить это число в отдельной переменной)  
Здесь же решить задачу c обратным порядком – т.е. построением числа an-1an-2… a2a1

**16**

Вводятся два массива, представляющие собой подстановку. Подстановка например:  
(0,1,2,3,4)  
(4,1,3,0,2)  
где первая строка - индексы, вторая - элементы массива. Умножить одну подстановку на другую, результат записать в первый массив, результат вывести на экран.

**операции с массивами, вставка/удаление в массив, сдвиги и разворот.**

**17**  
Реализовать с помощью длинной арифметики деление одного числа на другое. На экран должны выводиться целая часть и остаток от деления.  
  
**18**  
Реализовать с помощью длинной арифметики умножение одного числа на другое.

**19**  
Реализовать с помощью длинной арифметики умножение числа на цифру.

**Двумерные массивы.**

**20**

Подсчитать в двумерном массиве максимум по суммам элементов на главных диагоналях.  
Пример  
1 2 3  
1 2 3  
0 7 2  
Главные диагонали: 1 2 2, 2 3 0, 3 1 7.   
Суммы: 5, 5, 11. Максимум 11

**21**  
Проверить, что в двумерном массиве каждая сумма элементов на побочных диагоналях является четной

**22**

Вводится целочисленная квадратная матрица размера 2n+1. Заменить нулями элементы, расположенные на верхнем и нижнем треугольниках, образованных пересечением главной и побочной диагоналей. Правый и левый треугольники не трогать. Пример:  
  
Вход:  
1 2 3 4 5  
2 3 4 1 2  
0 3 2 3 1  
9 2 3 1 4   
3 8 0 8 6   
Выход:  
1 0 0 0 5  
2 3 0 1 2  
0 3 2 3 1  
9 2 0 1 4   
3 0 0 0 6  
  
**23**  
Вводится квадратная матрица размера n. Привести ее к треугольному виду и вывести на экран. Не забудьте, что надо обрабатывать случай поиска ненулевого элемента в столбце (поиск – существование – квантор).  
  
**24**  
Подсчитать в двумерном массиве максимум по суммам элементов на главных диагоналях.  
Пример  
1 2 3  
1 2 3  
0 7 2  
Главные диагонали: 1 2 2, 2 3 0, 3 1 7.   
Суммы: 5, 5, 11. Максимум 11  
  
**25**  
Реализовать с помощью двумерных массивов умножение матриц.  
  
**26**  
Проверить, что в трехмерном массиве в каждом его двумерном массиве существует такая строка, что в ней все элементы делятся на три.

**Строки - символьные тип. Тип строка. Методы класса строка.**

**27**

Вводится две строки. Написать функцию, сравнивающую строки в лексикографическом порядке (кто раньше, кто позже по алфавиту). Из строковых методов использовать только length и charAt.  
  
**28**  
Вводится массив строк. С помощью написанной функции выше отсортировать строки в лексикографическом порядке.

**29**

Вводится строка, представляющая собой слова на английском языке, записанные через пробел. Подсчитать, сколько слов начинаются на заглавную букву.

**30**

Вводится строка, представляющая собой английский текст. Подсчитать и вывести, сколько раз встречается каждая буква английского алфавита в тексте. Не различать регистр (‘A’ и ‘a’ – одна и та же буква).

**31**

Вводится строка, представляющая собой английский текст. Каждое второе вхождение строки “true” заменить на “false”. Не использовать методы строки, кроме charAt и length().  
  
**32**  
Вводится строка, представляющая собой английский текст. Каждое вхождение слова «mom” заменить на “dad”. Не использовать методы строки, кроме charAt и length().

**Регулярные выражения для строк. Поиск по регулярному выражению. Проверка строк на соответствие шаблону/правилу.**

**33**  
Напишите регулярное выражения для даты и времени в промежутке с 6 марта 1237 12:00 по 27 февраля 1978 21:35 в формате MM/DD/YYYY HH:MM  
  
**34**  
Напишите регулярное выражение для вещественного числа с периодом.  
Подходят: 0, -6, -0.5, +2, 0,0(64),   
Не подходят: -0, 001, 0,(35)00, -3,750  
  
**35**  
На вход подается массив строк, каждая строка представляет собой двоичный код. Правильным кодом называется строка, которая состоит либо только из нулей, либо только из единиц, либо нули и единицы в ней чередуются  
Например, 010101, 11, 00, 101 - правильные коды, 0110, 001, 11101 - неправильные.  
Написать программу с использованием регулярных выражений, которая выведет номера строк, представляющих собой правильный код (использовать matches)

**36**

Генерировать случайные положительные целые числа c помощью java.util.Random. Вывести первые 10 сгенерированных чисел, в которых нет трех четных цифр подряд. Остановить генератор, вывести общее количество сгенерированных чисел. Проверку осуществлять регулярным выражением. НЕ использовать математические операции для анализа числа (сделать двумя способами – с помощью matches и с помощью find)  
  
**37**  
Генерировать случайные положительные целые числа c помощью java.util.Random. Вывести первые 10 сгенерированных чисел, которые содержат более 3 и менее 6 четных цифр и ни одной нечетной. Остановить генератор, вывести общее количество сгенерированных чисел. Проверку осуществлять регулярным(-и) выражением(-ями). НЕ использовать математические операции для анализа числа (сделать двумя способами – с помощью matches и с помощью find).  
  
**38**  
Генерировать случайные положительные целые числа. Вывести первые 10 сгенерированных чисел, в которых есть как минимум два раза встречается группа из 2 четных цифр. Остановить генератор, вывести общее количество сгенерированных чисел. Проверку осуществлять регулярным выражением. НЕ использовать математические операции для анализа числа. (сделать двумя способами – с помощью matches и с помощью find).

**Языки и грамматики. Автоматы.**

*Для каждой следующей задачи рядом с диаграммой написать регулярное выражение, соответствующее языку, который распознается автоматом. Диаграммы можно нарисовать на компьютере, а можно на листке – и сфотографировать. Файлы картинок выложите в репозиторий.*  
  
**39**  
Нарисовать диаграмму автомата, проверяющего, что входное слово начинается на два одинаковых символа.  
  
**40**  
Нарисовать диаграмму автомата, проверяющего, что входное слово заканчивается на два нуля.  
  
  
**41**  
Нарисовать диаграмму автомата, проверяющего, что входное слово заканчивается на два разных символа.

**42**  
Нарисовать диаграмму автомата, распознающего слова, начинающиеся на два одинаковых символа и заканчивающиеся на два разных символа.  
  
**43**  
Нарисовать диаграмму автомата, распознающего слова, в котором нет подряд ни двух нулей, ни двух единиц.

**ООП - классы и объекты. Конструкторы, методы, атрибуты.**

**44**  
Реализовать модель игры – игроки последовательно друг другу наносят удары силой от 1 до 9, при этом у того, кого ударили, из очков здоровья (health points, hp) отнимается сила удара. Игра заканчивается, когда hp одного из игроков стало <= 0. В задаче должны быть классы Игрок и Игра, в которой весь процесс происходит. У каждого игрока есть имя и hp. Сила удара каждого игрока на каждом шаге вводится из консоли. Процесс игры должен принадлежать объекту класса Игра.  
  
**45**  
В задаче 44 добавить вероятностное распределение – сила удара обратна пропорциональна вероятности попасть. Можно выполнить эту задачу полностью в папке этого задания, тогда 44 будет зачтена.

**46**

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;

**47**  
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;  
  
**48**  
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;

**49**  
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.*  
**50**  
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);  
  
**51**  
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;  
  
**52**  
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.  
  
**53**  
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.  
  
**54**  
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.  
  
**55**  
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;

**56**  
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.  
  
**Абстракция. Выделение классов. Инкапсуляция**

**57**

Придумайте сущности (связанные с реальным миром, IT, Интернетом или чем-либо еще), удовлетворяющие следующим требованиям:  
  
Абстрактный класс  
- 2 атрибута,  
- 2 конструктора (один использует другой),  
- 3 метода (не конструкторы, не set-get) – один не абстрактный, второй абстрактный.  
  
Унаследуйте этот класс в двух других классах. Дочерние классы в добавление к унаследованному от родителей содержат:  
+ 1 атрибут,  
+ 1 свой метод,  
+ реализацию абстрактного метода родителя,  
+ переопределение неабстрактного родительского метода,  
+ 2 конструктора (один использует родительский конструктор, а другой использует первый).  
  
В методе Main создайте массив из объектов дочерних классов с восходящим преобразованием до родительского класса (по аналогии с Animal, Cat и Dog). Затем покажите работу полиморфизма на примере переопределенных абстрактного и не абстрактного метода (по аналогии с voice() у кошки и собаки).  
  
В заключение приведите для одного из дочерних классов еще один возможный родительский класс (т.е. в других языках мы бы реализовали множественное наследование), чтобы была видна проблема ромбовидного наследования (оба родителя содержат один и тот же по заголовку метод, но имеют его разную реализацию).

**Наследование и полиморфизм. Интерфейсы. Связи между классами и интерфейсами.**

**58**

Примитивные типы могут содержать только заданное количество цифр (long – уже 8 байтов, не больше). Но число можно трактовать как набор цифр.  
Реализовать:  
• Интерфейс Number – натуральное число. Методы:  
o Number add(Number n) – прибавить целое число n  
o Number sub(Number n) – вычитание из нашего числа числа n. Т.к. n – натуральное, то если n больше числа, то выкинуть исключение NotNaturalNumberException (реализуйте его самостоятельно как наследник от Exception)  
o int compareTo(Number n) – сравнить число с n. Вернуть 1, если число больше n, -1, если меньше, 0, если равно  
Далее реализовать 2 класса:  
• SimpleLongNumber – класс, в котором число хранится в примитивном типе long.  
• VeryLongNumber – класс, в котором число хранится как массив цифр или строка цифр.  
После реализации классов в методе main создать массив Number [], в котором будут храниться как SimpleLongNumber, так и VeryLongNumber. Вычислить сумму элементов массива (используя add).

**Обработка данных. Хранение инфы в файлах, парсинг данных из файлов. Запросы к этим данным.**

**59**

Дан датасет (набор данных) в формате CSV о скорости открытия браузерами некоторых веб приложений.  
Создать класс, который может хранить строки этого датасета, считать данные в массив/коллекцию объектов этого класса и вывести на экран следующие данные (обработав массив):  
• Какой сайт открывается быстрее всего в среднем по всем браузерам?  
• Какой браузер быстрее всего работает в среднем по всем сайтам? (поле Average (sec) не используйте)  
• Вычислить среднее время открытия браузером сайта в среднем по всем браузерам и всем сайтам  
Ссылка на датасет: https://opendata.socrata.com/Business/Browser-Speed/e3fb-j9c6, нажать на Export, нажать на CSV.   
Подробности о CSV: https://ru.wikipedia.org/wiki/CSV  
  
**60**  
Даны 5 файлов, содержащих информацию о базе данных пользователей в соц.сетях:  
• users.txt с колонками id, username, password, gender, email  
• messages.txt с колонками sender\_id, receiver\_id (оба поля – ссылки на id в users), timesent, text, status (прочитан/не прочитан)  
• subscriptions.txt с колонками subscriber\_id, subscription\_id (оба поля – ссылки на id в users).  
Создать классы, способные хранить информацию о каждой строке этого файла (User, Message, Subsciption). При этом учесть связи между объектами (чтобы они были ссылками, т.е не senderId типа long, а sender типа User). Пол user и статус сообщения реализовать как перечисления (enum).  
Считать информацию из соответствующих файлов в массив/коллекцию соответствующего типа. Выполнить над ними следующие запросы (реализовать методами):  
• Вывести переписку пользователей (вводятся их имена)  
• Вывести список друзей (взаимных подписчиков). Выводятся имена  
• Выяснить, пользователи какого пола отправляли наибольшее количество сообщений.  
• Вывести статистику, сколько сообщений, отправленных пользователями мужского пола женскому, было прочитано и не прочитано (подсчитать проценты), и наоборот – женского мужскому.

**Пакеты**

**61**

Классы из задач по разработке ООП перенесите в пакет  
ru.kfpu.itis.group801.вашафамилия.basicClasses  
Классы из задач 60-66 перенесите в пакет  
ru.kfpu.itis.group801. вашафамилия.complexClasses  
Создать класс Main с main методом, где собран код всех main-методов написанных вами классов. Его поместите в пакет  
ru.kfpu.itis.group801.вашафамилия  
Скомпилируйте и запустите Main.

**62**

**Групповое задание по структуре БД соцсети**

**63**

**5 запросов к соцсетям**