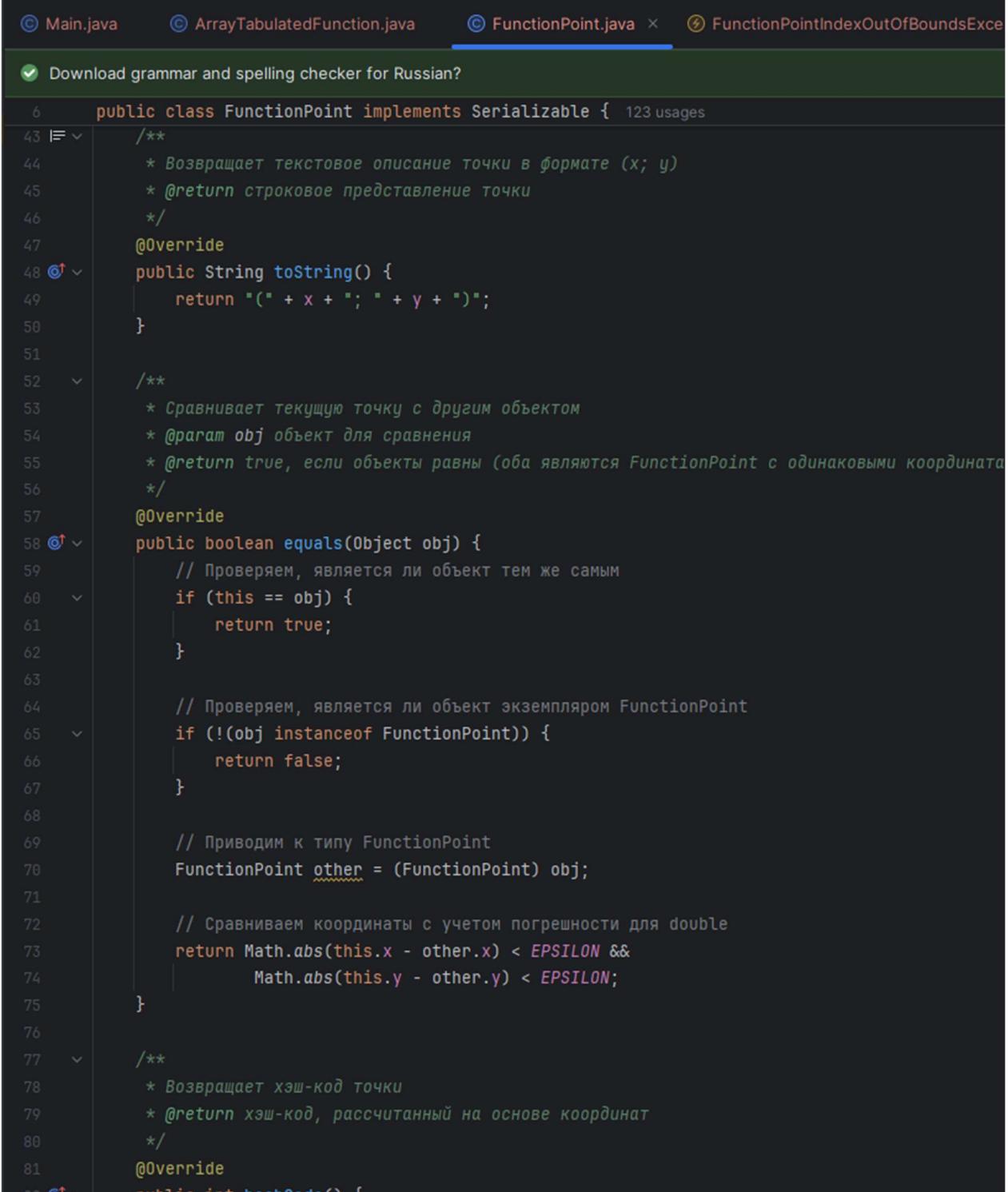


Лабораторная работа №5

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

Задание 1.

Переопределение методов в FunctionPoint.

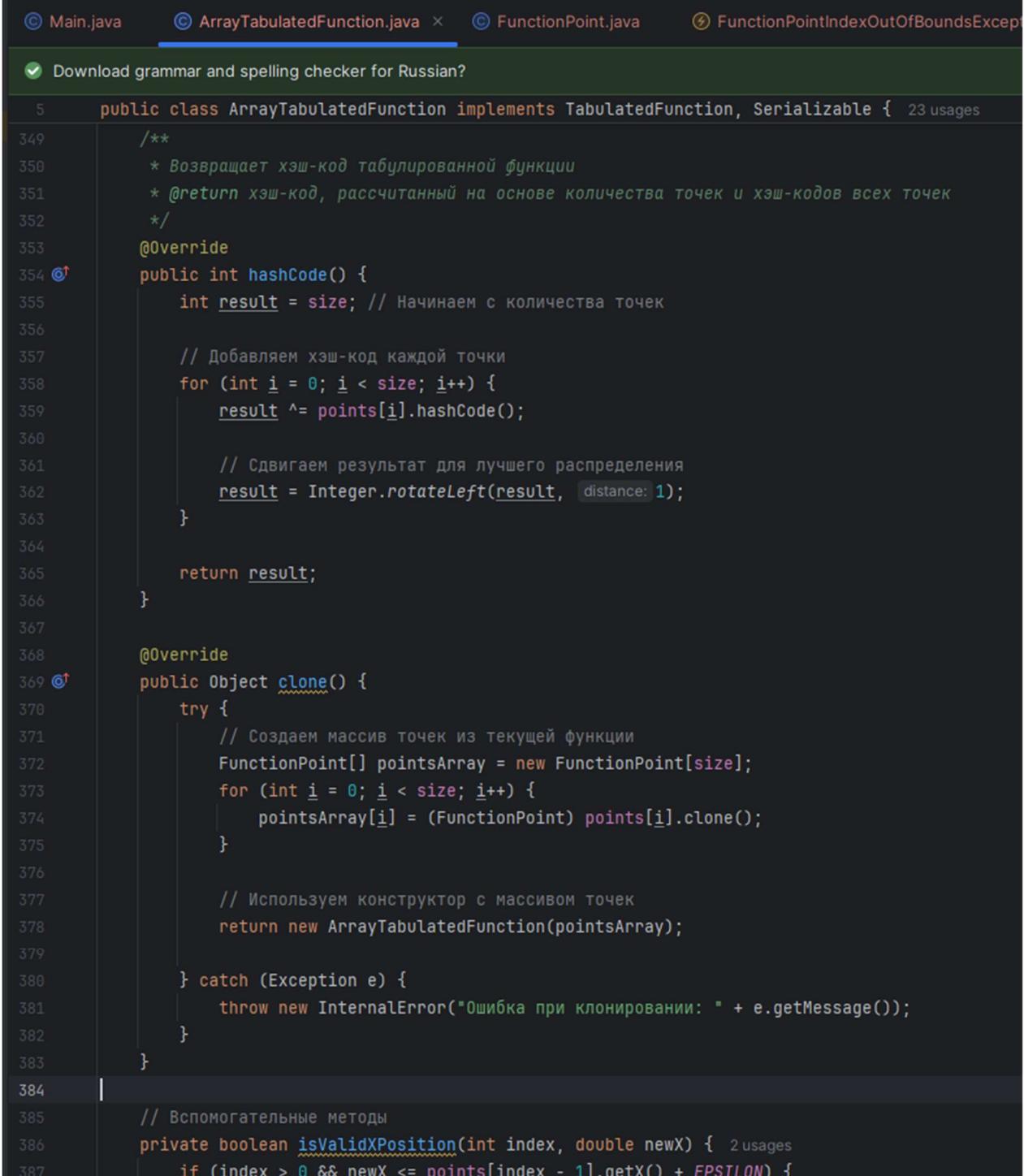


The screenshot shows a Java code editor with several tabs at the top: Main.java, ArrayTabulatedFunction.java, FunctionPoint.java (which is selected), and FunctionPointIndexOutOfBoundsException. A green bar at the top indicates a download option for a grammar and spelling checker for Russian. The code in FunctionPoint.java is as follows:

```
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.



The screenshot shows a Java code editor with four tabs at the top: Main.java, ArrayTabulatedFunction.java (selected), FunctionPoint.java, and FunctionPointIndexOutOfBoundsException. A green bar at the top has a checked checkbox for '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
380       } catch (Exception e) {
381           throw new InternalError("Ошибка при клонировании: " + e.getMessage());
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`.

The screenshot shows a code editor with several tabs at the top: `functionPoint.java`, `FunctionPointIndexOutOfBoundsException.java`, `InappropriateFunctionPointException.java`, `LinkedListTabulatedFunction.java` (which is the active tab), `Download Russian`, and `Never suggest Russia`. The code in the editor is for the `LinkedListTabulatedFunction` class, which implements `TabulatedFunction` and `Serializable`. The `hashCode()` method calculates a result by rotating the `FunctionPoint` values of the linked list. The `clone()` method creates a new list with cloned `FunctionPoint` objects.

```
437  /** and spelling checker for Russian? */
438  */
439  @Override
440  public int hashCode() {
441      int result = size; // Начинаем с количества точек
442
443      Node current = head;
444      while (current != null) {
445          result ^= current.point.hashCode();
446
447          // Сдвигаем результат для лучшего распределения
448          result = Integer.rotateLeft(result, distance: 1);
449          current = current.next;
450      }
451
452      return result;
453  }
454
455  /**
456      * Создает и возвращает глубокую копию текущей табулированной функции
457      * "Пересобирает" новый список без использования методов добавления
458      * @return копия объекта LinkedListTabulatedFunction
459  */
460  @Override
461  public Object clone() {
462      try {
463          if (size == 0) {
464              return new LinkedListTabulatedFunction(new FunctionPoint[0]);
465          }
466
467          // Создаем массив точек из текущего списка
468          FunctionPoint[] pointsArray = new FunctionPoint[size];
469          Node current = head;
470          int index = 0;
471
472          while (current != null) {
473              pointsArray[index] = (FunctionPoint) current.point.clone();
474              current = current.next;
475              index++;
476          }
477      }
478  }
```

Задание 4.

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

The screenshot shows a Java code editor with the file `TabulatedFunction.java` open. The interface defines methods for accessing and modifying function points, cloning the object, and printing the function. It also includes comments about methods from the `Function` interface.

```
1e.java Shift.java Composition.java Functions.java TabulatedFunctions.java Function.java TabulatedFunction.java
2
3  /**
4   * Интерфейс для табулированных функций (расширяет Function)
5   */
6  public interface TabulatedFunction extends Function, Cloneable {
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
```