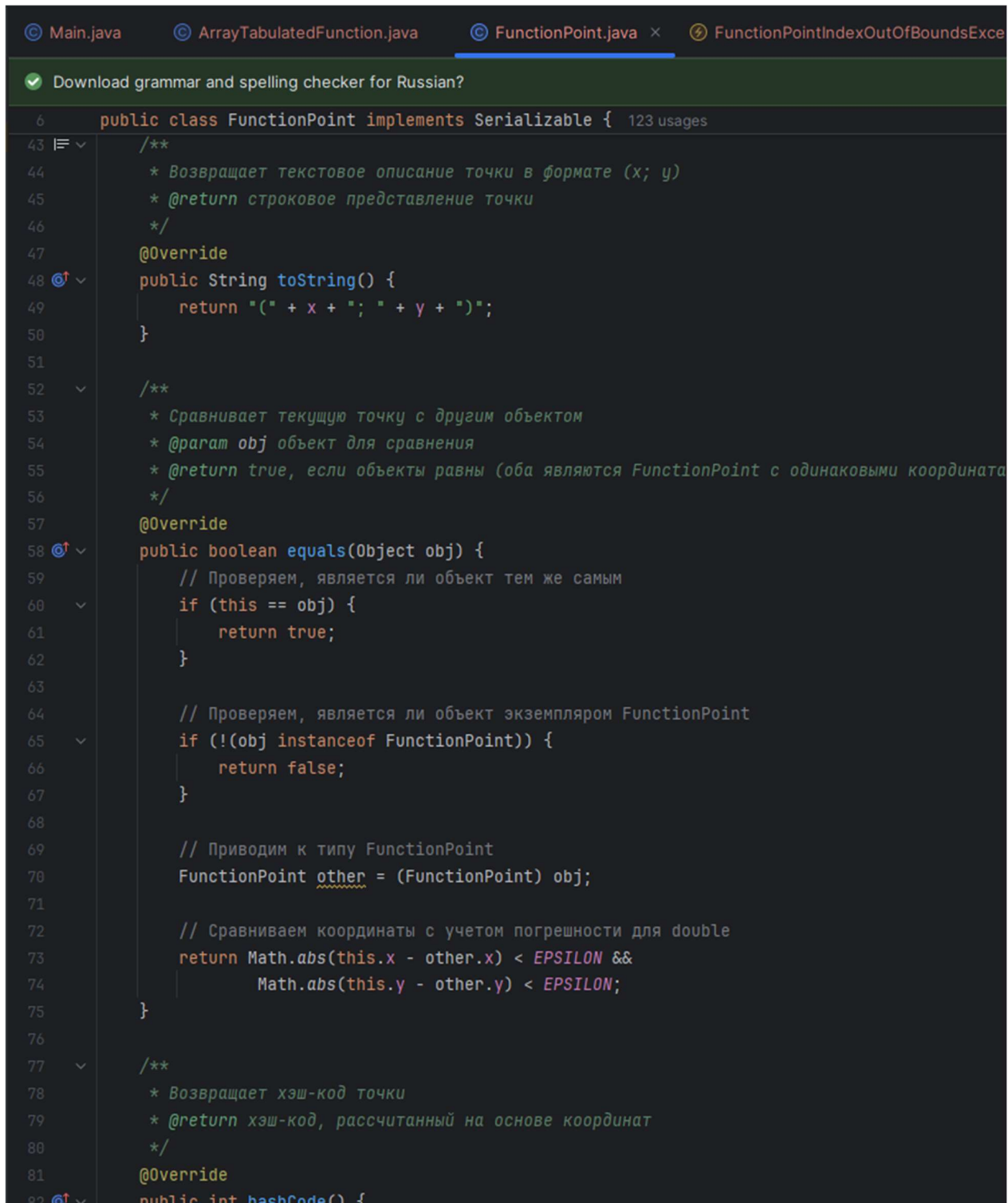


Лабораторная работа №5

Выполнил : Горынин Дмитрий, ст.гр. 6201-120303D.

Задание 1.

Переопределение методов в FunctionPoint.



```
6 public class FunctionPoint implements Serializable { 123 usages
43 /**
44  * Возвращает текстовое описание точки в формате (x; y)
45  * @return строковое представление точки
46  */
47 @Override
48 public String toString() {
49     return "(" + x + "; " + y + ")";
50 }
51
52 /**
53  * Сравнивает текущую точку с другим объектом
54  * @param obj объект для сравнения
55  * @return true, если объекты равны (оба являются FunctionPoint с одинаковыми координатами)
56  */
57 @Override
58 public boolean equals(Object obj) {
59     // Проверяем, является ли объект тем же самым
60     if (this == obj) {
61         return true;
62     }
63
64     // Проверяем, является ли объект экземпляром FunctionPoint
65     if (!(obj instanceof FunctionPoint)) {
66         return false;
67     }
68
69     // Приводим к типу FunctionPoint
70     FunctionPoint other = (FunctionPoint) obj;
71
72     // Сравниваем координаты с учетом погрешности для double
73     return Math.abs(this.x - other.x) < EPSILON &&
74         Math.abs(this.y - other.y) < EPSILON;
75 }
76
77 /**
78  * Возвращает хэш-код точки
79  * @return хэш-код, рассчитанный на основе координат
80  */
81 @Override
82 public int hashCode() {
```

Задание 2.

Переопределение методов в ArrayTabulatedFunction.

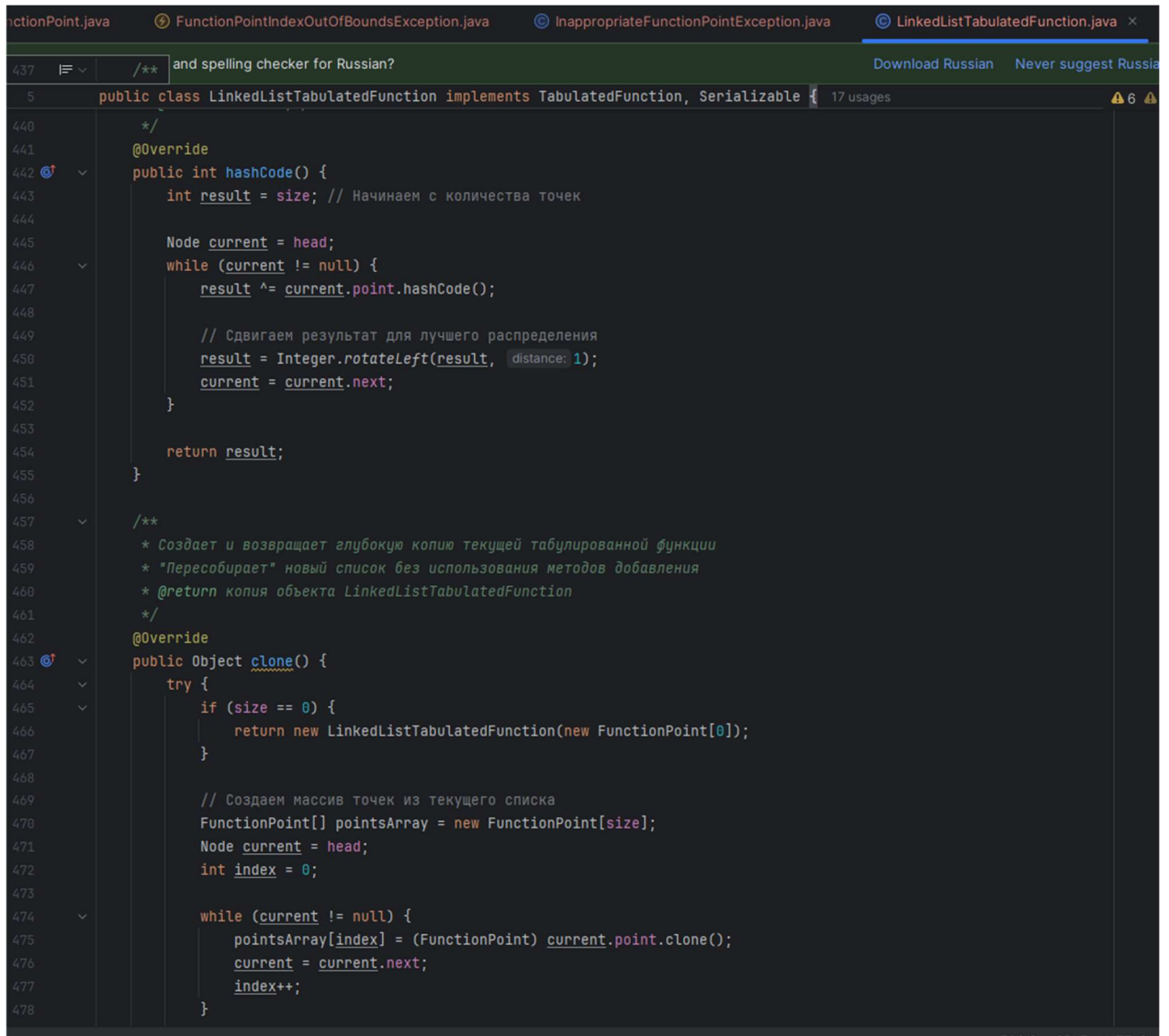
```

Main.java  ArrayTabulatedFunction.java  FunctionPoint.java  FunctionPointIndexOutOfBoundsException
Download grammar and spelling checker for Russian?

5      public class ArrayTabulatedFunction implements TabulatedFunction, Serializable { 23 usages
349          /**
350           * Возвращает хэш-код табулированной функции
351           * @return хэш-код, рассчитанный на основе количества точек и хэш-кодов всех точек
352           */
353          @Override
354          public int hashCode() {
355              int result = size; // Начинаем с количества точек
356
357              // Добавляем хэш-код каждой точки
358              for (int i = 0; i < size; i++) {
359                  result ^= points[i].hashCode();
360
361                  // Сдвигаем результат для лучшего распределения
362                  result = Integer.rotateLeft(result, distance: 1);
363              }
364
365              return result;
366          }
367
368          @Override
369          public Object clone() {
370              try {
371                  // Создаем массив точек из текущей функции
372                  FunctionPoint[] pointsArray = new FunctionPoint[size];
373                  for (int i = 0; i < size; i++) {
374                      pointsArray[i] = (FunctionPoint) points[i].clone();
375                  }
376
377                  // Используем конструктор с массивом точек
378                  return new ArrayTabulatedFunction(pointsArray);
379              } catch (Exception e) {
380                  throw new InternalError("Ошибка при клонировании: " + e.getMessage());
381              }
382          }
383      }
384
385      // Вспомогательные методы
386      private boolean isValidXPosition(int index, double newX) { 2 usages
387          if (index > 0 && newX <= points[index - 1].getX() + EPSILON) {
```

Задание 3.

Аналогично, переопределите методы `toString()`, `equals()`, `hashCode()` и `clone()` в классе `LinkedListTabulatedFunction`.



```
functionPoint.java  FunctionPointIndexOutOfBoundsException.java  InappropriateFunctionPointException.java  LinkedListTabulatedFunction.java x
and spelling checker for Russian?  Download Russian  Never suggest Russian
5  public class LinkedListTabulatedFunction implements TabulatedFunction, Serializable { 17 usages
440  */
441  @Override
442  public int hashCode() {
443      int result = size; // Начинаем с количества точек
444
445      Node current = head;
446      while (current != null) {
447          result ^= current.point.hashCode();
448
449          // Сдвигаем результат для лучшего распределения
450          result = Integer.rotateLeft(result, distance: 1);
451          current = current.next;
452      }
453
454      return result;
455  }
456
457  /**
458   * Создает и возвращает глубокую копию текущей табулированной функции
459   * "Пересобирает" новый список без использования методов добавления
460   * @return копия объекта LinkedListTabulatedFunction
461   */
462  @Override
463  public Object clone() {
464      try {
465          if (size == 0) {
466              return new LinkedListTabulatedFunction(new FunctionPoint[0]);
467          }
468
469          // Создаем массив точек из текущего списка
470          FunctionPoint[] pointsArray = new FunctionPoint[size];
471          Node current = head;
472          int index = 0;
473
474          while (current != null) {
475              pointsArray[index] = (FunctionPoint) current.point.clone();
476              current = current.next;
477              index++;
478          }
479      }
480  }
```

Задание 4.

Сделайте так, чтобы все объекты типа `TabulatedFunction` были клонируемыми с точки зрения JVM и внесите метод `clone()` в этот интерфейс.

```
le.java  Shift.java  Composition.java  Functions.java  TabulatedFunctions.java  Function.java  TabulatedFunction.java x  ⋮
2
3  /**
4   * Интерфейс для табулированных функций (расширяет Function)
5   */
6  public interface TabulatedFunction extends Function, Cloneable { 35 usages  3 implementations
7
8      // Методы доступа к точкам
9      int getPointsCount(); 14 usages  3 implementations
10
11      FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException; 5 usages  3 implementations
12
13      void setPoint(int index, FunctionPoint point) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;
14
15      double getPointX(int index) throws FunctionPointIndexOutOfBoundsException; 14 usages  3 implementations
16
17      void setPointX(int index, double x) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException; no usages
18
19      double getPointY(int index) throws FunctionPointIndexOutOfBoundsException; 10 usages  3 implementations
20
21      void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException; 3 usages  3 implementations
22
23      // Методы модификации точек
24      void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException, IllegalStateException; no usages  3 implementations
25
26      void addPoint(FunctionPoint point) throws InappropriateFunctionPointException; no usages  3 implementations
27
28      // Метод для вывода информации о функции
29      void printFunction(); 5 usages  3 implementations
30
31      // Метод для создания копии объекта
32      Object clone() throws CloneNotSupportedException; 3 implementations
33
34      // Методы из интерфейса Function остаются (наследуются):
35      // double getLeftDomainBorder();
36      // double getRightDomainBorder();
37      // double getFunctionValue(double x);
38  }
```

Вывод Main:

```
=== ТЕСТИРОВАНИЕ МЕТОДОВ TABULATED FUNCTION ===

1. СОЗДАНИЕ ТЕСТОВЫХ ОБЪЕКТОВ:
Создано:
  - 3 объекта ArrayTabulatedFunction
  - 3 объекта LinkedListTabulatedFunction

2. ТЕСТ МЕТОДА toString():

ArrayTabulatedFunction arrayFunc1:
  {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

ArrayTabulatedFunction arrayFunc2:
  {(0.0; 0.0), (2.5; 0.0), (5.0; 0.0), (7.5; 0.0), (10.0; 0.0)}

LinkedListTabulatedFunction linkedListFunc1:
  {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

LinkedListTabulatedFunction linkedListFunc2:
  {(0.0; 0.0), (2.5; 0.0), (5.0; 0.0), (7.5; 0.0), (10.0; 0.0)}

3. ТЕСТ МЕТОДА equals():

Сравнение одинаковых объектов одного класса:
  arrayFunc1.equals(arrayFunc1): true
  arrayFunc1.equals(arrayFunc1Copy): true

Сравнение разных объектов одного класса:
  arrayFunc1.equals(arrayFunc2): false
```

Сравнение объектов разных классов с одинаковыми точками:

```
arrayFunc1.equals(linkedListFunc1Copy): true
```

Сравнение объектов разных классов с разными точками:

```
arrayFunc2.equals(linkedListFunc2): true
```

Сравнение с null:

```
arrayFunc1.equals(null): false
```

Сравнение с объектом другого типа:

```
arrayFunc1.equals("строка"): false
```

4. ТЕСТ МЕТОДА hashCode():

Хэш-коды объектов:

```
arrayFunc1.hashCode(): -14679889
```

```
arrayFunc1Copy.hashCode(): -14679889
```

```
arrayFunc2.hashCode(): -2133851993
```

```
linkedListFunc1.hashCode(): -14679889
```

```
linkedListFunc1Copy.hashCode(): -14679889
```

```
linkedListFunc2.hashCode(): -2133851993
```

Проверка согласованности equals() и hashCode():

```
arrayFunc1.equals(arrayFunc1Copy): true
```

```
arrayFunc1.hashCode() == arrayFunc1Copy.hashCode(): true
```

```
arrayFunc1.equals(arrayFunc2): false
```

```
arrayFunc1.hashCode() == arrayFunc2.hashCode(): false
```


Тест изменения объекта и хэш-кода:

Исходный хэш-код arrayFunc3: -17301337

Меняем координату Y у точки с индексом 2 с 3.0 на 3.001...

Новый хэш-код arrayFunc3: -1289035411

Хэш-коды различаются

5. ТЕСТ МЕТОДА clone():

Тест клонирования ArrayTabulatedFunction:

Исходный объект arrayFunc1: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

Клон arrayFunc1Clone: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

arrayFunc1 == arrayFunc1Clone: false

arrayFunc1.equals(arrayFunc1Clone): true

Проверка глубокого клонирования (ArrayTabulatedFunction):

Изменяем исходный объект arrayFunc1...

Исходный объект arrayFunc1 после изменения: {(1.0; 2.0), (2.0; 4.0), (3.0; 100.0), (4.0; 8.0), (5.0; 10.0)}

Клон arrayFunc1Clone после изменения оригинала: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

Клон изменился? true

Тест клонирования LinkedListTabulatedFunction:

Исходный объект linkedListFunc1: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

Клон linkedListFunc1Clone: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

linkedListFunc1 == linkedListFunc1Clone: false

linkedListFunc1.equals(linkedListFunc1Clone): true

Проверка глубокого клонирования (LinkedListTabulatedFunction):

Изменяем исходный объект linkedListFunc1...

Исходный объект linkedListFunc1 после изменения: {(1.0; 2.0), (2.0; 4.0), (3.0; 200.0), (4.0; 8.0), (5.0; 10.0)}

Клон linkedListFunc1Clone после изменения оригинала: {(1.0; 2.0), (2.0; 4.0), (3.0; 6.0), (4.0; 8.0), (5.0; 10.0)}

Клон изменился? true

6. ДОПОЛНИТЕЛЬНЫЕ ТЕСТЫ:

Тест с функцией из 2 точек:

smallFunc.toString(): {(0.0; 0.0), (1.0; 0.0)}

smallFunc.hashCode(): 2145386504

Тест сравнения ArrayTabulatedFunction и LinkedListTabulatedFunction:

identicalArrayFunc.toString(): {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

identicalLinkedListFunc.toString(): {(0.0; 1.0), (1.0; 2.0), (2.0; 3.0)}

identicalArrayFunc.equals(identicalLinkedListFunc): true

identicalLinkedListFunc.equals(identicalArrayFunc): true

=== ТЕСТИРОВАНИЕ ЗАВЕРШЕНО ===

Process finished with exit code 0