**Задача** Range**.**

**Лекции, нужные для решения задачи:** все лекции с курса основ.

**Некоторые моменты:**

* Эту задачу для простоты делаем без epsilon
* Для простоты считаем, что начало всегда меньше конца

**Часть 1**. Создать класс Range (непрерывный вещественный числовой диапазон на прямой). В нём:

1. Объявить два вещественных поля from, to
2. Описать конструктор, при помощи которого заполняются поля
3. Реализовать геттеры и сеттеры для полей
4. Сделать метод для получения длины диапазона
5. Сделать метод isInside, который принимает вещественное число и возвращает boolean – результат проверки, принадлежит ли число диапазону

После этого написать небольшую программу с использованием этого класса.

Java - любой класс должен лежать в некотором пакете.  
C# - вместо полей, геттеров и сеттеров принято делать автосвойства.

**Часть 2 (Range\*)**. Доработать класс Range, реализовать методы:

* Получение интервала-пересечения двух интервалов. Если пересечения нет, выдать null.   
  Если есть, то выдать новый диапазон с соответствующими концами
* Получение объединения двух интервалов.  
  Может получиться 1 или 2 отдельных куска
* Получение разности двух интервалов.  
  Может получиться 0, 1 или 2 отдельных куска

В операциях, где может получиться 2 куска, нужно выдавать массив объектов Range.

Операции пересечения, объединения и разности – по смыслу такие же как для множеств, см. литературу по множествам.  
Разность нужна несимметричная – из первого интервала вычитаем второй.

Эти методы нужно сделать нестатическими.

В пересечении и разности если диапазоны пересекаются только по 1 концу, считаем, что пересечения нет.  
В объединении что есть.  
Так нужно сделать, чтобы результаты получались более логичными, т.к. в этой задаче мы не учитываем входит конец в диапазон или нет.

В main написать программу для проверки этих методов.

Чему научитесь:

* Создание простого класса с полями, конструктором, геттерами, сеттерами, методами
* Работа с this
* Создание пакетов
* Понимание операций теории множеств – пересечение, объединение, разность

**Задача 1**. Фигуры

**Часть 1.** Создать иерархию классов для геометрических фигур.

**Лекции, нужные для решения задачи:** 1-5.

Для них нужно создать интерфейс Shape (IShape в C#), в котором определены методы:

double getWidth()  
double getHeight()  
double getArea()  
double getPerimeter()

В C# имена методов с заглавной буквы.

Все фигуры должны реализовывать этот интерфейс и правильным образом определять данные методы

1. Square – квадрат

Должен иметь конструктор, принимающий длину стороны

1. Triangle – треугольник

Должен иметь конструктор, принимающий x1, y1, x2, y2, x3, y3 – шесть координат.  
В качестве ширины возвращать max(x1, x2, x3) – min(x1, x2, x3)  
В качестве высоты возвращать max(y1, y2, y3) – min(y1, y2, y3)

1. Rectangle – прямоугольник

Должен иметь конструктор, принимающий длины двух сторон

1. Circle – окружность

Должна иметь конструктор, принимающий радиус. В качестве ширины и высоты должен выдаваться диаметр

**Часть 2.** В main в коде объявить массив фигур, чтобы в нём было около 5-10 разных фигур. Задача – написать функцию, которая находит фигуру с максимальной площадью. Вызвать её для этого массива и распечатать информацию о фигуре в консоль.

Аналогично найдите фигуру со вторым по величине периметром.

Поиск фигур реализовать через стандартный метод Arrays.sort (в C# Array.Sort) с компаратором. Что такое компаратор почитайте сами, но если будут вопросы, задавайте.

**Часть 3.** Переопределите в фигурах методы toString, hashCode, equals (в C# - ToString, GetHashCode, Equals)

Чему научитесь:

* Интерфейсы, реализация интерфейса
* Виртуальные функции – как они работают и применяются
* Реализация toString, hashCode, equals
* Применение стандартной сортировки, компараторы, реализация интерфейсов

**Задача 2.** Вектор

**Лекции, нужные для решения задачи:** 1-6.

Реализовать класс Vector для векторов вещественных чисел размерности n. Компоненты вектора хранить массивом. Номера компонент отсчитываются от нуля

Конструктор вектора должен принимать число n – размерность вектора. Если n <= 0, то нужно бросать исключение IllegalArgumentException (ArgumentException в C#)

Для этого класса нужно реализовать:

1. Конструкторы
   1. Vector(n) – размерность n, все компоненты равны 0
   2. Vector(Vector) – конструктор копирования
   3. Vector(double[]) – заполнение вектора значениями из массива
   4. Vector(n, double[]) – заполнение вектора значениями из массива. Если длина массива меньше n, то считать что в остальных компонентах 0
2. Метод getSize() для получения размерности вектора
3. Реализовать метод toString(), чтобы выдавал информацию о векторе в формате { значения компонент через запятую }  
   Например, {1, 2, 3}
4. Реализовать нестатические методы:
   1. Прибавление к вектору другого вектора
   2. Вычитание из вектора другого вектора
   3. Умножение вектора на скаляр
   4. Разворот вектора (умножение всех компонент на -1)
   5. Получение длины вектора
   6. Получение и установка компоненты вектора по индексу
   7. Переопределить метод equals, чтобы был true ⬄ векторы имеют одинаковую размерность и соответствующие компоненты равны. Соответственно, переопределить hashCode
5. Реализовать статические методы:
   1. Сложение двух векторов – должен создаваться новый вектор
   2. Вычитание векторов – должен создаваться новый вектор
   3. Скалярное произведение векторов

Если операции выполняются над векторами разной размерности, то считать что у меньшего вектора в недостающих компонентах нули

В main написать небольшую тестовую программу

Чему научитесь:

* Реализация достаточно сложного класса с многими операциями
* Тестирование за собой
* Поймете, что методы зачастую независимы друг от друга, и их можно писать и проверять по-отдельности. Но в некоторых случаях методы должны быть согласованы между собой
* Начальные навыки работы с исключениями
* Часто надежнее делать копию данных, чтобы не было неожиданностей со ссылочными типами

**Задача 3.** Матрица

**Лекции, нужные для решения задачи:** 1-6.

Реализовать класс матрицы Matrix с использованием класса Vector – хранить строки как массив векторов.

Во всех методах, кроме конструкторов, если размеры входных данных неверные, то кидать исключение. В конструкторах нужно добить данные нулями до нужной длины.

Реализовать:

1. Конструкторы:
   1. Matrix(n, m) – матрица нулей размера nxm
   2. Matrix(Matrix) – конструктор копирования
   3. Matrix(double[][]) – из двумерного массива (в C# double[,])
   4. Matrix(Vector[]) – из массива векторов-строк
2. Методы:
   1. Получение размеров матрицы
   2. Получение и задание вектора-строки по индексу
   3. Получение вектора-столбца по индексу
   4. Транспонирование матрицы
   5. Умножение на скаляр
   6. Вычисление определителя матрицы
   7. toString определить так, чтобы результат получался в таком виде: {{1, 2}, {2, 3}}
   8. умножение матрицы на вектор
   9. Сложение матриц
   10. Вычитание матриц
3. Статические методы:
   1. Сложение матриц
   2. Вычитание матриц
   3. Умножение матриц

Чему научитесь:

* Создание класса на основе другого своего класса
* Некоторые операции здесь сложны алгоритмически