PROBLEM SET #3

Организация файлов

В репозитории создать папку problem-set-3. Для каждой задачи создавать папку Task**XX**PS**N**, где XX - двузначный номер задачи (если номер задачи 1, то ее номер - 01), а N - номер problem-set. Внутри папки все нужные для задачи классы.

Внутри файла самыми первым строчкам в комментариях код должен быть вам подписан по образцу.

```
/**
* @author Mikhail Abramskiy
* 09-53a
* Problem Set 3 Task 01 (для вспомогательного класса
указывайте для чего используется, например for Problem
Set 1 Task 01)
*/
```

Дедлайн сдачи Problem Set 3 – 24 декабря 2017 года, 10:00.

Количество баллов за Problem Set 3: 8 (восемь).

01

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.

02

В задаче 01 добавить вероятностное распределение – сила удара обратна пропорциональна вероятности попасть. Можно выполнить эту задачу полностью в папке этого задания, тогда 01 будет зачтена.

Во следующих задачах реализовать класс и протестировать все его методы в методе main.

03

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.
- Vector2D sub(Vector2D) - вычитание из вектора другого вектора, результат возвращается как новый объект;
- Vector2D mult(double) - умножение вектора на вещественное число, результат возвращается как новый объект;
- String toString() - строковое представление вектора;
- double length() – длина (модуль) вектора;
- double scalarProduct(Vector2D) - скалярное произведение вектора на другой вектор;

04

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. При передаче в конструктор знаменателя = 0 выбросьте исключение Arithmetic Exception.
- void reduce() - сокращение дроби;
- RationalFraction add(RationalFraction) - сложение дроби с другой дробью, результат возвращается как новый объект (не забудьте сократить)
- RationalFraction sub(RationalFraction) - вычитание из дроби другой дроби, результат возвращается как новый объект (не забудьте сократить);
- RationalFraction mult(RationalFraction) - умножение дроби на другую дробь, результат возвращается как новый объект (сократить)
- RationalFraction div(RationalFraction) - деление дроби на другую дробь, результат возвращается как новый объект (сократить)

- String toString() - строковое представление дроби (например, -2/3);
- double value() - десятичное значение дроби;

05

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;
- Matrix2x2 sub(Matrix2x2) - вычитание из матрицы другой матрицы;
- Matrix2x2 mult (double) - умножение матрицы на вещественное число;
- Matrix2x2 mult(Matrix2x2) - умножение матрицы на другую матрицу;
- double det() - определитель матрицы;
- void transpose() - транспонировать матрицу;
- Matrix2x2 inverseMatrix() - возвратить обратную матрицу для заданной. Если это невозможно, выбросить исключение MatrixException (реализуйте как класс-наследник Exception).
- Vector2D multVector(Vector2D) - умножить матрицу на двумерный вектор (считая его столбцом) и возвратить получившийся столбец в виде вектора.

А теперь комбинируем! ^^

Все вспомогательные операции над компонентами должны опираться на операции, написанные в классах 03-05.

06

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;

07

Создать класс RationalMatrix2x2 - двумерная матрица из RationalFraction. Аргументы - содержимое матрицы. Методы:

- RationalMatrix2x2() - конструктор для нулевой матрицы;
- RationalMatrix2x2(RationalFration) - конструктор для матрицы, у которой каждый элемент равен поданному числу;
- 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 возвратить получившийся столбец в виде вектора.

08

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

09-10

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.

11

В выполняемом проекте (командном) найти/придумать пример полиморфизма и реализовать его: должен быть абстрактный класс или интерфейс, два наследника / два реализующих интерфейс класса. Должен создавать массив, в который добавляет объекты обоих классов, а затем у них вызывается один и тот же метод, демонстрируя позднее связывание (пример как в лекции или как на паре).

12

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:
  - Number add(Number n) – прибавить целое число n
  - 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)
  - 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).