Лабораторная работа №3

Доработка пакета для работы с табулированными функциями

Выполнил: Скиба В.А.
Группа: 6204-010302D

Содержание

Задание 1: Изучение исключений

Ход выполнения:
Изучена иерархия исключений Java и их применение:

- `Exception` - базовый класс проверяемых исключений
- `RuntimeException` - непроверяемые исключения
- `IndexOutOfBoundsException` - выход за границы индекса
- `IllegalArgumentException` - неверный аргумент метода
- `IllegalStateException` - неверное состояние объекта

Результат:
Определены подходящие типы исключений для различных ошибок в работе с табулированными функциями.

Задание 2: Создание классов исключений

Ход выполнения:
Созданы два класса исключений в пакете `functions`:

- `FunctionPointIndexOutOfBoundsException` - наследует от `IndexOutOfBoundsException`
- `InappropriateFunctionPointException` - наследует от `RuntimeException`

Реализация: (в исходниках в каталоге `src/functions`)
Результат:
Созданы специализированные исключения для обработки ошибок работы с точками функций.

Задание 3: Модификация TabulatedFunction

Ход выполнения:
Добавлена обработка исключений в класс TabulatedFunction:

- Конструкторы - проверка параметров с выбрасыванием `IllegalArgumentException`
- Методы доступа - проверка индексов с выбрасыванием `FunctionPointIndexOutOfBoundsException`
- Методы изменения - проверка порядка точек с выбрасыванием `InappropriateFunctionPointException`

- `deletePoint()` - проверка минимального количества точек с выбрасыванием `IllegalStateException`

Результат:
Класс TabulatedFunction теперь корректно обрабатывает все ошибочные ситуации через исключения.

Задание 4: Создание LinkedListTabulatedFunction

Ход выполнения:
Реализован класс для работы с табулированными функциями на основе связного списка:

- Внутренний класс `FunctionNode` - элемент двусвязного списка
- Структура данных - циклический двусвязный список с выделенной головой
- Базовые методы: `getNodeByIndex()`, `addNodeByIndex()`, `deleteNodeByIndex()`

Особенности реализации:
Инкапсуляция класса `FunctionNode` как `private static`, оптимизация доступа через кэширование последнего использованного элемента, циклическая структура для упрощения операций.
Результат:
Создана базовая структура связного списка для хранения точек функции.

Задание 5: Реализация функциональности LinkedListTabulatedFunction

Ход выполнения:
Реализованы конструкторы и методы, аналогичные `ArrayTabulatedFunction`:

- Конструкторы - создание функции с равномерным распределением точек
- Методы интерфейса - `getLeftDomainBorder()`, `getFunctionValue()` и др.
- Обработка исключений - аналогично `ArrayTabulatedFunction`
- Оптимизации - прямой доступ к узлам списка

Результат:
Класс `LinkedListTabulatedFunction` предоставляет тот же функционал, что и `ArrayTabulatedFunction`, но с другой внутренней структурой данных.

Задание 6: Рефакторинг и создание интерфейса

Ход выполнения:
Проведён рефакторинг: класс, ранее называвшийся `TabulatedFunction`, приведён в соответствие с интерфейсом `TabulatedFunction`, создан единый интерфейс с общими методами, реализованный в обеих версиях (Array и LinkedList). Обновлены исключения — обеспечен непроверяемый характер там, где это уместно.
Результат:
Создана единая точка доступа к функциональности табулированных функций через интерфейс.

Задание 7: Тестирование

Ход выполнения:
Созданы комплексные тесты в классе `Main`:

- Сравнение реализаций — идентичность поведения Array и LinkedList версий
- Тестирование исключений — проверка всех сценариев обработки ошибок
- Демонстрация полиморфизма — работа через интерфейс `TabulatedFunction`

- Проверка инкапсуляции — возврат копий точек

Результат:
Обе реализации работают правильно, показывают одинаковые результаты и правильно обрабатывают ошибки.

Результаты работы

Что реализовано:

- ✅ Два класса исключений для обработки ошибок
- ✅ Модифицированный `ArrayTabulatedFunction` с обработкой исключений
- ✅ Новая реализация `LinkedListTabulatedFunction` на связном списке
- ✅ Единый интерфейс `TabulatedFunction` для полиморфизма
- ✅ Комплексное тестирование всех возможностей

Ключевые особенности:

- Инкапсуляция — внутренние структуры скрыты, возвращаются копии объектов
- Наследование — исключения наследуются от стандартных классов Java
- Полиморфизм — работа с разными реализациями через общий интерфейс
- Обработка ошибок — корректная реакция на все ошибочные ситуации

Пример вывода программы

```
=== Лабораторная работа №4 (сокращённый вывод) ===
Constructors test: Array pts=3, Linked pts=3
Function test: domain=[0.0,2.0] f(1.5)=2.5

--- Демонстрация: табулирование, IO и сериализация (сравнение) ---

Задание 8: Sin и Cos и табулирование
Значения sin и cos на [0, pi] с шагом 0.1:
x=0.00 sin=0.000000 cos=1.000000
x=0.10 sin=0.099833 cos=0.995004
x=0.20 sin=0.198669 cos=0.980067
x=0.30 sin=0.295520 cos=0.955336
x=0.40 sin=0.389418 cos=0.921061
x=0.50 sin=0.479426 cos=0.877583
x=0.60 sin=0.564642 cos=0.825336
x=0.70 sin=0.644218 cos=0.764842
x=0.80 sin=0.717356 cos=0.696707
x=0.90 sin=0.783327 cos=0.621610
x=1.00 sin=0.841471 cos=0.540302
x=1.10 sin=0.891207 cos=0.453596
x=1.20 sin=0.932039 cos=0.362358
x=1.30 sin=0.963558 cos=0.267499
x=1.40 sin=0.985450 cos=0.169967
x=1.50 sin=0.997495 cos=0.070737
x=1.60 sin=0.999574 cos=-0.029200
x=1.70 sin=0.991665 cos=-0.128844
x=1.80 sin=0.973848 cos=-0.227202
x=1.90 sin=0.946300 cos=-0.323290
x=2.00 sin=0.909297 cos=-0.416147
x=2.10 sin=0.863209 cos=-0.504846
x=2.20 sin=0.808496 cos=-0.588501
x=2.30 sin=0.745705 cos=-0.666276
x=2.40 sin=0.675463 cos=-0.737394
x=2.50 sin=0.598472 cos=-0.801144
x=2.60 sin=0.515501 cos=-0.856889
```

```
x=2.70 sin=0.427380 cos=-0.904072
x=2.80 sin=0.334988 cos=-0.942222
x=2.90 sin=0.239249 cos=-0.970958
x=3.00 sin=0.141120 cos=-0.989992
x=3.10 sin=0.041581 cos=-0.999135

Сравнение табулированных и аналитических значений (шаг 0.1):
x=0.00 sin=0.000000 tabSin=0.000000 | cos=1.000000 tabCos=1.000000
x=0.10 sin=0.099833 tabSin=0.097982 | cos=0.995004 tabCos=0.982723
x=0.20 sin=0.198669 tabSin=0.195963 | cos=0.980067 tabCos=0.965446
x=0.30 sin=0.295520 tabSin=0.293945 | cos=0.955336 tabCos=0.948170
x=0.40 sin=0.389418 tabSin=0.385907 | cos=0.921061 tabCos=0.914355
x=0.50 sin=0.479426 tabSin=0.472070 | cos=0.877583 tabCos=0.864608
x=0.60 sin=0.564642 tabSin=0.558234 | cos=0.825336 tabCos=0.814862
x=0.70 sin=0.644218 tabSin=0.643982 | cos=0.764842 tabCos=0.764620
x=0.80 sin=0.717356 tabSin=0.707935 | cos=0.696707 tabCos=0.688404
x=0.90 sin=0.783327 tabSin=0.771888 | cos=0.621610 tabCos=0.612188
x=1.00 sin=0.841471 tabSin=0.835841 | cos=0.540302 tabCos=0.535972
x=1.10 sin=0.891207 tabSin=0.883993 | cos=0.453596 tabCos=0.450633
x=1.20 sin=0.932039 tabSin=0.918022 | cos=0.362358 tabCos=0.357141
x=1.30 sin=0.963558 tabSin=0.952051 | cos=0.267499 tabCos=0.263648
x=1.40 sin=0.985450 tabSin=0.984808 | cos=0.169967 tabCos=0.169931
x=1.50 sin=0.997495 tabSin=0.984808 | cos=0.070737 tabCos=0.070437
x=1.60 sin=0.999574 tabSin=0.984808 | cos=-0.029200 tabCos=-0.029056
x=1.70 sin=0.991665 tabSin=0.984808 | cos=-0.128844 tabCos=-0.128549
x=1.80 sin=0.973848 tabSin=0.966204 | cos=-0.227202 tabCos=-0.224761
x=1.90 sin=0.946300 tabSin=0.932175 | cos=-0.323290 tabCos=-0.318254
x=2.00 sin=0.909297 tabSin=0.898147 | cos=-0.416147 tabCos=-0.411747
x=2.10 sin=0.863209 tabSin=0.862441 | cos=-0.504846 tabCos=-0.504272
x=2.20 sin=0.808496 tabSin=0.798488 | cos=-0.588501 tabCos=-0.580488
x=2.30 sin=0.745705 tabSin=0.734535 | cos=-0.666276 tabCos=-0.656704
x=2.40 sin=0.675463 tabSin=0.670582 | cos=-0.737394 tabCos=-0.732920
x=2.50 sin=0.598472 tabSin=0.594072 | cos=-0.801144 tabCos=-0.794171
x=2.60 sin=0.515501 tabSin=0.507908 | cos=-0.856889 tabCos=-0.843917
x=2.70 sin=0.427380 tabSin=0.421745 | cos=-0.904072 tabCos=-0.893664
x=2.80 sin=0.334988 tabSin=0.334698 | cos=-0.942222 tabCos=-0.940984
x=2.90 sin=0.239249 tabSin=0.236716 | cos=-0.970958 tabCos=-0.958261
x=3.00 sin=0.141120 tabSin=0.138735 | cos=-0.989992 tabCos=-0.975537
x=3.10 sin=0.041581 tabSin=0.040753 | cos=-0.999135 tabCos=-0.992814

Сумма квадратов табулированных аналогов (на [0, pi]):
x=0.00 val=1.000000
x=0.10 val=0.975345
x=0.20 val=0.970488
x=0.30 val=0.985429
x=0.40 val=0.984968
x=0.50 val=0.970398
x=0.60 val=0.975624
x=0.70 val=0.999358
x=0.80 val=0.975073
x=0.90 val=0.970586
x=1.00 val=0.985897
x=1.10 val=0.984515
x=1.20 val=0.970314
x=1.30 val=0.975910
x=1.40 val=0.998723
x=1.50 val=0.974808
x=1.60 val=0.970691
x=1.70 val=0.986371
x=1.80 val=0.984068
x=1.90 val=0.970237
x=2.00 val=0.976203
x=2.10 val=0.998094
x=2.20 val=0.974549
```

```
x=2.30 val=0.970802
x=2.40 val=0.986852
x=2.50 val=0.983628
x=2.60 val=0.970167
x=2.70 val=0.976503
x=2.80 val=0.997473
x=2.90 val=0.974298
x=3.00 val=0.970920
x=3.10 val=0.987341

Сравнение экспоненты и считанной из текстового файла (шаг 1):
x=0 orig=1.000000 read=1.000000
x=1 orig=2.718282 read=2.718282
x=2 orig=7.389056 read=7.389056
x=3 orig=20.085537 read=20.085537
x=4 orig=54.598150 read=54.598150
x=5 orig=148.413159 read=148.413159
x=6 orig=403.428793 read=403.428793
x=7 orig=1096.633158 read=1096.633158
x=8 orig=2980.957987 read=2980.957987
x=9 orig=8103.083928 read=8103.083928
x=10 orig=22026.465795 read=22026.465795

Сравнение логарифма и считанного из бинарного файла (шаг 1):
x=0 orig=NaN read=NaN
x=1 orig=0.000000 read=0.000000
x=2 orig=0.693147 read=0.693147
x=3 orig=1.098612 read=1.098612
x=4 orig=1.386294 read=1.386294
x=5 orig=1.609438 read=1.609438
x=6 orig=1.791759 read=1.791759
x=7 orig=1.945910 read=1.945910
x=8 orig=2.079442 read=2.079442
x=9 orig=2.197225 read=2.197225
x=10 orig=2.302585 read=2.302585

Сравнение сериализованной и десериализованной функции (шаг 1):
x=0 orig=0.000000 deser=0.000000
x=1 orig=1.000000 deser=1.000000
x=2 orig=2.000000 deser=2.000000
x=3 orig=3.000000 deser=3.000000
x=4 orig=4.000000 deser=4.000000
x=5 orig=5.000000 deser=5.000000
x=6 orig=6.000000 deser=6.000000
x=7 orig=7.000000 deser=7.000000
x=8 orig=8.000000 deser=8.000000
x=9 orig=9.000000 deser=9.000000
x=10 orig=10.000000 deser=10.000000

Сравнение Serializable vs Externalizable: сериализация в файлы и сравнение
размеров
Externalizable file: tf_externalizable.obj size=1684 bytes
Serializable(POJO) file: tf_serializable_pojo.obj size=2401 bytes

Первые 64 байта externalizable (hex):
AC ED 00 05 73 72 00 20 66 75 6E 63 74 69 6F 6E
73 2E 41 72 72 61 79 54 61 62 75 6C 61 74 65 64
46 75 6E 63 74 69 6F 6E E4 1B B7 A5 CC 42 1E 57
0C 00 00 78 70 7A 00 00 04 00 00 00 00 65 00 00


Первые 64 байта serializable POJO (hex):
AC ED 00 05 73 72 00 1F 66 75 6E 63 74 69 6F 6E
73 2E 54 61 62 75 6C 61 74 65 64 46 75 6E 63 74
```

```
69 6F 6E 44 61 74 61 00 00 00 00 00 00 00 01 02
00 01 5B 00 06 70 6F 69 6E 74 73 74 00 1A 5B 4C
```

Заключение

В ходе лабораторной работы успешно расширен пакет для работы с табулированными функциями. Реализованы обработка исключений, альтернативная реализация на связном списке и архитектура на основе интерфейсов. Все требования задания выполнены в полном объеме.

Места для скриншотов

MathUtil

```java
package functions;

public final class MathUtil {

    public static final double EPS = 1e-10;

    private MathUtil() {}

    public static boolean equals(double a, double b) {
        if (Double.isNaN(a) || Double.isNaN(b)) return false;
        if (a == b) return true;

        double diff = Math.abs(a - b);
        if (diff <= EPS) return true;

        double max = Math.max(Math.abs(a), Math.abs(b));

        return diff <= Math.max(EPS, max * 1e-12);
    }

    public static boolean less(double a, double b) {
        return a < b && !equals(a, b);
    }

    public static boolean lessOrEquals(double a, double b) {
        return a < b || equals(a, b);
    }

    public static boolean greater(double a, double b) {
        return a > b && !equals(a, b);
    }

    public static boolean greaterOrEquals(double a, double b) {
        return a > b || equals(a, b);
    }

    public static boolean isZero(double a) {
        return equals(a, b: 0.0);
    }
}
```

ArrayTabulatedFunction

```java
package functions;

import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.util.Arrays;

public class ArrayTabulatedFunction implements TabulatedFunction, Externalizable {
    private FunctionPoint[] points;
    private int pointsCount;

    public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница должна быть меньше правой: " + leftX + " >= " + rightX);
        }
        if (pointsCount < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2: " + pointsCount);
        }

        this.pointsCount = pointsCount;
        this.points = new FunctionPoint[pointsCount + 5];

        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, y: 0);
        }
    }

    public ArrayTabulatedFunction(double leftX, double rightX, double[] values) {
        if (leftX >= rightX) {
            throw new IllegalArgumentException("Левая граница должна быть меньше правой: " + leftX + " >= " + rightX);
        }
        if (values.length < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2: " + values.length);
        }

        this.pointsCount = values.length;
        this.points = new FunctionPoint[pointsCount + 5];

        double step = (rightX - leftX) / (pointsCount - 1);
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            points[i] = new FunctionPoint(x, values[i]);
        }
    }

    public ArrayTabulatedFunction(FunctionPoint[] points) {
        if (points == null) {
            throw new IllegalArgumentException(s: "Массив точек не может быть null");
        }
        if (points.length < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2: " + points.length);
        }

        for (int i = 0; i < points.length - 1; i++) {
            if (!MathUtil.less(points[i].getX(), points[i + 1].getX())) {
                throw new IllegalArgumentException(s: "Точки не упорядочены по X или содержат дубликаты");
            }
        }

        this.pointsCount = points.length;
        this.points = new FunctionPoint[pointsCount + 5];

        for (int i = 0; i < pointsCount; i++) {
            this.points[i] = new FunctionPoint(points[i]);
        }
        if (this.pointsCount < this.points.length) {
            Arrays.fill(this.points, this.pointsCount, this.points.length, val: null);
        }
    }

    public double getLeftDomainBorder() {...

    public double getRightDomainBorder() {
        return points[pointsCount - 1].getX();
    }

    public double getFunctionValue(double x) {...

    public int getPointsCount() {...

    public FunctionPoint getPoint(int index) {...

    public void setPoint(int index, FunctionPoint point) {...

    public double getPointX(int index) {...

    public void setPointX(int index, double x) {...

    public double getPointY(int index) {...

    public void setPointY(int index, double y) {...

    public void deletePoint(int index) {...

    public void addPoint(FunctionPoint point) {...

    public ArrayTabulatedFunction() {
    }

    @Override
    public void writeExternal(ObjectOutput out) throws IOException {
        out.writeInt(pointsCount);
        for (int i = 0; i < pointsCount; i++) {
            out.writeDouble(points[i].getX());
            out.writeDouble(points[i].getY());
        }
    }

    @Override
    public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
        int n = in.readInt();
        this.pointsCount = n;
        this.points = new FunctionPoint[n + 5];
        for (int i = 0; i < n; i++) {
            double x = in.readDouble();
            double y = in.readDouble();
            this.points[i] = new FunctionPoint(x, y);
        }
        if (this.pointsCount < this.points.length) {
            Arrays.fill(this.points, this.pointsCount, this.points.length, val: null);
        }
    }
}
```

TabulatedFunctions

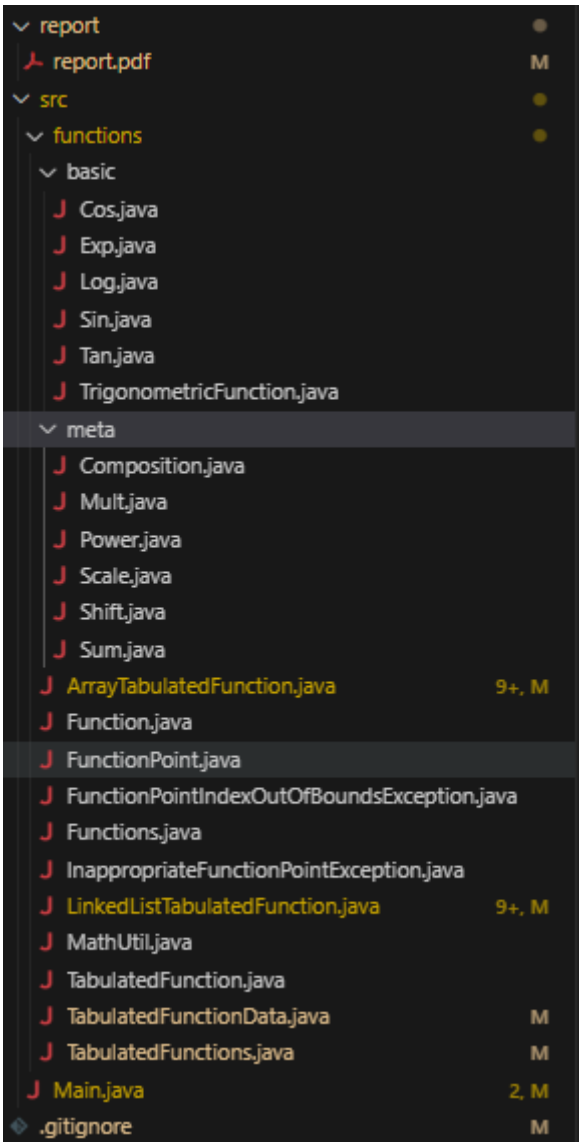
```java
package functions;

public interface TabulatedFunction extends Function {
    int getPointsCount();
    FunctionPoint getPoint(int index);
    void setPoint(int index, FunctionPoint point);
    double getPointX(int index);
    void setPointX(int index, double x);
    double getPointY(int index);
    void setPointY(int index, double y);

    void deletePoint(int index);
    void addPoint(FunctionPoint point);
}
```

ulatedFunctionData.java M ×

functions > TabulatedFunctionData.java > Java > {} functions

```java
package functions;

import java.io.Serializable;

public class TabulatedFunctionData implements Serializable {
    private static final long serialVersionUID = 1L;

    private final FunctionPoint[] points;

    public TabulatedFunctionData(FunctionPoint[] points) {
        this.points = new FunctionPoint[points.length];
        for (int i = 0; i < points.length; i++) this.points[i] = new FunctionPoint(points[i]);
    }

    public int getPointsCount() { return points.length; }
    public FunctionPoint getPoint(int i) { return new FunctionPoint(points[i]); }
}
```

```java
package functions;

import java.io.*;

public final class TabulatedFunctions {
    private static final double EPS = 1e-10;

    private TabulatedFunctions() {}

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
        if (leftX < function.getLeftDomainBorder() - EPS || rightX > function.getRightDomainBorder() + EPS)
            throw new IllegalArgumentException(s: "Границы табулирования выходят за область определения функции");
        if (pointsCount < 2) throw new IllegalArgumentException(s: "pointsCount < 2");

        double step = (rightX - leftX) / (pointsCount - 1);
        FunctionPoint[] pts = new FunctionPoint[pointsCount];
        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            pts[i] = new FunctionPoint(x, function.getFunctionValue(x));
        }
        return new ArrayTabulatedFunction(pts);
    }

    public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {
        DataOutputStream dos = new DataOutputStream(new BufferedOutputStream(out));
        try {
            dos.writeInt(function.getPointsCount());
            for (int i = 0; i < function.getPointsCount(); i++) {
                dos.writeDouble(function.getPointX(i));
                dos.writeDouble(function.getPointY(i));
            }
            dos.flush();
        } finally {
        }
    }

    public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException {
        DataInputStream dis = new DataInputStream(new BufferedInputStream(in));
        int n = dis.readInt();
        FunctionPoint[] pts = new FunctionPoint[n];
        for (int i = 0; i < n; i++) {
            double x = dis.readDouble();
            double y = dis.readDouble();
            pts[i] = new FunctionPoint(x, y);
        }
        return new ArrayTabulatedFunction(pts);
    }

    public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException {
        PrintWriter pw = new PrintWriter(new BufferedWriter(out));
        try {
            StringBuilder sb = new StringBuilder();
            sb.append(function.getPointsCount());
            for (int i = 0; i < function.getPointsCount(); i++) {
                sb.append(c: ' ').append(function.getPointX(i)).append(c: ' ').append(function.getPointY(i));
            }
            pw.println(sb.toString());
            pw.flush();
        } finally {
        }
    }

    public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException {
        StreamTokenizer st = new StreamTokenizer(in);
        st.parseNumbers();
        int n;
        if (st.nextToken() != StreamTokenizer.TT_NUMBER) throw new IOException(message: "Invalid format: expected number of points");
        n = (int) st.nval;
        FunctionPoint[] pts = new FunctionPoint[n];
        for (int i = 0; i < n; i++) {
            if (st.nextToken() != StreamTokenizer.TT_NUMBER) throw new IOException(message: "Invalid format: expected x");
            double x = st.nval;
            if (st.nextToken() != StreamTokenizer.TT_NUMBER) throw new IOException(message: "Invalid format: expected y");
            double y = st.nval;
            pts[i] = new FunctionPoint(x, y);
        }
        return new ArrayTabulatedFunction(pts);
    }
}
```


report
report.pdf M
src
functions
basic
Cos.java
Exp.java
Log.java
Sin.java
Tan.java
TrigonometricFunction.java
meta
Composition.java
Mult.java
Power.java
Scale.java
Shift.java
Sum.java
ArrayTabulatedFunction.java 9+, M
Function.java
FunctionPoint.java
FunctionPointIndexOutOfBoundsException.java
Functions.java
InappropriateFunctionPointException.java
LinkedListTabulatedFunction.java 9+, M
MathUtil.java
TabulatedFunction.java
TabulatedFunctionData.java M
TabulatedFunctions.java M
Main.java 2, M
.gitignore M