САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И КИБЕРБЕЗОПАСНОСТИ

Дисциплина: Объектно-ориентированное программирование (Java)

Курсовая работа:
Приложение на Java — Паттерн «Стратегия», аннотации и рефлексия, переводчик, Stream API; GUI на Swing

Студент: *Момотова Алиса*, группа 30322

Руководитель: *Алексей Маслаков*

Санкт-Петербург — 2025

Содержание

Аннотация

В работе реализовано десктопное приложение на Java (Swing), объединяющее четыре самостоятельных задания:

1. Паттерн «Стратегия» для класса `Hero` (пешком/лошадь/полёт).
2. Аннотация `@Repeat(int)` и рефлексивный вызов аннотированных `protected`/`private` методов произвольной сигнатуры.
3. Переводчик с чтением словаря из файла формата `левая | перевод`, пользовательские исключения `InvalidFileFormatException`, `FileReadException`, правило «самого длинного совпадения».
4. Набор задач на Stream API.

Все подсистемы подключены к единому GUI (`JTabbedPane`); вывод ведётся в нередактируемые текстовые области. Для задания 3 предусмотрен выбор файлов словаря/текста; для задания 4 — ввод данных.

1 Введение

Цель — разработать модульное Java-приложение, демонстрирующее шаблоны проектирования, аннотации и рефлексию, обработку исключений, Stream API и интеграцию подсистем в единый GUI. Стек: Java 17, Gradle, Swing. Пакеты: `gui`, `hero`, `lab2`, `translator`, `streamApi` (или `org.example`).

2 Постановка задачи

2.1 Задание 1: Паттерн «Стратегия»

Класс `Hero` перемещается по стратегиям `MoveStrategy` (`Walk`, `Horse`, `Fly`). Метод `move` возвращает описание действия.

2.2 Задание 2: Аннотация и рефлексия

Определить `@Repeat(int)`; создать класс с публичными/защищёнными/приватными методами и аннотировать часть из них. Из другого класса вызвать *только* `protected`/`private` аннотированные методы требуемое число раз, подставляя дефолтные аргументы.

2.3 Задание 3: Переводчик

Словарь UTF-8: `левая | перевод`. Перевод построчно, регистр игнорируется; при множестве кандидатов — берётся *самый длинный* слева. Исключения: `InvalidFileFormatExceptio` (формат), `FileReadException` (файл).

2.4 Задание 4: Stream API

Методы: среднее `List<Integer>`; строки в верхний регистр с префиксом `_new_`; квадраты элементов, встречающихся ровно один раз; последний элемент коллекции (или исключение); сумма чётных чисел массива; преобразование списка строк в `Map<Character,String>`.

3 Архитектура и диаграмма классов

Пакеты

- **gui**: `App` — точка входа (единственный `main`).
- **hero**: `Hero`, `Position`, `MoveStrategy`, `Walk`, `Horse`, `Fly`.
- **lab2**: `Repeat` (аннотация), `SampleService`, `Invoker`.

- **translator**: `Dictionary`, `DictionaryLoader`, `Translator`, `InvalidFileFormatException`, `FileReadException`.
- **streamApi/org.example**: `StreamTasks`.

Диаграмма классов

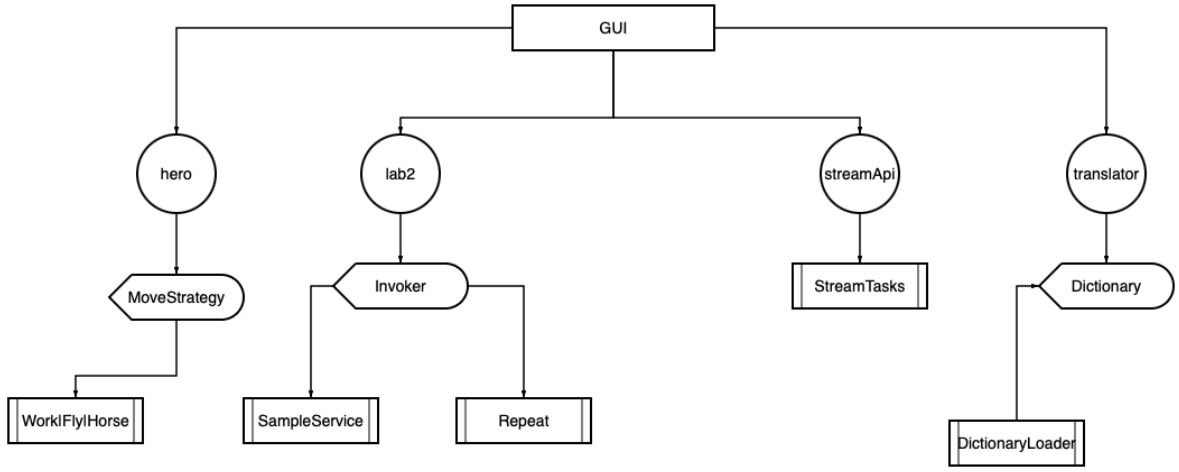

Рис. 1: Enter Caption

4 Реализация

4.1 Задание 1: «Стратегия»

Идея. `Hero` зависит от абстракции `MoveStrategy`. Переключение через `setStrategy`. `move` делегирует стратегии и возвращает строку для GUI.

4.2 Задание 2: Аннотация и рефлексия

Аннотация. `@Repeat(int)`: `RetentionPolicy.RUNTIME`, `Target(ElementType.METHOD)`.

Инвокер. `Invoker` находит методы с `@Repeat`, оставляет `protected`/`private`, делает `setAccessible(true)`, генерирует дефолтные аргументы и вызывает нужное число раз, логируя результаты.

4.3 Задание 3: Переводчик

Формат. «левая | перевод», строки с # игнорируются. Ошибки формата/чтения — пользовательские исключения.

Алгоритм. Токенизация по пробелам, на каждой позиции — поиск самого длинного совпадения по ключам словаря (регистронезависимо).

4.4 Задание 4: Stream API

Методы реализованы стандартными операциями потоков: `map/mapToInt/filter/reduce/group`

5 Графический интерфейс (Swing)

Главное окно — `JFrame` + `JTabbedPane`. Для каждой задачи отдельная вкладка; все результаты — в `JTextArea` с `setEditable(false)`. Для переводчика — выбор словаря/текста (`JFileChooser`); для Stream API — поля ввода.

6 Сборка и запуск

6.1 Структура проекта

```
coursework-java/
 app/
    build.gradle
    src/main/java/
```

```
gui/          (App.java - единственный main)
hero/         (Hero, MoveStrategy, Walk, Horse, Fly, Position)
lab2/         (Repeat, SampleService, Invoker)
translator/   (Dictionary, DictionaryLoader, Translator, Exceptions)
streamApi/    (StreamTasks)  # или org/example/StreamTasks
```

6.2 Gradle

```
plugins { id 'application' }
application { mainClass = 'gui.App' }
```

Сборка и запуск:

```
cd coursework-java/app
./gradlew clean build
./gradlew run
```

7 Перечень выполненных работ

1. Реализован паттерн «Стратегия» для перемещения героя.
2. Создана аннотация `@Repeat` и рефлексивный инвокер.
3. Разработан переводчик с пользовательскими исключениями и правилом длиннейшего совпадения.
4. Имплементированы шесть методов на Stream API.
5. Создан общий GUI (Swing) с вкладками и вводом/выводом.
6. Подготовлены инструкции по сборке/запуску Gradle-проекта.
7. Построена UML-диаграмма.

8 Заключение

Показано применение интерфейсов и шаблонов, аннотаций и рефлексии, потоковых операций, исключений и GUI на Swing. Архитектура модульна, что упрощает демонстрацию и дальнейшее расширение.

Приложение: примеры кода

Задание 1 — Паттерн «Стратегия»

```
package hero;

public class Hero {
    private MoveStrategy strategy;
    public Hero(MoveStrategy s) { this.strategy = s; }
    public void setStrategy(MoveStrategy s) { this.strategy = s; }
    public String move(Position from, Position to) {
        return "[%s] %s".formatted(strategy.name(), strategy.move(from,
   to));
    }
}
```

Листинг 1: Класс Hero

```
package hero;

public interface MoveStrategy {
    String move(Position from, Position to);
    default String name() { return getClass().getSimpleName(); }
}
```

Листинг 2: Интерфейс MoveStrategy

```
package hero;
public class Fly implements MoveStrategy {
    @Override public String move(Position from, Position to) {
        return "              %s    %s".formatted(from, to);
    }
}
package hero;
public class Horse implements MoveStrategy {
    @Override public String move(Position from, Position to) {
        return "                                  %s    %s".formatted(from,
   to);
    }
}
package hero;
public class Walk implements MoveStrategy {
    @Override public String move(Position from, Position to) {
        return "                           %s    %s".formatted(from, to);
    }
}
```

Листинг 3: Классы Walk, Horse, Fly

```java
package hero;
public record Position(int x, int y) {}
```

Листинг 4: Класс позиции

Задание 2 — Аннотация и рефлексия

```java
package lab2;

import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Repeat {
    int value();
}
```

Листинг 5: Аннотация Repeat

```java
package lab2;

import java.lang.reflect.*;
import java.util.ArrayList;
import java.util.List;

public final class Invoker {
    public static List<String> run(Object target) {
        List<String> logs = new ArrayList<>();
        for (Method m : target.getClass().getDeclaredMethods()) {
            Repeat rep = m.getAnnotation(Repeat.class);
            if (rep == null) continue;
            int mod = m.getModifiers();
            if (!(Modifier.isProtected(mod) || Modifier.isPrivate(mod)))
    continue;

            m.setAccessible(true);
            Object[] args = defaultsFor(m.getParameterTypes());

            for (int i = 0; i < rep.value(); i++) {
                try {
                    Object r = m.invoke(target, args);
                    logs.add(m.getName() + " #" + (i+1) + " -> " +
   String.valueOf(r));
                } catch (InvocationTargetException ite) {
                    logs.add(m.getName() + " threw: " + ite.
   getTargetException());
```

```
                } catch (Exception e) {
                    logs.add(m.getName() + " ERROR: " + e);
                }
            }
        }
        return logs;
    }

    public static List<String> runAll() {
        return run(new SampleService());
    }

    private static Object[] defaultsFor(Class<?>[] types) {
        Object[] a = new Object[types.length];
        for (int i = 0; i < types.length; i++) a[i] = defaultFor(types[i
    ]);
        return a;
    }
    private static Object defaultFor(Class<?> t) {
        if (t.isPrimitive()) {
            if (t == int.class) return 0;
            if (t == long.class) return 0L;
            if (t == boolean.class) return false;
            if (t == double.class) return 0.0d;
            if (t == float.class) return 0.0f;
            if (t == short.class) return (short)0;
            if (t == byte.class) return (byte)0;
            if (t == char.class) return '\0';
        } else {
            if (t == String.class) return "";
            if (t == Integer.class) return 0;
            if (t == Long.class) return 0L;
            if (t == Boolean.class) return false;
            if (t == Double.class) return 0.0d;
            if (t == Float.class) return 0.0f;
            if (t == Short.class) return (short)0;
            if (t == Byte.class) return (byte)0;
            if (t == Character.class) return '\0';
            if (t.isArray()) return Array.newInstance(t.getComponentType
    (), 0);
        }
        return null;
    }

    private Invoker() {}
}
```

Листинг 6: Класс Invoker

```
package lab2;

public class SampleService {
    public void pub1(int x) { System.out.println("pub1:"+x); }
    public String pub2(String s, boolean b) { String r=s+"|"+b; System.
   out.println("pub2:"+r); return r; }

    @Repeat(2)
    protected void prot1(int a, String s) { System.out.println("prot1:"+
   a+","+s); }

    @Repeat(3)
    protected int prot2() { System.out.println("prot2"); return 42; }

    @Repeat(4)
    private void priv1(double d) { System.out.println("priv1:"+d); }

    @Repeat(1)
    private String priv2(Integer v, Object o) { String r="priv2:"+v+","
   +(o==null?"null":o.toString()); System.out.println(r); return r; }
}
```

Листинг 7: Класс SampleService

Задание 3 — Переводчик

```
package translator;

import java.util.Collections;
import java.util.Map;

public class Dictionary {
    private final Map<String, String> map;
    private final int maxPhraseWordLen;

    public Dictionary(Map<String, String> map, int maxPhraseWordLen) {
        this.map = map;
        this.maxPhraseWordLen = maxPhraseWordLen;
    }

    public Map<String, String> getMap() {
        return Collections.unmodifiableMap(map);
    }
```

```
18
19      public int getMaxPhraseWordLen() {
20          return maxPhraseWordLen;
21      }
22  }
```

Листинг 8: Класс Dictionary

```
1   package translator;
2   import java.io.BufferedReader;
3   import java.io.IOException;
4   import java.nio.charset.StandardCharsets;
5   import java.nio.file.Files;
6   import java.nio.file.Path;
7   import java.util.LinkedHashMap;
8   import java.util.Locale;
9   import java.util.Map;
10
11  public class DictionaryLoader {
12
13      public static Dictionary load(Path path) throws FileReadException,
        InvalidFileFormatException {
14          Map<String, String> map = new LinkedHashMap<>();
15          int maxWords = 1;
16
17          try (BufferedReader br = Files.newBufferedReader(path,
        StandardCharsets.UTF_8)) {
18              String line;
19              int lineNo = 0;
20              while ((line = br.readLine()) != null) {
21                  lineNo++;
22                  if (lineNo == 1) {
23                      line = stripBOM(line);
24                  }
25                  String raw = line.trim();
26                  if (raw.isEmpty() || raw.startsWith("#")) {
27                      continue;
28                  }
29
30                  int bar = raw.indexOf('|');
31                  if (bar < 0) {
32                      throw new InvalidFileFormatException(
33                          "
                      " + lineNo + " (                                    '|'): " +
        line);
34                  }
35
36                  String left = raw.substring(0, bar).trim();
```

```
37                  String right = raw.substring(bar + 1).trim();
38
39                  if (left.isEmpty() || right.isEmpty()) {
40                      throw new InvalidFileFormatException(
41                              "
                       " + lineNo + " (                         /
                  ).");
42                  }
43
44                  String normKey = normalizeKey(left);
45                  map.put(normKey, right);
46
47                  int words = left.trim().split("\\s+").length;
48                  if (words > maxWords) maxWords = words;
49              }
50          } catch (IOException ioe) {
51              throw new FileReadException("
                                                : " + path + " (" + ioe.
    getMessage() + ")", ioe);
52          } catch (SecurityException se) {
53              throw new FileReadException("
                                : " + path, se);
54          }
55
56          if (map.isEmpty()) {
57              throw new InvalidFileFormatException("

    .");
58          }
59          return new Dictionary(map, maxWords);
60      }
61
62      static String normalizeKey(String s) {
63
64          return s.trim().replaceAll("\\s+", " ").toLowerCase(Locale.ROOT)
    ;
65      }
66
67      private static String stripBOM(String s) {
68          if (s != null && !s.isEmpty() && s.charAt(0) == '\uFEFF') {
69              return s.substring(1);
70          }
71          return s;
72      }
73  }
```

Листинг 9: Класс DictionaryLoader

```java
package translator;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Translator {
    private final Map<String, String> dict;
    private final int maxLen;

    private static final Pattern WORD = Pattern.compile("[\\p{L}\\p{M}\\p{N}']+");

    public Translator(Dictionary dictionary) {
        this.dict = dictionary.getMap();
        this.maxLen = dictionary.getMaxPhraseWordLen();
    }

    public String translateLine(String input) {
        if (input == null || input.isEmpty()) return input;

        List<WordSpan> words = new ArrayList<>();
        Matcher m = WORD.matcher(input);
        while (m.find()) {
            words.add(new WordSpan(m.start(), m.end(), input.substring(m.start(), m.end())));
        }

        StringBuilder out = new StringBuilder(input.length() + 32);
        int pos = 0;

        for (int i = 0; i < words.size(); i++) {
            WordSpan w = words.get(i);
            if (pos < w.start) {
                out.append(input, pos, w.start);
                pos = w.start;
            }
            int bestK = 0;
            String bestTrans = null;

            int limit = Math.min(maxLen, words.size() - i);
            for (int k = limit; k >= 1; k--) {
                if (!onlyWhitespaceBetween(input, words, i, k)) {
```

```
                    continue;
                }
                String key = buildKeyLower(words, i, k);
                String trans = dict.get(key);
                if (trans != null) {
                    bestK = k;
                    bestTrans = trans;
                    break;
                }
            }

            if (bestK > 0) {
                out.append(bestTrans);
                pos = words.get(i + bestK - 1).end;
                i += (bestK - 1);
            } else {
                out.append(w.text);
                pos = w.end;
            }
        }
        if (pos < input.length()) {
            out.append(input, pos, input.length());
        }

        return out.toString();
    }

    private static boolean onlyWhitespaceBetween(String input, List<
    WordSpan> words, int i, int k) {
        for (int t = i; t < i + k - 1; t++) {
            int a = words.get(t).end;
            int b = words.get(t + 1).start;
            if (a > b) return false;
            String between = input.substring(a, b);
            if (!between.chars().allMatch(Character::isWhitespace)) {
                return false;
            }
        }
        return true;
    }

    private static String buildKeyLower(List<WordSpan> words, int i, int
    k) {
        StringBuilder sb = new StringBuilder();
        for (int t = 0; t < k; t++) {
            if (t > 0) sb.append(' ');
            sb.append(words.get(i + t).text.toLowerCase(Locale.ROOT));
```

```
        }
        return sb.toString();
    }

    private static final class WordSpan {
        final int start, end;
        final String text;
        WordSpan(int s, int e, String txt) { start = s; end = e; text =
    txt; }
    }
}
```

Листинг 10: Класс Translator

```
package translator;

public class FileReadException extends Exception {
    public FileReadException(String message) { super(message); }
    public FileReadException(String message, Throwable cause) { super(
    message, cause); }
}
package translator;

public class InvalidFileFormatException extends Exception {
    public InvalidFileFormatException(String message) { super(message);
    }
    public InvalidFileFormatException(String message, Throwable cause) {
    super(message, cause); }
}
```

Листинг 11: Исключения

Задание 4 — Stream API

```
package streamapi;

import java.util.*;
import java.util.stream.Collectors;

public class StreamTasks {
    public static double average(List<Integer> numbers) {
        return numbers.stream()
                .mapToInt(Integer::intValue)
                .average()
                .orElseThrow(() -> new IllegalArgumentException("
                        "));
    }
```

```
    public static List<String> transformStrings(List<String> strings) {
        return strings.stream()
                .map(s -> "_new_" + s.toUpperCase())
                .collect(Collectors.toList());
    }
    public static List<Integer> uniqueSquares(List<Integer> numbers) {
        return numbers.stream()
                .filter(n -> Collections.frequency(numbers, n) == 1)
                .map(n -> n * n)
                .collect(Collectors.toList());
    }
    public static <T> T getLast(Collection<T> collection) {
        return collection.stream()
                .reduce((first, second) -> second)
                .orElseThrow(() -> new NoSuchElementException("
                                ));
    }
    public static int sumEven(int[] arr) {
        return Arrays.stream(arr)
                .filter(n -> n % 2 == 0)
                .sum();
    }
    public static Map<Character, String> toMap(List<String> strings) {
        return strings.stream()
                .collect(Collectors.toMap(
                        s -> s.charAt(0),
                        s -> s.substring(1),
                        (v1, v2) -> v1
                ));
    }
}
```

Листинг 12: Класс StreamTasks

Графический интерфейс (Swing)

```
package gui;

import hero.*;
import lab2.Invoker;
import streamapi.StreamTasks;
import translator.Dictionary;
import translator.DictionaryLoader;
import translator.FileReadException;
import translator.InvalidFileFormatException;
import translator.Translator;
```

```
11 import java.util.Arrays;
12 import java.util.List;
13 import java.util.stream.Collectors;
14
15 import javax.swing.*;
16 import javax.swing.border.EmptyBorder;
17 import java.awt.*;
18 import java.io.IOException;
19 import java.nio.file.Path;
20
21 public class App{
22     public static void main(String[] args) {
23         SwingUtilities.invokeLater(App::createAndShow);
24     }
25
26     static void createAndShow() {
27         JFrame f = new JFrame("                :                1  4  ");
28         f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
29         f.setSize(900, 650);
30
31         JTabbedPane tabs = new JTabbedPane();
32         tabs.add("1. Hero (Strategy)", heroTab());
33         tabs.add("2.                  ", annotationTab());
34         tabs.add("3.                    ", translatorTab());
35         tabs.add("4. Stream API", streamTab());
36
37         f.setContentPane(tabs);
38         f.setLocationRelativeTo(null);
39         f.setVisible(true);
40     }
41
42     // ---------- TAB 1: Strategy ----------
43     static JPanel heroTab() {
44         JPanel p = panel();
45         JTextArea out = readonlyArea();
46         Position from = new Position(0, 0);
47         Position to   = new Position(10, 5);
48         Hero hero = new Hero(new Walk());
49
50         ButtonGroup g = new ButtonGroup();
51         JRadioButton rbWalk = new JRadioButton("            ", true);
52         JRadioButton rbHorse = new JRadioButton("             ");
53         JRadioButton rbFly = new JRadioButton("          ");
54         g.add(rbWalk); g.add(rbHorse); g.add(rbFly);
55
56         JButton moveBtn = new JButton("                          (0,0)
       (10,5)");
```

```
57          moveBtn.addActionListener(e -> {
58              if (rbWalk.isSelected())  hero.setStrategy(new Walk());
59              if (rbHorse.isSelected()) hero.setStrategy(new Horse());
60              if (rbFly.isSelected())   hero.setStrategy(new Fly());
61              String msg = hero.move(from, to);
62              out.append(msg + "\n");
63          });
64
65          p.add(row(rbWalk, rbHorse, rbFly, moveBtn));
66          p.add(scroll(out));
67          return p;
68      }
69
70      // ---------- TAB 2:                        ----------
71      static JPanel annotationTab() {
72          JPanel p = panel();
73          JTextArea out = readonlyArea();
74          JButton run = new JButton("
                                protected/private              ");
75          run.addActionListener(e -> {
76              List<String> logs = Invoker.runAll();
77              logs.forEach(s -> out.append(s + "\n"));
78          });
79          p.add(row(run));
80          p.add(scroll(out));
81          return p;
82      }
83
84      // ---------- TAB 3:                           ----------
85      static JPanel translatorTab() {
86          JPanel p = panel();
87          JTextArea dictPath = readonlyArea(); dictPath.setText("
                    : (                  )");
88          JTextArea input = new JTextArea(8, 80);
89          JTextArea output = readonlyArea();
90
91          JButton pickDict = new JButton("                                ...
    ");
92          final Dictionary[] holder = new Dictionary[1];
93
94          pickDict.addActionListener(e -> {
95              JFileChooser fc = new JFileChooser();
96              if (fc.showOpenDialog(p) == JFileChooser.APPROVE_OPTION) {
97                  Path path = fc.getSelectedFile().toPath();
98                  try {
99                      holder[0] = DictionaryLoader.load(path);
100                     dictPath.setText("               : " + path);
```

```
                } catch (InvalidFileFormatException | FileReadException ex) {
                    output.append("            : " + ex.getMessage() + "\n");
                }
            }
        });

        JButton pickText = new JButton("
                ...");
        pickText.addActionListener(e -> {
            JFileChooser fc = new JFileChooser();
            if (fc.showOpenDialog(p) == JFileChooser.APPROVE_OPTION) {
                try {
                    String txt = java.nio.file.Files.readString(fc.getSelectedFile().toPath());
                    input.setText(txt);
                } catch (IOException ex) {
                    output.append("
                : " + ex.getMessage() + "\n");
                }
            }
        });

        JButton translate = new JButton("                        ");
        translate.addActionListener(e -> {
            output.setText("");
            if (holder[0] == null) {
                output.append("
                  .\n");
                return;
            }
            Translator tr = new Translator(holder[0]);
            String[] lines = input.getText().split("\\R");
            for (String line : lines) {
                output.append(tr.translateLine(line) + "\n");
            }
        });

        p.add(row(pickDict, pickText, translate));
        p.add(scroll(dictPath));
        p.add(labeled("         (            /               ):", new JScrollPane(input)));
        p.add(labeled("                   (read-only):", scroll(output)));
        return p;
     }
```

```
140
141     // ---------- TAB 4: Stream API ----------
142     static JPanel streamTab() {
143         JPanel p = panel();
144         JTextField intsField = new JTextField("1,2,2,3,4", 40);
145         JTextField strsField = new JTextField("apple,banana,cherry", 40)
    ;
146         JTextArea out = readonlyArea();
147
148         JButton run = new JButton("                                    ");
149         run.addActionListener(e -> {
150             try {
151                 List<Integer> nums = Arrays.stream(intsField.getText().
    split("\\s*,\\s*"))
152                         .filter(s -> !s.isBlank())
153                         .map(Integer::parseInt).collect(Collectors.
    toList());
154                 List<String> strs = Arrays.stream(strsField.getText().
    split("\\s*,\\s*"))
155                         .filter(s -> !s.isBlank()).toList();
156
157                 out.setText("");
158                 out.append("              : " + StreamTasks.average(nums
    ) + "\n");
159                 out.append("                                    : " +
    StreamTasks.transformStrings(strs) + "\n");
160                 out.append("                                       : " +
    StreamTasks.uniqueSquares(nums) + "\n");
161                 out.append("                                     : " +
    StreamTasks.getLast(strs) + "\n");
162                 int[] arr = nums.stream().mapToInt(Integer::intValue).
    toArray();
163                 out.append("                         : " + StreamTasks.
    sumEven(arr) + "\n");
164                 out.append("Map                 : " + StreamTasks.toMap(
    strs) + "\n");
165             } catch (Exception ex) {
166                 out.append("            : " + ex.getMessage() + "\n");
167             }
168         });
169
170         p.add(labeled("                        (
                ):", intsField));
171         p.add(labeled("                        (
                ):", strsField));
172         p.add(row(run));
173         p.add(scroll(out));
```

```
174         return p;
175     }
176 
177     // ------- UI                  -------
178     static JPanel panel() {
179         JPanel p = new JPanel();
180         p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
181         p.setBorder(new EmptyBorder(12,12,12,12));
182         return p;
183     }
184     static JScrollPane scroll(JTextArea ta) {
185         JScrollPane sp = new JScrollPane(ta);
186         sp.setPreferredSize(new Dimension(820, 240));
187         return sp;
188     }
189     static JTextArea readonlyArea() {
190         JTextArea ta = new JTextArea(8, 80);
191         ta.setEditable(false);
192         ta.setLineWrap(true);
193         ta.setWrapStyleWord(true);
194         return ta;
195     }
196     static JPanel row(Component... cs) {
197         JPanel r = new JPanel(new FlowLayout(FlowLayout.LEFT));
198         for (Component c : cs) r.add(c);
199         return r;
200     }
201     static JPanel labeled(String title, Component c) {
202         JPanel r = new JPanel();
203         r.setLayout(new BorderLayout(8,8));
204         r.add(new JLabel(title), BorderLayout.NORTH);
205         r.add(c, BorderLayout.CENTER);
206         return r;
207     }
208 }
```

Листинг 13: Главное окно App.java

Подключение исходников