Лабораторная работа №1

Автор: Терин Ярослав

Группа: 6203-010302D

Задание 1 и 2

Создали проект. В котором отдельно создали пакет “functions”. В нем внутри создали класс “Function Point”. Внутри хранятся два поля: x и y. которые приватные по причине инкапсуляции. Хранящие числа с плавающими запятыми.

Так же для инициализации сделали 3 конструктора, первый конструктор копирования, который создает, еще такой объект который подали на вход. Второй конструктор задает координаты по умолчанию - 0 для x и 0 для y. Третий конструктор уже создавал объект с входящими значениями.

Для получения доступа к координатам были добавлены отдельные методы - “getX”и “getY”.

Задание 3

Здесь мы создали новый класс, который должен будет являться шаблоном для создания табулированный функции. В нем тоже создали два конструктора но с разными параметрами.

Первый из которых имел в виде параметров только границы данной функции - “leftx” и “rightx” и количество точек. В таком случае, по заданию нужно было распределить все точки x на равном от друг друга расстоянии внутри этих границ. Для этого мы вычислил интервал, на который должны быть отделены точки. Это производилось по формуле |rightX-leftX|/(pointsCount-1), где “pointsCount” это количество точек. Для хранение точек был создан массив из объектов “FunctionPoint” Потом, массив заполнялся x-ами в соответствии с этим интервалом через цикл создавая для каждой новой точки объект. Все значения у заполнились нулями.

Для второго часть параметров осталось не измененной но для точки у в качестве аргумента был массив из значений с плавающей запятой. Алгоритм заполнения остался тем же, но внутрь конструктора в качестве у уже передавались значения из переданного массива. Для вычисления интервала пришлось указать, что длина исходного массива точек, это длина переданного массива. С помощью стандартного метода “.length”.

Задание 4

В классе “Tabulated Function” мы начали добавлять методы. Добавили два метода для получения границ функции. “getLeftDomainBorder” и “getRightDomainBorder”. Возвращения значений границ осуществлялось через массив и возвращало 0-й или предпоследний элемент, предпоследний так как индексация начинается с нуля.

Теперь планировалось написать метод, который будет вычислять значения у-ка вне зависимости от существование подходящего к немцу х-а в массиве точек “getFunctionValue”. Для этого использовалась формула линейной интерполяции. Которая на вход принимала 4 координата, 2 х-а и 2 у-а и точку х а на выход значения точки у в этом х. Для начало нужно было определить индекс данного х-а. Индекс был найден через цикл с условием диапазона х-а. Предварительно метод сразу возвращал нужный у, когда х уже был в реестре функции. Так как для формулы нужно было два индекса, в начале была добавлена еще одна переменная (“второй индекс”). Потом применили формулы и нашли нужный у, для работы со значениями используя массив и “геттеры”.

Задание 5

Далее реализуем еще несколько методов описание их ниже.

Метод “PointsGetCount” возвращает количество точек в массиве функции

Метод “getPoint” - На вход принимаем объект точка, внутри создаем новый объект и возвращает точную копию объекта точка.

Метод “setPoint” - Заменяем определенную точку по переданному индексу. Предварительно проверяя, входит ли точка в заданные нами границы. Для этого сравниваем полученную точку с соседями - если индекс в точке не последний значит у нее должна существовать соседняя точка больше ее по x. Тоже самое условие проверяем с другой стороны.

Метод “getPointX” - Метод табулированный функции возвращает x, пользуйся методом класса “Function Point” “гетором” x.

Метод “getPointY” - Делает тоже самое что и метод сверху только для y.

Метод “setPointY” - Заменяем или создаем новую точку с определенным y, используя все прошлые методы и конструкторы. При этом не делаем не какой дополнительной проверки для y-ка как в “setPoint”.

Задание 6

По аналогии с 5м заданием ниже описаны еще методы

Метод “deletePoint” удаляем нужную точку снова по индексу. Для этого с начало проверяем индекс на нужные границы, делаем это по индексу. Если индекс оказывается внутри границ присваиваем этому элементу в массиве значения “null”, таким образом стирая его. Затем в цикле переносим все элементы от индекса на один вперед. Таким образом в итоге в конце формируяться два одинаковых элемента один из которых мы тоже удаляем. Последнее уменьшаем длину.

Метод “addPoint” должен добавлять точку, которая подается на вход. От “setPoint” отличается тем, что должна создаться новая точка, а не заменится старая. С начало опять вычисляем индекс точки для правильного упорядочения x по возрастанию. Предварительно делая проверку для определения, если данная точка больше всех присваивая индексу длину массива. Потом для сдвига всех точек после и до место будущего расположения новый точки, создаем временный массив. Следующим шагом переносим все точки до позиции вставки, используя стандартную функцию “arraycopy”. Затем вставляем новую точку и переписываем оставшиеся точки. Теперь у нас есть два массива один с добавленной точки дугой без, присваиваем временный массив оригинальному массиву, таким образом стирая временный. Под конец увеличиваем количество точек.

Задание 7

Здесь скомпонуем программу в “main”. Импортируем два написанных класса - “Tabulated Function” и “Function Point”. Проверяем работу двух конструкторов. С начало первого на нем мы проверим все написанные функции. Для выбора функции я взял y = 5x, объявив единичную функцию через конструктор в цикле присвоил значения у-ку x*5. На фото ниже представлена работа методов функции. В конце мы создаем еще одну функцию через 2-й конструктор, используя массив из чисел плавающий запятой. Демонстрация тоже на фото.

```
Создали функцию через первый конструктор (y=5x)
X: 1.0 Y: 5.0
X: 2.0 Y: 10.0
X: 3.0 Y: 15.0
X: 4.0 Y: 20.0
X: 5.0 Y: 25.0
X: 6.0 Y: 30.0
X: 7.0 Y: 35.0
X: 8.0 Y: 40.0
X: 9.0 Y: 45.0
X: 10.0 Y: 50.0

22.5 Функция определения значения в произвольной точке 4.5

Добавили y=20 по индексу 3
X: 1.0 Y: 5.0
X: 2.0 Y: 10.0
X: 3.0 Y: 15.0
X: 4.0 Y: 80.0
X: 5.0 Y: 25.0
X: 6.0 Y: 30.0
X: 7.0 Y: 35.0
X: 8.0 Y: 40.0
X: 9.0 Y: 45.0
X: 10.0 Y: 50.0

10 Количество точек
```

```
Добвавили точку test_point 8.5, 17
X: 1.0 Y: 5.0
X: 2.0 Y: 10.0
X: 3.0 Y: 15.0
X: 4.0 Y: 80.0
X: 5.0 Y: 25.0
X: 6.0 Y: 30.0
X: 7.0 Y: 35.0
X: 8.0 Y: 40.0
X: 8.5 Y: 17.0
X: 9.0 Y: 45.0
X: 10.0 Y: 50.0

Удалили элемент с индексом 9
X: 1.0 Y: 5.0
X: 2.0 Y: 10.0
X: 3.0 Y: 15.0
X: 4.0 Y: 80.0
X: 5.0 Y: 25.0
X: 6.0 Y: 30.0
X: 7.0 Y: 35.0
X: 8.0 Y: 40.0
X: 8.5 Y: 17.0
X: 10.0 Y: 50.0
```

```
Проверка второго конструктора
X: 1.0 Y: 4.4
X: 1.1111111111111112 Y: 5.5
X: 1.2222222222222223 Y: 6.6
X: 1.3333333333333333 Y: 7.7
X: 1.4444444444444444 Y: 8.8
X: 1.5555555555555554 Y: 9.9
X: 1.6666666666666665 Y: 10.0
```