Русанов Александр Юрьевич ПА-01

Последовательность

Первым делом разработал модель данных Earthquake. Реализовал класс CSVReader с поддержкой кодировки UTF-8. Для анализа данных создал класс EarthquakeAnalyzer. Реализовал сбор статистики: общее количество землетрясений, средние/максимальные/минимальные значения магнитуды и глубины, распределение по штатам, категоризация по магнитудам и глубине.

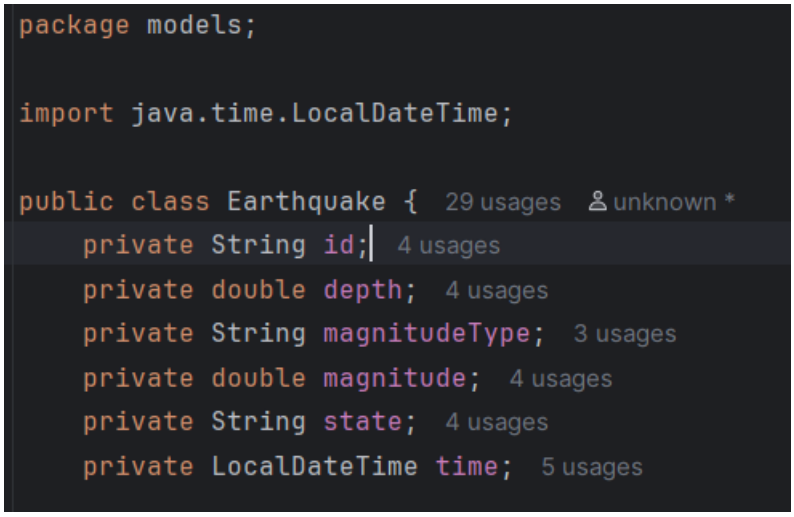
Для визуализации результатов разработал TextChartGenerator. Создал DatabaseManager для работы с SQLite. Для выполнения аналитических запросов создал класс SQLQueries. В классе Main интегрировал все компоненты в логическую последовательность.

Earthquake.java

Этот файл является основой всей системы - он определяет структуру данных, с которой работает вся программа. Это классический POJO-класс, который инкапсулирует все атрибуты землетрясения.

Это фундамент всего проекта, файл написанный первым

Класс Earthquake - Это "контейнер для данных" - просто хранит информацию об одном землетрясении.



```java
public Earthquake() {}  no usages  unknown

public Earthquake(String id, double depth, String magnitudeType,  1 usage  unknown
                  double magnitude, String state, LocalDateTime time) {
    this.id = id;
    this.depth = depth;
    this.magnitudeType = magnitudeType;
    this.magnitude = magnitude;
    this.state = state;
    this.time = time;
}
```

```java
public String getId() { return id; }  7 usages  unknown
public void setId(String id) { this.id = id; }  no usages  unknown

public double getDepth() { return depth; }  10 usages  unknown
public void setDepth(double depth) { this.depth = depth; }  no usages  unknown

public String getMagnitudeType() { return magnitudeType; }  1 usage  unknown
public void setMagnitudeType(String magnitudeType) { this.magnitudeType = magnitudeType; }  no usages

public double getMagnitude() { return magnitude; }  7 usages  unknown
public void setMagnitude(double magnitude) { this.magnitude = magnitude; }  no usages  unknown

public String getState() { return state; }  22 usages  unknown
public void setState(String state) { this.state = state; }  no usages  unknown

public LocalDateTime getTime() { return time; }  17 usages  unknown
public void setTime(LocalDateTime time) { this.time = time; }  no usages  unknown

@Override  unknown
public String toString() {
    String timeStr = (time != null) ?
            time.format(java.time.format.DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")) :
            "Нет данных";

    return String.format("Earthquake{id='%s', magnitude=%.2f, depth=%.0f, state='%s', time=%s}",
            id, magnitude, depth, state, timeStr);
}
}
```

EarthquakeAnalyzer.java

Этот класс отвечает за всю логику анализа данных. Он работает с коллекцией объектов Earthquake и предоставляет различные статистические методы.

Файл был написан третьим

Класс EarthquakeAnalyzer - Берет список землетрясений и вычисляет всевозможные статистики

```java
package models;

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

public class EarthquakeAnalyzer {  3 usages  unknown
    private List<Earthquake> earthquakes;  22 usages

    public EarthquakeAnalyzer() {  1 usage  unknown
        this.earthquakes = new ArrayList<>();
    }

    public void addEarthquake(Earthquake earthquake) {  1 usage  unknown
        earthquakes.add(earthquake);
    }

    public List<Earthquake> getEarthquakes() {  no usages  unknown
        return earthquakes;
    }
```

```java
    public Map<String, Object> getStatistics() {  1 usage  unknown
        Map<String, Object> stats = new LinkedHashMap<>();

        stats.put("Всего землетрясений", earthquakes.size());

        if (!earthquakes.isEmpty()) {
            // Статистика по магнитудам
            DoubleSummaryStatistics magnitudeStats = earthquakes.stream()  Stream<Earthquake>
                    .mapToDouble(Earthquake::getMagnitude)  DoubleStream
                    .summaryStatistics();

            // Статистика по глубине
            DoubleSummaryStatistics depthStats = earthquakes.stream()  Stream<Earthquake>
                    .filter( Earthquake eq -> eq.getDepth() > 0)
                    .mapToDouble(Earthquake::getDepth)  DoubleStream
                    .summaryStatistics();

            // Статистика по времени
            long withTime = earthquakes.stream()
                    .filter( Earthquake eq -> eq.getTime() != null)
                    .count();
```

```
// Заполняем статистику
stats.put("Средняя магнитуда", magnitudeStats.getAverage());
stats.put("Максимальная магнитуда", magnitudeStats.getMax());
stats.put("Минимальная магнитуда", magnitudeStats.getMin());
stats.put("Средняя глубина (м)", depthStats.getCount() > 0 ? depthStats.getAverage() : 0);
stats.put("Максимальная глубина (м)", depthStats.getCount() > 0 ? depthStats.getMax() : 0);
stats.put("С временем", withTime);
stats.put("Без времени", earthquakes.size() - withTime);
stats.put("Процент с временем", earthquakes.size() > 0 ?
        String.format("%.1f%%", (withTime * 100.0 / earthquakes.size())) : "0%");
stats.put("Уникальных штатов", stateCounts.size());
stats.put("Самый частый штат", topState);
```

```
  return distribution.entrySet().stream()
          .sorted(( Entry<String, Long> a,  Entry<String, Long> b) -> {
              String[] order = {"< 2.0", "2.0 - 2.9", "3.0 - 3.9", "4.0 - 4.9", "5.0 - 5.9", ">= 6.0"};
              int indexA = Arrays.asList(order).indexOf(a.getKey());
              int indexB = Arrays.asList(order).indexOf(b.getKey());
              return Integer.compare(indexA, indexB);
          })
          .collect(Collectors.toMap(
                  Map.Entry::getKey,
                  Map.Entry::getValue,
                  ( Long v1,  Long v2) -> v1,
                  LinkedHashMap::new));
```

```
public Map<String, Long> getDepthDistribution() {  1 usage   unknown
    return earthquakes.stream()  Stream<Earthquake>
            .filter( Earthquake eq -> eq.getDepth() > 0)
            .collect(Collectors.groupingBy(
                     Earthquake eq -> {
                        double depth = eq.getDepth();
                        if (depth < 5000) return "Мелкие (< 5 км)";
                        else if (depth < 10000) return "Средние (5-10 км)";
                        else if (depth < 20000) return "Глубокие (10-20 км)";
                        else if (depth < 50000) return "Очень глубокие (20-50 км)";
                        else return "Экстремальные (> 50 км)";
                    },
                    Collectors.counting())) Map<String, Long>
            .entrySet().stream()  Stream<Entry<...>>
            .sorted(( Entry<String, Long> a,  Entry<String, Long> b) -> {
                String[] order = {
```

```
                    String[] order = {
                            "Мелкие (< 5 км)", "Средние (5-10 км)", "Глубокие (10-20 км)",
                            "Очень глубокие (20-50 км)", "Экстремальные (> 50 км)"
                    };
                    int indexA = Arrays.asList(order).indexOf(a.getKey());
                    int indexB = Arrays.asList(order).indexOf(b.getKey());
                    return Integer.compare(indexA, indexB);
                })
                .collect(Collectors.toMap(
                        Map.Entry::getKey,
                        Map.Entry::getValue,
                        ( Long v1,  Long v2) -> v1,
                        LinkedHashMap::new));
    }
```

CSVReader.java

Этот класс отвечает за чтение и преобразование данных из CSV-файла в объекты Java. Файл написанный вторым.

Класс CSVReader - Это переводчик с языка CSV на язык Java. Берет текстовый файл с запятыми и превращает его в список объектов Earthquake

```
package parser;

import models.Earthquake;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.*;
```

```java
public class CSVReader {  3 usages  unknown *
    public List<Earthquake> readCSV(String filename) {  1 usage  unknown *
        List<Earthquake> earthquakes = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(
                new InputStreamReader(new FileInputStream(filename), StandardCharsets.UTF_8))) {

            System.out.println("=== Начало чтения CSV файла ===");

            // Читаем заголовок для отладки
            String header = br.readLine();
            if (header != null) {
                System.out.println("Заголовок файла: " + header);
                System.out.println("Колонки: " + Arrays.toString(header.split( regex: ",")));
            }
```

```java
            String line; // Объявляем переменную здесь
            while ((line = br.readLine()) != null) {
                lineNum++;
                try {
                    Earthquake eq = parseLine(line);
                    if (eq != null) {
                        earthquakes.add(eq);
                        successCount++;

                    } else {
                        errorCount++;
                        if (errorCount <= 3) {
                            System.err.println("Не удалось распарсить строку " + lineNum + ": " +
                                    (line.length() > 100 ? line.substring(0, 100) + "..." : line));
                        }
                    }
                } catch (Exception e) {
                    errorCount++;
                    if (errorCount <= 3) {
                        System.err.println("Ошибка в строке " + lineNum + ": " + e.getMessage());
                    }
                }
            }
```

```java
private Earthquake parseLine(String line) {  1 usage  unknown
    try {
        // Парсинг CSV
        String[] parts = parseCSVLine(line);

        // Вывод отладки для первой строки
        if (parts.length < 6) {
            System.err.println("Недостаточно колонок в строке: " + parts.length +
                    " (ожидается минимум 6). Строка: " +
                    (line.length() > 50 ? line.substring(0, 50) + "..." : line));
            return null;
        }

        // Парсим каждое поле с проверкой
        String id = parts[0].trim();
        if (id.isEmpty()) {
            id = "UNKNOWN-" + UUID.randomUUID().toString().substring(0, 8);
        }

        double depth = parseDouble(parts[1]);
        String magnitudeType = parts[2].trim();
        double magnitude = parseDouble(parts[3]);
        String state = parts[4].trim().replace( target: "\"",  replacement: "");

        String timeStr = parts[5].trim();
        LocalDateTime time = parseDateTime(timeStr);
```

```java
// Создаем объект землетрясения
Earthquake eq = new Earthquake(id, depth, magnitudeType, magnitude, state, time);
```

DatabaseManager.java

Управляет подключением к базе данных, создает таблицы и сохраняет данные.

Класс DatabaseManager - Берет данные из оперативной памяти и аккуратно складывает в базу данных для долгосрочного хранения.

Файл был написан четвертым

```java
package database;

import models.Earthquake;
import java.sql.*;
import java.util.List;
```

```java
public DatabaseManager(String dbName) throws Exception {  1 usage  unknown
    try {
        Class.forName( className: "org.sqlite.JDBC");
        connection = DriverManager.getConnection( url: "jdbc:sqlite:" + dbName);
        System.out.println("Подключение к базе данных установлено: " + dbName);
    } catch (ClassNotFoundException e) {
        System.err.println("SQLite JDBC драйвер не найден!");
        throw e;
    } catch (SQLException e) {
        System.err.println("Ошибка подключения к базе данных: " + e.getMessage());
        throw e;
    }
}
```

```java
public void createTables() {  1 usage  unknown
    try (Statement stmt = connection.createStatement()) {
        System.out.println("Создание таблиц в базе данных...");

        stmt.execute( sql: "DROP TABLE IF EXISTS earthquakes");
        stmt.execute( sql: "DROP TABLE IF EXISTS regions");
```

```java
// Создаем таблицу регионов
stmt.execute( sql: "CREATE TABLE IF NOT EXISTS regions (" +
        "region_id INTEGER PRIMARY KEY AUTOINCREMENT," +
        "name TEXT UNIQUE NOT NULL," +
        "created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP)");
```

```java
// Создаем таблицу землетрясений
stmt.execute( sql: "CREATE TABLE IF NOT EXISTS earthquakes (" +
        "earthquake_id TEXT PRIMARY KEY," +
        "region_id INTEGER," +
        "magnitude REAL NOT NULL," +
        "depth REAL DEFAULT 0.0," +
        "magnitude_type TEXT," +
        "time TIMESTAMP," +
        "state TEXT," +
        "created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP," +
        "FOREIGN KEY (region_id) REFERENCES regions(region_id))");
```

```
// Создаем индексы для ускорения запросов
stmt.execute( sql: "CREATE INDEX IF NOT EXISTS idx_time ON earthquakes(time)");
stmt.execute( sql: "CREATE INDEX IF NOT EXISTS idx_magnitude ON earthquakes(magnitude)");
stmt.execute( sql: "CREATE INDEX IF NOT EXISTS idx_depth ON earthquakes(depth)");
stmt.execute( sql: "CREATE INDEX IF NOT EXISTS idx_region ON earthquakes(region_id)");
```

```
// Вставляем регионы
String regionSql = "INSERT OR IGNORE INTO regions (name) VALUES (?)";
try (PreparedStatement pstmt = connection.prepareStatement(regionSql)) {
    for (Earthquake eq : earthquakes) {
        if (eq.getState() != null && !eq.getState().isEmpty()) {
            pstmt.setString( parameterIndex: 1, eq.getState());
            pstmt.addBatch();
            regionCount++;
        }
    }
    if (regionCount > 0) {
        pstmt.executeBatch();
        System.out.println("Добавлено регионов: " + regionCount);
    }
}
```

```java
// Вставляем землетрясения
String eqSql = "INSERT OR REPLACE INTO earthquakes " +
        "(earthquake_id, region_id, magnitude, depth, magnitude_type, time, state) " +
        "VALUES (?, ?, ?, ?, ?, ?, ?)";

try (PreparedStatement pstmt = connection.prepareStatement(eqSql)) {
    for (Earthquake eq : earthquakes) {
        int regionId = getRegionId(eq.getState());

        pstmt.setString( parameterIndex: 1, eq.getId());
        pstmt.setInt( parameterIndex: 2, regionId);
        pstmt.setDouble( parameterIndex: 3, eq.getMagnitude());
        pstmt.setDouble( parameterIndex: 4, eq.getDepth());
        pstmt.setString( parameterIndex: 5, eq.getMagnitudeType());
```

```java
            if (eq.getTime() != null) {
                pstmt.setTimestamp( parameterIndex: 6, Timestamp.valueOf(eq.getTime()));
                timeCount++;
            } else {
                pstmt.setNull( parameterIndex: 6, Types.TIMESTAMP);
            }

            pstmt.setString( parameterIndex: 7, eq.getState());
            pstmt.addBatch();
            earthquakeCount++;
        }

        pstmt.executeBatch();
    }
```

SQLQueries.java

Содержит SQL-запросы для анализа данных из базы.

Класс SQLQueries - Ходит в базу данных с заранее подготовленными вопросами и приносит ответы.

Файл был написан пятым

```java
package database;

import java.sql.*;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
```

```java
// 1. Землетрясения с магнитудой больше 4.0
public void getStrongEarthquakes() {  1 usage   unknown
    System.out.println("\n=== Таблица 1: Сильные землетрясения (магнитуда > 4.0) ===");
    String sql = "SELECT earthquake_id, magnitude, depth, time " +
            "FROM earthquakes " +
            "WHERE magnitude > 4.0 " +
            "ORDER BY magnitude DESC " +
            "LIMIT 10";

    executeAndPrintQueryWithTime(sql, new String[]{"ID", "Магнитуда", "Глубина", "Время"});
}
```

```java
// 2. Самые глубокие землетрясения
public void getDeepestEarthquakes(int limit) {  1 usage   unknown
    System.out.println("\n=== Таблица 2: Самые глубокие землетрясения ===");
    String sql = String.format(
            "SELECT earthquake_id, depth, magnitude, time " +
                    "FROM earthquakes " +
                    "WHERE depth > 0 " +
                    "ORDER BY depth DESC " +
                    "LIMIT %d", limit);

    executeAndPrintQueryWithTime(sql, new String[]{"ID", "Глубина (м)", "Магнитуда", "Время"});
}
```

```java
// 3. Землетрясения по годам
public void getEarthquakesByYear() {  1 usage   unknown
    System.out.println("\n=== Таблица 3: Землетрясения по годам ===");
    try {
        // Проверяем наличие данных о времени
        String checkSql = "SELECT COUNT(*) as cnt FROM earthquakes WHERE time IS NOT NULL";
        ResultSet checkRs = dbManager.executeQuery(checkSql);

        int timeCount = 0;
        if (checkRs.next()) {
            timeCount = checkRs.getInt( columnLabel: "cnt");
        }

        if (timeCount == 0) {
            System.out.println("В данных нет информации о времени землетрясений");
            return;
        }
```

```
// Запрос для анализа по годам
String sql = "SELECT " +
        "strftime('%Y', datetime(time/1000, 'unixepoch')) as year, " +
        "COUNT(*) as count, " +
        "AVG(magnitude) as avg_magnitude, " +
        "MAX(magnitude) as max_magnitude, " +
        "MIN(magnitude) as min_magnitude " +
        "FROM earthquakes " +
        "WHERE time IS NOT NULL AND time != 0 " +
        "GROUP BY year " +
        "HAVING year IS NOT NULL AND year != '' " +
        "ORDER BY year DESC";

ResultSet rs = dbManager.executeQuery(sql);
```

```
// 4. Статистика по магнитудам
public void getAverageMagnitudeByType() {  1 usage   unknown
    System.out.println("\n=== Таблица 4: Общая статистика ===");
    try {
        String sql = "SELECT " +
                "COUNT(*) as total_count, " +
                "SUM(CASE WHEN time IS NOT NULL AND time != 0 THEN 1 ELSE 0 END) as with_time_count, " +
                "AVG(magnitude) as avg_magnitude, " +
                "MAX(magnitude) as max_magnitude, " +
                "MIN(magnitude) as min_magnitude, " +
                "AVG(depth) as avg_depth, " +
                "MAX(depth) as max_depth " +
                "FROM earthquakes";

        ResultSet rs = dbManager.executeQuery(sql);
```

```
// 5. Топ землетрясений
public void getTopEarthquakes() {  1 usage   unknown
    System.out.println("\n=== Таблица 5: Топ-10 землетрясений по магнитуде ===");
    try {
        String sql = "SELECT earthquake_id, magnitude, depth, time " +
                "FROM earthquakes " +
                "WHERE magnitude > 0 " +
                "ORDER BY magnitude DESC " +
                "LIMIT 10";

        ResultSet rs = dbManager.executeQuery(sql);

        System.out.printf("%-15s | %-10s | %-10s | %-25s\n",
                "ID", "Магнитуда", "Глубина", "Время");
        System.out.println("-".repeat( count: 65));
```

TextChartGenerator.java

Создает ASCII-графики для визуализации данных в консоли.

Класс TextChartGenerator - Берет сухие статистические данные и превращает их в наглядные текстовые диаграммы.

```
package visualization;

import java.util.Map;

public class TextChartGenerator {
```

```java
public static void printBarChart(String title, Map<String, Number> data,  3 usages  unknown
    System.out.println("\n" + title);
    System.out.println("=".repeat( count: 60));
    System.out.printf("%-30s | %-20s | %s%n", categoryLabel, valueLabel, "График");
    System.out.println("-".repeat( count: 60));

    double maxValue = data.values().stream() Stream<Number>
            .mapToDouble(Number::doubleValue) DoubleStream
            .max() OptionalDouble
            .orElse( other: 1.0);

    int maxBarLength = 50;

    for (Map.Entry<String, Number> entry : data.entrySet()) {
        String category = entry.getKey();
        double value = entry.getValue().doubleValue();
        int barLength = (int) ((value / maxValue) * maxBarLength);

            String bar = "".repeat(Math.max(0, barLength));
            System.out.printf("%-30s | %-20.2f | %s%n",
                    truncate(category,  maxLength: 30), value, bar);
        }
    }

    public static void printPieChart(String title, Map<String, Number> data) {
        System.out.println("\n" + title);
        System.out.println("=".repeat( count: 60));

        double total = data.values().stream() Stream<Number>
                .mapToDouble(Number::doubleValue) DoubleStream
                .sum();
```

```java
    if (total == 0) {
        System.out.println("Нет данных для отображения");
        return;
    }

    System.out.printf("%-30s | %-10s | %-10s | %s%n",
            "Категория", "Значение", "Процент", "Доля");
    System.out.println("-".repeat( count: 60));

    for (Map.Entry<String, Number> entry : data.entrySet()) {
        String category = entry.getKey();
        double value = entry.getValue().doubleValue();
        double percentage = (value / total) * 100;

        int dots = Math.max(1, (int) (percentage / 2));
        String share = "•".repeat(dots);
```

```java
        System.out.printf("%-30s | %-10.2f | %-10.1f%% | %s%n",
                truncate(category, maxLength: 30), value, percentage, share);
    }
}

public static void printStatisticsTable(Map<String, Object> data, String title) {  1 usage   unknown
    System.out.println("\n" + title);
    System.out.println("=".repeat( count: 60));
    System.out.printf("%-40s | %-15s%n", "Параметр", "Значение");
    System.out.println("-".repeat( count: 60));
```

Main.java

Координирует работу всех компонентов программы.

```java
import models.EarthquakeAnalyzer;
import parser.CSVReader;
import database.DatabaseManager;
import database.SQLQueries;
import visualization.TextChartGenerator;
import java.util.*;
import java.util.stream.Collectors;

public class Main {  unknown
```

```java
try {
    // 1. Отладка CSV файла
    CSVReader csvReader = new CSVReader();
    csvReader.debugCSV( filename: "Землетрясения.csv");

    // 2. Чтение данных из CSV файла
    System.out.println("\n2. Чтение данных из CSV файла...");
    List<models.Earthquake> earthquakes = csvReader.readCSV( filename: "Землетрясения.csv");

    if (earthquakes.isEmpty()) {
        System.out.println("ОШИБКА: Не удалось прочитать данные из CSV файла");
        return;
    }
```

```java
// Анализ данных о времени
int withTime = 0;
int withoutTime = 0;
for (models.Earthquake eq : earthquakes) {
    if (eq.getTime() != null) {
        withTime++;
    } else {
        withoutTime++;
    }
}
```

```java
System.out.println("\nСтатистика времени:");
System.out.println("  С временем: " + withTime + " (" +
        String.format("%.1f%%", earthquakes.size() > 0 ? (withTime * 100.0 / earthquakes.size()) : 0) + ")");
System.out.println("  Без времени: " + withoutTime + " (" +
        String.format("%.1f%%", earthquakes.size() > 0 ? (withoutTime * 100.0 / earthquakes.size()) : 0) + ")");
```

```java
// 3. Создание анализатора
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("3. Создание анализатора...");
EarthquakeAnalyzer analyzer = new EarthquakeAnalyzer();
for (models.Earthquake eq : earthquakes) {
    analyzer.addEarthquake(eq);
}
```

```java
// 4. Вывод общей статистики
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("4. Общая статистика:");

Map<String, Object> stats = analyzer.getStatistics();
TextChartGenerator.printStatisticsTable(stats, title: "Общая статистика данных");

// 5. Текстовая визуализация данных
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("5. Визуализация данных:");
```

```java
System.out.println("\n=== Распределение по штатам ===");
Map<String, Long> earthquakesByState = analyzer.getEarthquakeCountByState();
if (!earthquakesByState.isEmpty()) {
    Map<String, Number> stateData = new HashMap<>();
    for (Map.Entry<String, Long> entry : earthquakesByState.entrySet()) {
        stateData.put(entry.getKey(), entry.getValue());
    }
    TextChartGenerator.printBarChart(
            title: "Количество землетрясений по штатам (топ-15)",
            stateData,
            categoryLabel: "Штат",
            valueLabel: "Количество землетрясений");
```

```java
    System.out.println("\nТоп-5 штатов по количеству землетрясений:");
    int count = 0;
    for (Map.Entry<String, Long> entry : earthquakesByState.entrySet()) {
        if (count >= 5) break;
        System.out.printf("  %d. %s: %d землетрясений\n",
                count + 1, entry.getKey(), entry.getValue());
        count++;
    }
} else {
    System.out.println("Нет данных по штатам для визуализации");
}
```

```java
// 5.2 Распределение по магнитудам
System.out.println("\n=== Распределение по магнитудам ===");
Map<String, Long> magnitudeDistribution = analyzer.getMagnitudeDistribution();
if (!magnitudeDistribution.isEmpty()) {
    Map<String, Number> magnitudeData = new HashMap<>();
    for (Map.Entry<String, Long> entry : magnitudeDistribution.entrySet()) {
        magnitudeData.put(entry.getKey(), entry.getValue());
    }
    TextChartGenerator.printPieChart(
            title: "Распределение землетрясений по магнитудам",
            magnitudeData);
} else {
    System.out.println("Нет данных по магнитудам для визуализации");
}
```

```java
// 6. Работа с базой данных
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("6. Работа с базой данных:");

DatabaseManager dbManager = new DatabaseManager( dbName: "earthquakes.db");
dbManager.createTables();
dbManager.saveEarthquakes(earthquakes);
```

```java
// 7. Выполнение SQL запросов
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("7. Результаты SQL запросов:");

SQLQueries queries = new SQLQueries(dbManager);

// 7.1 Сильные землетрясения
queries.getStrongEarthquakes();

// 7.2 Самые глубокие землетрясения
queries.getDeepestEarthquakes( limit: 10);

// 7.3 Землетрясения по годам
queries.getEarthquakesByYear();

// 7.4 Общая статистика
queries.getAverageMagnitudeByType();
```

```java
// 8. Дополнительный анализ из анализатора
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("8. Дополнительный анализ:");

// 8.1 Топ-10 землетрясений по магнитуде (из анализатора)
System.out.println("\nТоп-10 землетрясений по магнитуде (анализ в памяти):");
List<models.Earthquake> topByMagnitude = analyzer.getTopByMagnitude( limit: 10);
if (!topByMagnitude.isEmpty()) {
    System.out.printf("%-15s | %-10s | %-10s | %-20s | %-25s\n",
            "ID", "Магнитуда", "Глубина", "Штат", "Время");
    System.out.println("-".repeat( count: 85));

    for (models.Earthquake eq : topByMagnitude) {
        String timeStr = (eq.getTime() != null) ?
                eq.getTime().format(java.time.format.DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")) :
                "Нет данных";
```

```java
// 9. Итоговая статистика
System.out.println("\n" + "=".repeat( count: 50));
System.out.println("9. Итоговая статистика:");

System.out.println("Всего обработано записей: " + earthquakes.size());
System.out.println("Успешно считано из CSV: " + earthquakes.size());
System.out.println("Записей с временем: " + withTime +
        String.format(" (%.1f%%)", earthquakes.size() > 0 ? (withTime * 100.0 / earthquakes.size()) : 0));
```

```java
        // Закрываем соединение с базой данных
        dbManager.close();

        System.out.println("\n" + "=".repeat( count: 50));
        System.out.println("Анализ завершен успешно!");
        System.out.println("=".repeat( count: 50));

    } catch (Exception e) {
        System.err.println("\nОШИБКА выполнения программы: " + e.getMessage());
        e.printStackTrace();
```

```java
        // Выводим только релевантные части стека вызовов
        System.err.println("\nСтек вызовов (отфильтрованный):");
        boolean foundRelevant = false;
        for (StackTraceElement element : e.getStackTrace()) {
            if (element.getClassName().contains("earthquake") ||
                    element.getClassName().contains("Earthquake") ||
                    element.getClassName().contains("parser") ||
                    element.getClassName().contains("database") ||
                    element.getClassName().contains("models")) {
                System.err.println("  " + element);
                foundRelevant = true;
            }
        }
```

Вывод в консоль

```
=== Результаты чтения ===
Всего строк в файле: 1648
Успешно прочитано: 1647
Ошибок чтения: 0
Процент успеха: 100,0%
```

```
Общая статистика данных
============================================================
Параметр                                  | Значение
------------------------------------------------------------
Всего землетрясений                       | 1647,00
Средняя магнитуда                         | 2,38
Максимальная магнитуда                    | 5,20
Минимальная магнитуда                     | 0,00
Средняя глубина (м)                       | 7561,55
Максимальная глубина (м)                  | 32300,00
С временем                                | 1647,00
Без времени                               | 0,00
Процент с временем                        | 100,0%
Уникальных штатов                         | 111,00
Самый частый штат                         | West Virginia
Период данных                             | 1973 - 2015
```

```
Топ-5 штатов по количеству землетрясений:
  1. West Virginia: 865 землетрясений
  2. New York: 308 землетрясений
  3. Pennsylvania: 227 землетрясений
  4. Virginia: 26 землетрясений
  5. Ohio: 23 землетрясений
```

Графики

```
=== Распределение по штатам ===

Количество землетрясений по штатам (топ-15)
============================================================
Штат                          | Количество землетрясений | График
------------------------------------------------------------
West Virginia                 | 865,00                   | ████████████████████████████████████████
New York                      | 308,00                   | ██████████████
Youngstown-akron urban ar...  | 6,00                     |
Greater New York Area         | 11,00                    |
Chesapeake bay region         | 9,00                     |
Greater philadelphia area     | 10,00                    |
Ohio                          | 23,00                    | █
Ontario-quebec border reg...  | 16,00                    |
Virginia                      | 26,00                    | █
Southern quebec               | 18,00                    | █
New Jersey                    | 22,00                    | █
Pennsylvania                  | 227,00                   | ██████████
```

```
Распределение землетрясений по магнитудам
============================================================
Категория                     | Значение   | Процент    | Доля
------------------------------------------------------------
5.0 - 5.9                     | 2,00       | 0,1       % | ·
3.0 - 3.9                     | 97,00      | 5,9       % | ··
< 2.0                         | 228,00     | 13,8      % | ••••••
2.0 - 2.9                     | 1310,00    | 79,5      % | ••••••••••••••••••••••••••••••••••••
4.0 - 4.9                     | 10,00      | 0,6       % | ·
```

```
Распределение землетрясений по глубине
============================================================
Глубина                       | Количество землетрясений | График
------------------------------------------------------------
Мелкие (< 5 км)               | 552,00                   | ████████████████████████████████████████
Средние (5-10 км)             | 554,00                   | ████████████████████████████████████████
Глубокие (10-20 км)           | 517,00                   | ██████████████████████████████████████
Очень глубокие (20-50 км)     | 24,00                    | ██
```

Распределение землетрясений по годам

Год	Количество землетрясений	График
2012	116,00	▒▒▒▒▒▒▒▒▒▒
2011	24,00	▒▒
2010	20,00	▒
2009	28,00	▒▒
1998	4,00	
1976	3,00	
2008	15,00	▒
1997	6,00	
1975	5,00	
2007	13,00	▒
1996	3,00	
1974	5,00	
2006	8,00	
1995	6,00	
1973	2,00	
2005	5,00	
1994	7,00	
2004	4,00	
1993	9,00	
2003	7,00	
1992	2,00	
2002	9,00	
1991	8,00	
1991	8,00	
1979	5,00	
1978	9,00	
1999	2,00	
1977	1,00	
2001	7,00	
1990	2,00	
2000	6,00	
1987	7,00	
1986	9,00	
1985	8,00	
1984	6,00	
1983	9,00	
2015	137,00	▒▒▒▒▒▒▒▒▒
1982	3,00	
2014	536,00	▒▒
1981	5,00	
2013	580,00	▒▒
1980	5,00	
1988	1,00	

Таблицым

```
=== Таблица 1: Сильные землетрясения (магнитуда > 4.0) ===
ID                  Магнитуда           Глубина             Время
--------------------------------------------------------------------------------
usp000b2u2          5,20                11000 м             2002-04-20 10:50:47
usp0001yuv          5,10                12500 м             1983-10-07 10:18:46
usp0000gyg          4,70                5000 м              1976-06-19 05:54:13
usp00066h6          4,60                5000 м              1994-01-16 01:49:16
usp0008vh3          4,50                5000 м              1998-09-25 19:52:52
usp000ewg9          4,30                1000 м              2006-11-02 17:53:02
usp00066h5          4,20                5000 м              1994-01-16 00:42:43
usp00023tg          4,20                5000 м              1984-04-23 01:36:00
usp0000bd0          4,20                10000 м             1975-06-09 18:39:23
```

```
=== Таблица 2: Самые глубокие землетрясения ===
ID                  Глубина (м)         Магнитуда           Время
--------------------------------------------------------------------------------
4221124             32300 м             2,80                2012-12-28 16:24:42
4213045             31500 м             2,20                2012-07-28 16:45:51
usc000rccm          29600 м             2,60                2014-06-06 22:15:40
4488459             26200 м             2,70                2013-10-09 16:18:55
4372160             26000 м             2,20                2013-08-23 18:17:33
4219251             25700 м             2,30                2012-10-30 22:06:22
usp0002x3n          24000 м             3,40                1986-08-13 04:55:18
usp000hn29          23900 м             2,40                2010-10-13 17:30:43
usp0002x3v          23000 м             2,30                1986-08-13 10:29:24
usp000j71v          22900 м             2,20                2011-08-26 04:55:20
```

```
=== Таблица 3: Землетрясения по годам ===
Найдено записей со временем: 1647
```

Год	Количество	Средняя маг.	Макс. маг.	Мин. маг.
2015	137	2,15	3,10	1,10
2014	536	2,18	3,10	1,02
2013	580	2,39	3,40	1,20
2012	116	2,65	3,50	1,90
2011	24	2,55	4,00	0,40
2010	20	2,57	3,40	2,10
2009	28	2,49	3,30	1,10
2008	15	2,07	3,40	0,00
2007	13	2,63	3,20	2,10
2006	8	2,36	4,30	0,90
2005	5	2,52	3,50	1,90
2004	4	2,65	2,90	2,20
2003	7	2,80	3,80	2,20
2002	9	2,94	5,20	1,10
2001	7	2,76	3,90	1,90
2000	6	2,82	3,80	2,00
1999	2	2,70	2,90	2,50
1998	4	3,18	4,50	2,40
1997	6	2,87	3,20	2,20
1996	3	2,53	2,80	2,20
1995	6	2,87	3,10	2,50
1994	7	3,16	4,60	2,40
1993	9	2,58	3,90	1,60
1992	2	2,80	3,10	2,50
1991	8	3,10	4,00	2,20

1991	8	3,10	4,00	2,20
1990	2	2,55	2,60	2,50
1988	1	3,50	3,50	3,50
1987	7	3,09	3,80	2,70
1986	9	2,52	3,40	1,80
1985	8	2,76	3,60	2,00
1984	6	3,23	4,20	2,40
1983	9	3,12	5,10	2,00
1982	3	2,70	2,80	2,60
1981	5	2,96	3,50	2,10
1980	5	3,28	3,70	2,80
1979	5	3,00	3,60	2,50
1978	9	2,62	3,10	1,50
1977	1	3,10	3,10	3,10
1976	3	3,30	4,70	2,40
1975	5	3,18	4,20	2,20
1974	5	3,16	3,60	2,50
1973	2	3,60	3,80	3,40
ИТОГО	1647	2,38		

=== Таблица 5: Топ-10 землетрясений по магнитуде ===

ID	Магнитуда	Глубина	Время
usp000b2u2	5,20	11000	2002-04-20 10:50:47
usp0001yuv	5,10	12500	1983-10-07 10:18:46
usp0000gyg	4,70	5000	1976-06-19 05:54:13
usp00066h6	4,60	5000	1994-01-16 01:49:16
usp0008vh3	4,50	5000	1998-09-25 19:52:52
usp000ewg9	4,30	1000	2006-11-02 17:53:02
usp00066h5	4,20	5000	1994-01-16 00:42:43
usp00023tg	4,20	5000	1984-04-23 01:36:00
usp0000bd0	4,20	10000	1975-06-09 18:39:23
usp0000de4	4,00	3000	1975-11-03 20:54:55

```
=== Анализ по последнему году (2015) ===

=== Таблица 6: Землетрясения по месяцам за 2015 год ===
Месяц      | Количество      | Ср. магнитуда
-------------------------------------------
Январь     | 42              | 2,37
Февраль    | 10              | 2,18
Март       | 32              | 2,25
Апрель     | 2               | 1,94
Май        | 45              | 1,97
Июнь       | 1               | 1,33
Июль       | 3               | 1,55
Август     | 2               | 1,46
```

```
Топ-10 землетрясений по магнитуде (анализ в памяти):
ID             | Магнитуда  | Глубина    | Штат                | Время
-------------------------------------------------------------------------------
usp000b2u2     | 5,20       | 11000      | New York            | 2002-04-20 10:50:47
usp0001yuv     | 5,10       | 12500      | New York            | 1983-10-07 10:18:46
usp0000gyg     | 4,70       | 5000       | West Virginia       | 1976-06-19 05:54:13
usp00066h6     | 4,60       | 5000       | Pennsylvania        | 1994-01-16 01:49:16
usp0008vh3     | 4,50       | 5000       | Pennsylvania        | 1998-09-25 19:52:52
usp000ewg9     | 4,30       | 1000       | West Virginia       | 2006-11-02 17:53:02
usp0000bd0     | 4,20       | 10000      | New York            | 1975-06-09 18:39:23
usp00023tg     | 4,20       | 5000       | Pennsylvania        | 1984-04-23 01:36:00
usp00066h5     | 4,20       | 5000       | Pennsylvania        | 1994-01-16 00:42:43
3372376        | 4,00       | 7000       | WEST VIRGINIA       | 2011-07-22 23:02:39
```

```
Топ-5 самых глубоких землетрясений:
ID             | Глубина (м) | Магнитуда  | Штат
-----------------------------------------------------------
4221124        | 32300       | 2,80       | WEST VIRGINIA
4213045        | 31500       | 2,20       | WEST VIRGINIA
usc000rccm     | 29600       | 2,60       | 13km NNE of Sisso...
4488459        | 26200       | 2,70       | WEST VIRGINIA
4372160        | 26000       | 2,20       | PENNSYLVANIA
```

Статистика по всем штатам:

Штат	Количество
West Virginia	865
New York	308
Pennsylvania	227
Virginia	26
Ohio	23
New Jersey	22
Southern quebec	18
Ontario-quebec border region	16
Greater New York Area	11
Greater philadelphia area	10
Chesapeake bay region	9
Youngstown-akron urban area	6
Vermont	3
Eastern kentucky	2
Lake erie	2