

Отчёт

Лабораторная работа №4

Выполнил: Салихов Камран Рустам угли

Группа: 6204-010302D

В ходе выполнения задания 1 в классах **ArrayTabulatedFunction** и **LinkedListTabulatedFunction** были добавлены конструкторы, получающие сразу все точки функции в виде массива объектов типа **FunctionPoint**. Если точек задано меньше двух, или если точки в массиве не упорядочены по значению абсциссы, конструкторы выбрасывают исключение **IllegalArgumentException**. В обоих случаях обеспечена корректная инкапсуляция.

В ходе выполнения задания 2 в пакете **functions** был создан интерфейс **Function**, описывающий функции одной переменной и содержащий следующие методы:

public double getLeftDomainBorder() – возвращает значение левой границы области определения функции;

public double getRightDomainBorder() – возвращает значение правой границы области определения функции;

public double getFunctionValue(double x) – возвращает значение функции в заданной точке.

Были исключены соответствующие методы из интерфейса **TabulatedFunction**, а сам **TabulatedFunction** был изменён для наследования от **Function**:

public interface TabulatedFunction extends Function

В ходе выполнения задания 3 был создан пакет **functions.basic**, в котором были описаны классы ряда функций, заданных аналитически.

В пакете был создан публичный класс **Exp**, объекты которого вычисляют значение экспоненты. Класс реализует интерфейс **Function**.

Аналогично, был создан класс **Log**, объекты которого вычисляют значение логарифма по заданному основанию. Основание передается как параметр конструктора.

Был создан класс **TrigonometricFunction**, реализующий интерфейс **Function** и описывающий методы получения границ области определения.

Были созданы наследующие от него публичные классы **Sin**, **Cos** и **Tan**, объекты которых вычисляют значения синуса, косинуса и тангенса.

В ходе выполнения задания 4 был создан пакет **functions.meta**, в котором были описаны классы функций, позволяющие комбинировать функции.

Был создан класс **Sum**, объекты которого представляют собой функции, являющиеся суммой двух других функций. Класс реализует интерфейс **Function**.

Аналогично, был создан класс **Mult**, объекты которого представляют собой функции, являющиеся произведением двух других функций.

Был создан класс **Power**, объекты которого представляют собой функции, являющиеся степенью другой функции.

Был создан класс **Scale**, объекты которого описывают функции, полученные из исходных функций путём масштабирования вдоль осей координат. Конструктор класса получает ссылку на объект исходной функции, а также коэффициенты масштабирования вдоль оси абсцисс и оси ординат.

Аналогично, был создан класс **Shift**, объекты которого описывают функции, полученные из исходных функций путём сдвига вдоль осей координат.

Был создайте класс **Composition**, объекты которого описывают композицию двух исходных функций. Конструктор класса получает ссылки на объекты первой и второй функции.

В ходе выполнения задания 5 в пакете **functions** был создан класс **Functions**, содержащий вспомогательные статические методы для работы с функциями. Класс содержит следующие методы:

public static Function shift(Function f, double shiftX, double shiftY) – возвращает объект функции, полученной из исходной сдвигом вдоль осей;

public static Function scale(Function f, double scaleX, double scaleY) – возвращает объект функции, полученной из исходной масштабированием вдоль осей;

public static Function power(Function f, double power) – возвращает объект функции, являющейся заданной степенью исходной;

public static Function sum(Function f1, Function f2) –

возвращает объект функции, являющейся суммой двух исходных;

public static Function mult(Function f1, Function f2) –

возвращает объект функции, являющейся произведением двух исходных;

public static Function composition(Function f1, Function f2) –

возвращает объект функции, являющейся композицией двух исходных.

При написании методов были использованы созданные ранее классы из пакета **functions.meta**.

Был реализован приватный конструктор для запрета создания объектов:

private Functions() {}

В ходе выполнения задания 6 в пакете **functions** был создан класс **TabulatedFunctions**, содержащий вспомогательные статические методы для работы с табулированными функциями. Был реализован приватный конструктор для запрета создания объектов:

private TabulatedFunctions() {}

В классе был описан метод **public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount)**, получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек. Если указанные границы для табулирования выходят за область определения функции, метод выбрасывает исключение **IllegalArgumentException**.

Возвращает объект **ArrayTabulatedFunction** с полученными точками.

В ходе выполнения задания 7 в класс **TabulatedFunctions** были добавлены следующие методы.

Метод вывода табулированной функции в байтовый поток
**public static void
outputTabulatedFunction(TabulatedFunction function,
OutputStream out)** выводит значения в указанный поток, по которым потом восстанавливается табулированная функция (количество точек в ней и значения координат точек).

Метод ввода табулированной функции из байтового потока
**public static TabulatedFunction
inputTabulatedFunction(InputStream in)** считывает из указанного потока данные о табулированной функции, создаёт и настраивает её объект и возвращать его из метода.

Метод записи табулированной функции в символьный поток
**public static void writeTabulatedFunction(TabulatedFunction
function, Writer out)** выводит значения в указанный поток, по которым потом восстанавливается табулированная функция (количество точек в ней и значения координат точек).

Метод чтения табулированной функции из символьного потока
**public static TabulatedFunction
readTabulatedFunction(Reader in)** считывает из указанного потока данные о табулированной функции, создаёт и настраивает её объект и возвращать его из метода.

Методы **TabulatedFunctions** пробрасывают **IOException**, потому что только вызывающий код знает, как обработать ошибку в конкретном контексте.

Потоки не закрываются внутри методов, так как управление ресурсами остаётся за вызывающим кодом. Используется **flush()** для потоков вывода, который гарантирует, что всё записано в поток.

В ходе выполнения задания 8 была проверена работа написанных классов. Далее текст работы с консолью:

вывод Sin и Cos на отрезке от 0 до π с шагом 0.1:

$F(0.0) = 0.0$

$F(0.1) = 0.09983341664682815$

$F(0.2) = 0.19866933079506122$

$F(0.30000000000000004) = 0.2955202066613396$

$F(0.4) = 0.3894183423086505$

$F(0.5) = 0.479425538604203$

$F(0.6) = 0.5646424733950354$

$F(0.7) = 0.644217687237691$

$F(0.7999999999999999) = 0.7173560908995227$

$F(0.8999999999999999) = 0.7833269096274833$

$F(0.9999999999999999) = 0.8414709848078964$

$F(1.0999999999999999) = 0.8912073600614353$

$F(1.2) = 0.9320390859672263$

$F(1.3) = 0.963558185417193$

$F(1.4000000000000001) = 0.9854497299884603$

$F(1.5000000000000002) = 0.9974949866040544$

$F(1.6000000000000003) = 0.9995736030415051$
 $F(1.7000000000000004) = 0.9916648104524686$
 $F(1.8000000000000005) = 0.973847630878195$
 $F(1.9000000000000006) = 0.9463000876874142$
 $F(2.0000000000000004) = 0.9092974268256815$
 $F(2.1000000000000005) = 0.8632093666488735$
 $F(2.2000000000000006) = 0.8084964038195899$
 $F(2.3000000000000007) = 0.7457052121767197$
 $F(2.4000000000000001) = 0.6754631805511503$
 $F(2.5000000000000001) = 0.5984721441039558$
 $F(2.6000000000000001) = 0.5155013718214634$
 $F(2.7000000000000001) = 0.42737988023382895$
 $F(2.8000000000000001) = 0.33498815015590383$
 $F(2.9000000000000002) = 0.23924932921398112$
 $F(3.0000000000000003) = 0.1411200080598659$
 $F(3.1000000000000004) = 0.04158066243328916$
 $F(0.0) = 1.0$
 $F(0.1) = 0.9950041652780258$
 $F(0.2) = 0.9800665778412416$
 $F(0.30000000000000004) = 0.955336489125606$
 $F(0.4) = 0.9210609940028851$
 $F(0.5) = 0.8775825618903728$
 $F(0.6) = 0.8253356149096783$
 $F(0.7) = 0.7648421872844885$
 $F(0.7999999999999999) = 0.6967067093471655$
 $F(0.8999999999999999) = 0.6216099682706645$
 $F(0.9999999999999999) = 0.5403023058681398$
 $F(1.0999999999999999) = 0.4535961214255775$

$F(1.2) = 0.3623577544766736$
 $F(1.3) = 0.26749882862458735$
 $F(1.4000000000000001) = 0.16996714290024081$
 $F(1.5000000000000002) = 0.07073720166770268$
 $F(1.6000000000000003) = -0.029199522301289037$
 $F(1.7000000000000004) = -0.12884449429552508$
 $F(1.8000000000000005) = -0.22720209469308753$
 $F(1.9000000000000006) = -0.32328956686350396$
 $F(2.0000000000000004) = -0.4161468365471428$
 $F(2.1000000000000005) = -0.5048461045998579$
 $F(2.2000000000000006) = -0.5885011172553463$
 $F(2.3000000000000007) = -0.6662760212798248$
 $F(2.4000000000000001) = -0.737393715541246$
 $F(2.5000000000000001) = -0.8011436155469343$
 $F(2.6000000000000001) = -0.8568887533689478$
 $F(2.7000000000000001) = -0.9040721420170617$
 $F(2.8000000000000001) = -0.9422223406686585$
 $F(2.90000000000000012) = -0.9709581651495908$
 $F(3.00000000000000013) = -0.9899924966004456$
 $F(3.10000000000000014) = -0.9991351502732795$

вывод табулированных Sin и Cos на отрезке от 0 до π с шагом 0.1:

Tabulated $F(0.0) = 0.0$
 Tabulated $F(0.1) = 0.09798155360510165$
 Tabulated $F(0.2) = 0.1959631072102033$
 Tabulated $F(0.30000000000000004) = 0.29394466081530496$
 Tabulated $F(0.4) = 0.38590680571121505$
 Tabulated $F(0.5) = 0.47207033789781927$

Tabulated $F(0.6) = 0.5582338700844235$
Tabulated $F(0.7) = 0.6439824415274444$
Tabulated $F(0.7999999999999999) = 0.707935358675545$
Tabulated $F(0.8999999999999999) = 0.7718882758236456$
Tabulated $F(0.9999999999999999) = 0.8358411929717461$
Tabulated $F(1.0999999999999999) = 0.8839933571281424$
Tabulated $F(1.2) = 0.9180219935851436$
Tabulated $F(1.3) = 0.9520506300421447$
Tabulated $F(1.4000000000000001) = 0.984807753012208$
Tabulated $F(1.5000000000000002) = 0.984807753012208$
Tabulated $F(1.6000000000000003) = 0.984807753012208$
Tabulated $F(1.7000000000000004) = 0.984807753012208$
Tabulated $F(1.8000000000000005) = 0.966204042925035$
Tabulated $F(1.9000000000000006) = 0.932175406468034$
Tabulated $F(2.0000000000000004) = 0.8981467700110329$
Tabulated $F(2.1000000000000005) = 0.8624409082617227$
Tabulated $F(2.2000000000000006) = 0.7984879911136221$
Tabulated $F(2.3000000000000007) = 0.7345350739655215$
Tabulated $F(2.4000000000000001) = 0.670582156817421$
Tabulated $F(2.5000000000000001) = 0.594071569547527$
Tabulated $F(2.6000000000000001) = 0.5079080373609227$
Tabulated $F(2.7000000000000001) = 0.42174450517431844$
Tabulated $F(2.8000000000000001) = 0.3346977889881713$
Tabulated $F(2.9000000000000012) = 0.23671623538306957$
Tabulated $F(3.0000000000000013) = 0.1387346817779678$
Tabulated $F(3.1000000000000014) = 0.04075312817286608$
Tabulated $F(0.0) = 1.0$
Tabulated $F(0.1) = 0.9827232084876878$

Tabulated $F(0.2) = 0.9654464169753757$
Tabulated $F(0.30000000000000004) = 0.9481696254630635$
Tabulated $F(0.4) = 0.914354644443779$
Tabulated $F(0.5) = 0.864608105941514$
Tabulated $F(0.6) = 0.8148615674392491$
Tabulated $F(0.7) = 0.7646204979800333$
Tabulated $F(0.7999999999999999) = 0.6884043792119047$
Tabulated $F(0.8999999999999999) = 0.6121882604437761$
Tabulated $F(0.9999999999999999) = 0.5359721416756473$
Tabulated $F(1.0999999999999999) = 0.45063345391435955$
Tabulated $F(1.2) = 0.35714054363391856$
Tabulated $F(1.3) = 0.26364763335347763$
Tabulated $F(1.4000000000000001) = 0.16993052093895483$
Tabulated $F(1.5000000000000002) = 0.07043744393442489$
Tabulated $F(1.6000000000000003) = -0.029055633070105086$
Tabulated $F(1.7000000000000004) = -0.12854871007463503$
Tabulated $F(1.8000000000000005) = -0.22476145104951817$
Tabulated $F(1.9000000000000006) = -0.3182543613299591$
Tabulated $F(2.0000000000000004) = -0.41174727161039987$
Tabulated $F(2.1000000000000005) = -0.5042718354168345$
Tabulated $F(2.2000000000000006) = -0.5804879541849632$
Tabulated $F(2.3000000000000007) = -0.656704072953092$
Tabulated $F(2.4000000000000001) = -0.7329201917212208$
Tabulated $F(2.5000000000000001) = -0.7941706620070894$
Tabulated $F(2.6000000000000001) = -0.8439172005093544$
Tabulated $F(2.7000000000000001) = -0.8936637390116193$
Tabulated $F(2.8000000000000001) = -0.9409837494179168$
Tabulated $F(2.9000000000000012) = -0.958260540930229$

Tabulated $F(3.00000000000000013) = -0.9755373324425413$

Tabulated $F(3.10000000000000014) = -0.9928141239548535$

вывод суммы квадратов на отрезке от 0 до π с шагом 0.1:

$F(0.0) = 1.0$

$F(0.1) = 0.9753452893472049$

$F(0.2) = 0.9704883234380686$

$F(0.30000000000000004) = 0.9854291022725908$

$F(0.4) = 0.9849684785101429$

$F(0.5) = 0.9703975807827336$

$F(0.6) = 0.975624427798983$

$F(0.7) = 0.9993578909268825$

$F(0.7999999999999999) = 0.9750730613812004$

$F(0.8999999999999999) = 0.9705859765791769$

$F(0.9999999999999999) = 0.9858966365208117$

$F(1.0999999999999999) = 0.9845147652334687$

$F(1.2) = 0.9703137486131723$

$F(1.3) = 0.9759104767365345$

$F(1.4000000000000001) = 0.9987226923395387$

$F(1.5000000000000002) = 0.9748077439009694$

$F(1.6000000000000003) = 0.9706905402060587$

$F(1.7000000000000004) = 0.9863710812548067$

$F(1.8000000000000005) = 0.9840679624425679$

$F(1.9000000000000006) = 0.9702368269293845$

$F(2.0000000000000004) = 0.9762034361598597$

$F(2.1000000000000005) = 0.9980944042379682$

$F(2.2000000000000006) = 0.9745493369065118$

$F(2.3000000000000007) = 0.9708020143187142$

$F(2.4000000000000001) = 0.9868524364745752$
 $F(2.5000000000000001) = 0.9836280701374409$
 $F(2.6000000000000001) = 0.9701668157313703$
 $F(2.7000000000000001) = 0.9765033060689583$
 $F(2.8000000000000001) = 0.9974730266221714$
 $F(2.9000000000000012) = 0.974297840397828$
 $F(3.0000000000000013) = 0.9709203989171432$
 $F(3.1000000000000014) = 0.9873407021801173$

Проверка с различным количеством точек:

количество точек табуляции: 3

значения суммы квадратов:

$F(0.0) = 1.0$
 $F(0.1) = 0.8807817402178708$
 $F(0.2) = 0.7777748698185154$
 $F(0.30000000000000004) = 0.6909793888019344$
 $F(0.4) = 0.6203952971681272$
 $F(0.5) = 0.5660225949170943$
 $F(0.6) = 0.5278612820488353$
 $F(0.7) = 0.5059113585633503$
 $F(0.7999999999999999) = 0.5001728244606394$
 $F(0.8999999999999999) = 0.5106456797407023$
 $F(0.9999999999999999) = 0.5373299244035394$
 $F(1.0999999999999999) = 0.5802255584491507$
 $F(1.2) = 0.6393325818775359$
 $F(1.3) = 0.7146509946886953$
 $F(1.4000000000000001) = 0.8061807968826287$
 $F(1.5000000000000002) = 0.9139219884593364$

$F(1.6000000000000003) = 0.9635080262662963$
 $F(1.7000000000000004) = 0.8490240876615192$
 $F(1.8000000000000005) = 0.7507515384395163$
 $F(1.9000000000000006) = 0.6686903786002871$
 $F(2.0000000000000004) = 0.6028406081438322$
 $F(2.1000000000000005) = 0.5532022270701514$
 $F(2.2000000000000006) = 0.5197752353792446$
 $F(2.3000000000000007) = 0.5025596330711118$
 $F(2.4000000000000001) = 0.5015554201457533$
 $F(2.5000000000000001) = 0.5167625966031685$
 $F(2.6000000000000001) = 0.5481811624433582$
 $F(2.7000000000000001) = 0.5958111176663217$
 $F(2.8000000000000001) = 0.6596524622720592$
 $F(2.90000000000000012) = 0.739705196260571$
 $F(3.00000000000000013) = 0.8359693196318567$
 $F(3.10000000000000014) = 0.9484448323859167$

количество точек табуляции: 5

значения суммы квадратов:

$F(0.0) = 1.0$
 $F(0.1) = 0.9349117663199064$
 $F(0.2) = 0.8888163567108488$
 $F(0.30000000000000004) = 0.8617137711728265$
 $F(0.4) = 0.8536040097058402$
 $F(0.5) = 0.8644870723098893$
 $F(0.6) = 0.8943629589849742$
 $F(0.7) = 0.9432316697310947$
 $F(0.7999999999999999) = 0.9893117483522796$

$F(0.8999999999999999) = 0.926996815809249$
 $F(0.9999999999999999) = 0.8836747073372538$
 $F(1.0999999999999999) = 0.8593454229362943$
 $F(1.2) = 0.8540089626063706$
 $F(1.3) = 0.8676653263474826$
 $F(1.4000000000000001) = 0.9003145141596303$
 $F(1.5000000000000002) = 0.9519565260428137$
 $F(1.6000000000000003) = 0.9790284496050897$
 $F(1.7000000000000004) = 0.9194868181991217$
 $F(1.8000000000000005) = 0.8789380108641893$
 $F(1.9000000000000006) = 0.8573820276002928$
 $F(2.0000000000000004) = 0.854818868407432$
 $F(2.1000000000000005) = 0.8712485332856068$
 $F(2.2000000000000006) = 0.9066710222348173$
 $F(2.3000000000000007) = 0.9610863352550636$
 $F(2.4000000000000001) = 0.9691501037584304$
 $F(2.5000000000000001) = 0.9123817734895252$
 $F(2.6000000000000001) = 0.8746062672916557$
 $F(2.7000000000000001) = 0.855823585164822$
 $F(2.8000000000000001) = 0.8560337271090238$
 $F(2.90000000000000012) = 0.8752366931242617$
 $F(3.00000000000000013) = 0.913432483210535$
 $F(3.10000000000000014) = 0.9706210973678441$

количество точек табуляции: 10

значения суммы квадратов:

$F(0.0) = 1.0$

$F(0.1) = 0.9753452893472049$

$F(0.2) = 0.9704883234380686$
 $F(0.30000000000000004) = 0.9854291022725908$
 $F(0.4) = 0.9849684785101429$
 $F(0.5) = 0.9703975807827336$
 $F(0.6) = 0.975624427798983$
 $F(0.7) = 0.9993578909268825$
 $F(0.7999999999999999) = 0.9750730613812004$
 $F(0.8999999999999999) = 0.9705859765791769$
 $F(0.9999999999999999) = 0.9858966365208117$
 $F(1.0999999999999999) = 0.9845147652334687$
 $F(1.2) = 0.9703137486131723$
 $F(1.3) = 0.9759104767365345$
 $F(1.4000000000000001) = 0.9987226923395387$
 $F(1.5000000000000002) = 0.9748077439009694$
 $F(1.6000000000000003) = 0.9706905402060587$
 $F(1.7000000000000004) = 0.9863710812548067$
 $F(1.8000000000000005) = 0.9840679624425679$
 $F(1.9000000000000006) = 0.9702368269293845$
 $F(2.0000000000000004) = 0.9762034361598597$
 $F(2.1000000000000005) = 0.9980944042379682$
 $F(2.2000000000000006) = 0.9745493369065118$
 $F(2.3000000000000007) = 0.9708020143187142$
 $F(2.4000000000000001) = 0.9868524364745752$
 $F(2.5000000000000001) = 0.9836280701374409$
 $F(2.6000000000000001) = 0.9701668157313703$
 $F(2.7000000000000001) = 0.9765033060689583$
 $F(2.8000000000000001) = 0.9974730266221714$
 $F(2.90000000000000012) = 0.974297840397828$

$$F(3.00000000000000013) = 0.9709203989171432$$

$$F(3.10000000000000014) = 0.9873407021801173$$

количество точек табуляции: 20

значения суммы квадратов:

$$F(0.0) = 1.0$$

$$F(0.1) = 0.9934801762782068$$

$$F(0.2) = 0.9954813685939703$$

$$F(0.30000000000000004) = 0.9958763728929726$$

$$F(0.4) = 0.9933589338027065$$

$$F(0.5) = 0.9993625107499969$$

$$F(0.6) = 0.9936326956262082$$

$$F(0.7) = 0.995117641167469$$

$$F(0.7999999999999999) = 0.9963026540644755$$

$$F(0.8999999999999999) = 0.9932689681997069$$

$$F(0.9999999999999999) = 0.9987562983724949$$

$$F(1.0999999999999999) = 0.9938164918467103$$

$$F(1.2) = 0.9947851906134687$$

$$F(1.3) = 0.9967602121084793$$

$$F(1.4000000000000001) = 0.9932102794692081$$

$$F(1.5000000000000002) = 0.9981813628674935$$

$$F(1.6000000000000003) = 0.9940315649397132$$

$$F(1.7000000000000004) = 0.994484016931969$$

$$F(1.8000000000000005) = 0.997249047024984$$

$$F(1.9000000000000006) = 0.9931828676112102$$

$$F(2.0000000000000004) = 0.9976377042349929$$

$$F(2.1000000000000005) = 0.994277914905217$$

$$F(2.2000000000000006) = 0.9942141201229702$$

$F(2.3000000000000007) = 0.9977691588139896$
 $F(2.4000000000000001) = 0.993186732625713$
 $F(2.5000000000000001) = 0.9971253224749932$
 $F(2.6000000000000001) = 0.9945555417432219$
 $F(2.7000000000000001) = 0.9939755001864723$
 $F(2.8000000000000001) = 0.9983205474754961$
 $F(2.9000000000000012) = 0.9932218745127168$
 $F(3.0000000000000013) = 0.9966442175874943$
 $F(3.1000000000000014) = 0.9948644454537273$

количество точек табуляции: 50

значения суммы квадратов:

$F(0.0) = 1.0$
 $F(0.1) = 0.9989873510354049$
 $F(0.2) = 0.9995678268629896$
 $F(0.3000000000000004) = 0.9991045883172991$
 $F(0.4) = 0.9992528910078733$
 $F(0.5) = 0.9993390628810876$
 $F(0.6) = 0.9990551924346515$
 $F(0.7) = 0.9996907747267703$
 $F(0.7999999999999999) = 0.9989747311433237$
 $F(0.8999999999999999) = 0.9998518123520567$
 $F(0.9999999999999999) = 0.9990115071338901$
 $F(1.0999999999999999) = 0.9994564152056128$
 $F(1.2) = 0.9991655204063505$
 $F(1.3) = 0.9991782553410629$
 $F(1.4000000000000001) = 0.9994367709607055$
 $F(1.5000000000000002) = 0.9990173327584072$

$F(1.6000000000000003) = 0.9998252587969546$
 $F(1.7000000000000004) = 0.9989736474576458$
 $F(1.8000000000000005) = 0.999715160910517$
 $F(1.9000000000000006) = 0.999047199438779$
 $F(2.0000000000000004) = 0.9993565397546393$
 $F(2.1000000000000005) = 0.9992379887018059$
 $F(2.2000000000000006) = 0.9991151558806561$
 $F(2.3000000000000007) = 0.9995460152467273$
 $F(2.4000000000000001) = 0.998991009288567$
 $F(2.5000000000000001) = 0.9999712790735427$
 $F(2.6000000000000001) = 0.9989840999783719$
 $F(2.7000000000000001) = 0.9995900456753811$
 $F(2.8000000000000001) = 0.9990944279500713$
 $F(2.9000000000000012) = 0.9992682005100699$
 $F(3.0000000000000013) = 0.9993219932036649$
 $F(3.1000000000000014) = 0.9990635926266531$

количество точек табуляции: 100

значения суммы квадратов:

$F(0.0) = 1.0$
 $F(0.1) = 0.9998707262777297$
 $F(0.2) = 0.9997875329627255$
 $F(0.3000000000000004) = 0.9997504200549877$
 $F(0.4) = 0.9997593875545162$
 $F(0.5) = 0.999814435461311$
 $F(0.6) = 0.9999155637753722$
 $F(0.7) = 0.9999442080013095$
 $F(0.7999999999999999) = 0.9998328692780962$

$F(0.8999999999999999) = 0.999767610962149$
 $F(0.9999999999999999) = 0.9997484330534683$
 $F(1.0999999999999999) = 0.9997753355520536$
 $F(1.2) = 0.9998483184579052$
 $F(1.3) = 0.9999673817710231$
 $F(1.4000000000000001) = 0.9998953965006281$
 $F(1.5000000000000002) = 0.9998019927764716$
 $F(1.6000000000000003) = 0.9997546694595815$
 $F(1.7000000000000004) = 0.9997534265499575$
 $F(1.8000000000000005) = 0.9997982640476$
 $F(1.9000000000000006) = 0.9998891819525085$
 $F(2.0000000000000004) = 0.9999751146303213$
 $F(2.1000000000000005) = 0.9998535654979555$
 $F(2.2000000000000006) = 0.999778096772856$
 $F(2.3000000000000007) = 0.9997487084550227$
 $F(2.4000000000000001) = 0.9997654005444558$
 $F(2.5000000000000001) = 0.9998281730411551$
 $F(2.6000000000000001) = 0.9999370259451208$
 $F(2.7000000000000001) = 0.9999223291266008$
 $F(2.8000000000000001) = 0.999818714993292$
 $F(2.90000000000000012) = 0.9997611812672494$
 $F(3.00000000000000013) = 0.999749727948473$
 $F(3.10000000000000014) = 0.9997843550369631$

Вывод: чем больше точек в табулированной функции, тем лучше приближение к 1.

Экспонента: запись в файл и чтение обратно

значения исходной и считанной функций Exp на отрезке от 0 до 10 с шагом 1:
исходная функция:

```
F(0.0) = 1.0  
F(1.0) = 2.718281828459045  
F(2.0) = 7.38905609893065  
F(3.0) = 20.085536923187668  
F(4.0) = 54.598150033144236  
F(5.0) = 148.4131591025766  
F(6.0) = 403.4287934927351  
F(7.0) = 1096.6331584284585  
F(8.0) = 2980.9579870417283  
F(9.0) = 8103.083927575384  
F(10.0) = 22026.465794806718
```

считанная функция:

```
Tabulated F(0.0) = 1.0  
Tabulated F(1.0) = 2.718281828459045  
Tabulated F(2.0) = 7.38905609893065  
Tabulated F(3.0) = 20.085536923187668  
Tabulated F(4.0) = 54.59815003314424  
Tabulated F(5.0) = 148.4131591025766  
Tabulated F(6.0) = 403.4287934927351  
Tabulated F(7.0) = 1096.6331584284585  
Tabulated F(8.0) = 2980.9579870417283  
Tabulated F(9.0) = 8103.083927575384  
Tabulated F(10.0) = 22026.46579480672
```

Содержимое файла **tabulated_exp.txt**:

```
11 0.0 1.0 1.0 2.718281828459045 2.0 7.38905609893065 3.0  
20.085536923187668 4.0 54.598150033144236 5.0  
148.4131591025766 6.0 403.4287934927351 7.0  
1096.6331584284585 8.0 2980.9579870417283 9.0  
8103.083927575384 10.0 22026.465794806718
```

Логарифм: запись в файл и чтение обратно

значения исходной и считанной функций Log на отрезке от 0 до 10 с шагом 1:

исходная функция:

```
F(0.0) = NaN
F(1.0) = 0.0
F(2.0) = 0.6931471805599453
F(3.0) = 1.0986122886681098
F(4.0) = 1.3862943611198906
F(5.0) = 1.6094379124341003
F(6.0) = 1.791759469228055
F(7.0) = 1.9459101490553132
F(8.0) = 2.0794415416798357
F(9.0) = 2.1972245773362196
F(10.0) = 2.302585092994046
```

считанная функция:

```
Tabulated F(0.0) = NaN
Tabulated F(1.0) = 0.0
Tabulated F(2.0) = 0.6931471805599453
Tabulated F(3.0) = 1.0986122886681098
Tabulated F(4.0) = 1.3862943611198906
Tabulated F(5.0) = 1.6094379124341003
Tabulated F(6.0) = 1.791759469228055
Tabulated F(7.0) = 1.9459101490553132
Tabulated F(8.0) = 2.0794415416798357
Tabulated F(9.0) = 2.1972245773362196
Tabulated F(10.0) = 2.302585092994046
```

Содержимое файла **tabulated_log.txt**:

```
@      ?ш      ?р      @      ?ж.Вюь9п@      ?с"кz
@      ?ц.Вюь9п@      ?щААчнКЗ@      ?ь«
ь* @      ?я"r©2ZW@      @ ŸI?;«s@      @@"кz
@$      @@k±»μU@
```

Вывод по **writeTabulatedFunction / readTabulatedFunction**.

Преимущества: читаемость, легко проверить корректность данных;

Недостатки: размер, поскольку числа хранятся в десятичном виде

Вывод по **outputTabulatedFunction / inputTabulatedFunction**.

Преимущества: минимальный размер, корректное сохранение NaN;

Недостатки: нечитаемость

В ходе выполнения задания 9 была проверена работа написанных классов с использованием интерфейсов **java.io.Serializable** и **java.io.Externalizable**.

LinkedListTabulatedFunction реализует **Serializable**, а **ArrayTabulatedFunction** реализует **Externalizable**.

Был создан дополнительный метод **getAllPoints()**. Это метод интерфейса **TabulatedFunction**, который возвращает копию всех точек функции в виде массива. Он нужен, чтобы взять точки из любой табулированной функции и передать их в конструктор другой реализации и протестировать два случая сериализации.

Для **Serializable**:

serializable: значения исходной и считанной функций Log на отрезке от 0 до 10 с шагом 1:

исходная функция:

```
Tabulated F(0.0) = 0.0
Tabulated F(1.0) = 1.0
Tabulated F(2.0) = 2.0
Tabulated F(3.0) = 3.0
Tabulated F(4.0) = 4.0
Tabulated F(5.0) = 5.0
Tabulated F(6.0) = 6.0
Tabulated F(7.0) = 7.0
Tabulated F(8.0) = 8.0
Tabulated F(9.0) = 9.0
Tabulated F(10.0) = 10.0
```

считанная функция:

```
Tabulated F(0.0) = 0.0
Tabulated F(1.0) = 1.0
Tabulated F(2.0) = 2.0
Tabulated F(3.0) = 3.0
Tabulated F(4.0) = 4.0
Tabulated F(5.0) = 5.0
Tabulated F(6.0) = 6.0
Tabulated F(7.0) = 7.0
Tabulated F(8.0) = 8.0
Tabulated F(9.0) = 9.0
Tabulated F(10.0) = 10.0
```

Содержимое файла **serializable_tabulated_log.txt**:

```
~н Bsr %functions.LinkedListTabulatedFunction@y83 Bb@ @I @lastAccessedIndexI @sizeL @headt 4Lfunctions/LinkedListTabulatedFunction$FunctionNode;L @lastAccessedNodeq ~ Bxp
sr 2functions.LinkedListTabulatedFunction$FunctionNode]-KTTs+1@ @L @nextq ~ @L @pointtt @Lfunctions/FunctionPoint;L @prevq ~ Bxpsq ~ Bsq ~ Bsq ~ Bsq ~ Bsq ~ Bsq ~
Bsq ~ Bsq ~ Bsq ~ Bq ~ Bsr @functions.FunctionPointIc|N@BfNK@ @D Bx0 Bxyp@ $ @ $ q ~ Bsq ~ B@ " @ " q ~ Bsq ~ B@ @ @ q ~
sq ~ B@ @ @ q ~
sq ~ B@ @ @ q ~
sq ~ B@ @ @ q ~
sq ~ B@ @ @ q ~ sq ~ B@ @ @ @ q ~ Bsq ~ B@ @ @ q ~ sq ~ B?p ?p q ~ Bsq ~ @ q ~ Bpq ~ Bq ~ @
```


Для Externalizable:

```
externalizable: значения исходной и считанной функций Log на отрезке от 0 до 10 с шагом 1:
исходная функция:
Tabulated F(0.0) = 0.0
Tabulated F(1.0) = 1.0
Tabulated F(2.0) = 2.0
Tabulated F(3.0) = 3.0
Tabulated F(4.0) = 4.0
Tabulated F(5.0) = 5.0
Tabulated F(6.0) = 6.0
Tabulated F(7.0) = 7.0
Tabulated F(8.0) = 8.0
Tabulated F(9.0) = 9.0
Tabulated F(10.0) = 10.0

считанная функция:
Tabulated F(0.0) = 0.0
Tabulated F(1.0) = 1.0
Tabulated F(2.0) = 2.0
Tabulated F(3.0) = 3.0
Tabulated F(4.0) = 4.0
Tabulated F(5.0) = 5.0
Tabulated F(6.0) = 6.0
Tabulated F(7.0) = 7.0
Tabulated F(8.0) = 8.0
Tabulated F(9.0) = 9.0
Tabulated F(10.0) = 10.0

Process finished with exit code 0
```

Содержимое файла **externalizable_tabulated_log.txt**:

Вывод по **Serializable**:

Преимущества: автоматическая сериализация

Недостатки: большой размер

Вывод по **Externalizable**:

Преимущества: можно сериализовать только необходимые данные, меньше размер

Недостатки: сложность реализации