Лабораторная работа №2

Задание на лабораторную работу

Разработать набор классов для работы с функциями одной переменной, заданными в табличной форме.

В данной работе можно не делать проверки корректности параметров методов, если только задание не требует этого в явной форме. Необходимые для этого модификации будут выполняться в следующей работе.

В ходе выполнения работы запрещено использовать классы из пакета `java.util`.

Задание 1 *(0,1 балла)*

Создать пакет `functions`, в котором далее будут создаваться классы программы.

Задание 2 *(0,7 балла)*

В пакете `functions` создать класс `FunctionPoint`, объект которого должен описывать одну точку табулированной функции.

Состояние объектов должно содержать два аспекта: координату точки по оси абсцисс и координату точки по оси ординат. При написании класса следует учесть особенности инкапсуляции.

В классе должны быть описаны следующие конструкторы:

- `FunctionPoint(double x, double y)` – создаёт объект точки с заданными координатами;
- `FunctionPoint(FunctionPoint point)` – создаёт объект точки с теми же координатами, что у указанной точки;
- `FunctionPoint()` – создаёт точку с координатами (0; 0).

Задание 3 *(0,7 балла)*

В пакете `functions` создать класс `TabulatedFunction`, объект которого должен описывать табулированную функцию.

Для хранения данных о точках должен использоваться массив типа `FunctionPoint`. При этом разумно организовать работу с массивом так, чтобы точки в нём были всегда упорядочены по значению координаты *x*.

В классе должны быть описаны следующие конструкторы:

`TabulatedFunction(double leftX, double rightX, int pointsCount)` – создаёт объект табулированной функции по заданным левой и правой границе области определения, а также количеству точек для табулирования (значения функции в точках при этом следует считать равными 0);

`TabulatedFunction(double leftX, double rightX, double[] values)` – аналогичен предыдущему конструктору, но вместо количества точек получает значения функции в виде массива.

В обоих случаях точки должны создаваться через равные интервалы по *x*.

Задание 4 *(0,6 балла)*

В классе `TabulatedFunction` описать методы, необходимые для работы с функцией.

Метод `double getLeftDomainBorder()` должен возвращать значение левой границы области определения табулированной функции. Очевидно, что оно совпадает с абсциссой самой левой точки в описывающей функцию таблице.

Аналогично, метод `double getRightDomainBorder()` должен возвращать значение правой границы области определения табулированной функции.

Метод `double getFunctionValue(double x)` должен возвращать значение функции в точке *x*, если эта точка лежит в области определения функции. В противном случае метод должен возвращать значение неопределённости (оно хранится, например, в поле `NaN` класса `Double`). При расчёте значения функции следует использовать линейную интерполяцию, т.е. считать, что на интервале между заданными в таблице точками функция является прямой линией. Для написания кода метода рекомендуется воспользоваться уравнением прямой, проходящей через две заданные различающиеся точки.

Задание 5 *(1,5 балла)*

В классе `TabulatedFunction` описать методы, необходимые для работы с точками табулированной функции. Считать, что нумерация точек начинается с ноля.

Метод `int getPointsCount()` должен возвращать количество точек.

Метод `FunctionPoint getPoint(int index)` должен возвращать ссылку на объект, описывающий точку, имеющую указанный номер. При написании метода обеспечьте корректную инкапсуляцию.

Метод `void setPoint(int index, FunctionPoint point)` должен заменять указанную точку табулированной функции на заданную. При написании метода обеспечьте корректную инкапсуляцию. В случае если координата *x* задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции, то замену точки проводить не следует. Например, для функции, определяемой точками {(0; 0), (1; 1), (2; 4)}, точку с индексом 1 нельзя заменить точкой (-1; 5).

Метод `double getPointX(int index)` должен возвращать значение абсциссы точки с указанным номером.

Метод `void setPointX(int index, double x)` должен изменять значение абсциссы точки с указанным номером. Аналогично методу `setPoint()`, данные метод не должен изменять точку, если новое значение попадает в другой интервал табулирования.

Метод `double getPointY(int index)` должен возвращать значение ординаты точки с указанным номером.

Метод void `setPointY(int index, double y)` должен изменять значение ординаты точки с указанным номером.

Задание 6 *(1 балл)*

В классе `TabulatedFunction` описать методы, изменяющие количество точек табулированной функции.

Метод `void deletePoint(int index)` должен удалять заданную точку табулированной функции.

Метод `void addPoint(FunctionPoint point)` должен добавлять новую точку табулированной функции. При написании метода обеспечьте корректную инкапсуляцию.

При написании методов следует учитывать, что точки в массиве должны быть упорядочены по значению координаты *x*.

Для копирования участков массивов следует воспользоваться методом `arraycopy()` класса `System`.

Также следует понимать, что создание нового массива каждый раз при выполнении операций удаления и вставки точки является расточительством по отношению к памяти и скорости работы программы. Поэтому длина массива в общем случае не должна совпадать с количеством точек в табулированной функции, а замена массива на массив большей длины должна производиться только в некоторых случаях.

Задание 7 *(0,4 балла)*

Проверить работу написанных классов.

В пакете по умолчанию (вне пакета `functions`) нужно создать класс `Main`, содержащий точку входа программы.

В методе `main()` создайте экземпляр класса `TabulatedFunction` и задайте для него табулированные значения какой-нибудь известной вам функции.

Выведите в консоль значения функции на ряде точек. Рекомендуется использовать такой шаг (или такие точки), чтобы среди них оказались точки вне области определения функции, а также чтобы несколько точек попали в один интервал табулированной функции.

Проверьте, как изменяется результат работы программы после изменения точек, добавления и удаления точек.