

# Вопросы к экзамену по курсу «Программирование на Java»

В ходе практической части экзамена будет доступна документация по Java (с помощью zeal), по Spring Boot (с помощью zeal), JUnit (с помощью пользовательского контента для zeal: <https://github.com/Kapeli/Dash-User-Contributions/tree/master/docsets/JUnit5>, JavaFX (самостоятельно будет импортировано преподавателем)).

Кроме того, практическое задание можно выполнять с собственном ручным конспектом.

Для Spring проектов будет предоставлен результат запуска <https://start.spring.io/>

По практической части задания проводится опрос; по теоретической – наряду с вопросами билета задаются дополнительные (краткие) вопросы.

**В билете должно быть два теоретических и один практический вопрос**

## Теоретические вопросы

1. Обзор языков высокого уровня: особенности статической и динамической типизации, плюсы и минусы, основные парадигмы программирования и их поддержка в Java
2. Java: архитектура выполнения - JVM, JDK, компиляция и интерпретация
3. Java: особенности логических операторов `&&` и `||`, тернарного оператора, префиксных и постфиксных инкрементов/декрементов
4. Java: примитивные и ссылочные типы данных, автоупаковка и распаковка
5. Java: работа с классом `Scanner` для ввода данных

6. Java: сравнение массивов и коллекций List, особенности использования
7. Java: отличия List и Set, принципы работы с множествами
8. Java: String vs StringBuilder - производительность и применение
9. Java: Map интерфейс, основные реализации и сценарии использования
10. Объектно-ориентированное программирование: принципы и их описание
11. Java: реализация инкапсуляции, наследования и полиморфизма в Java
12. Java: абстрактные классы и интерфейсы, отличия и применение
13. Java: классы и объекты в Java, синтаксис определения класса
14. Java: модификатор abstract, назначение и примеры использования
15. Java: перечисления (enum), синтаксис и практическое применение
16. Java: запечатанные классы (sealed classes), синтаксис и ограничения наследования
17. Java: статические методы и поля, особенности использования static
18. Java: вложенные классы, отличия статических и нестатических внутренних классов
19. Java: record классы, назначение и преимущества для DTO
20. Java: основные методы класса Object, контракты
21. Java: принципы создания unit-тестов, требования к качеству тестов, средства Java для создания автотестов
22. Java: основные методы класса String
23. Структуры данных: односвязные списки, реализация методов добавления и удаления в Java
24. Объектно-ориентированное программирование: итераторы (Iterator), назначение и принцип работы, самописная реализация
25. Java: лямбда-выражения в Java, синтаксис и применение

1. Java: многопоточность в Java, понятие потока и процесса
2. Java: создание потоков - Thread, Runnable, ExecutorService
3. Java: синхронизация потоков, ключевое слово synchronized
4. Java: взаимная блокировка (deadlock), причины и методы предотвращения
5. Java: гонки данных (race conditions), потокобезопасность
6. Java: атомарные переменные (AtomicInteger, AtomicLong), назначение
7. Java: пул потоков, создание фиксированного и однопоточного пула
8. Java: функциональные интерфейсы Predicate и Function
9. Java: потокобезопасные коллекции
10. JavaFX: архитектура - Stage, Scene, контейнеры
11. JavaFX: компоновка - HBox, VBox, GridPane, AnchorPane, Pane, Canvas, TabPane
12. JavaFX: элементы управления - TextField, Label, Button, CheckBox
13. JavaFX: выбор значений - ComboBox, RadioButton, DatePicker, ColorPicker
14. JavaFX: работа с файлами
15. JavaFX: использование TableView
16. JDBC: подключение SQLite
17. JDBC: выполнение SQL запросов в Java, PreparedStatement, ResultSet
18. Spring Boot: архитектура, аннотации @SpringBootApplication, @RestController
19. Spring Boot: конфигурация, application.properties, встроенный Tomcat
20. Rest API: описание, HTTP методы и статусы
21. Spring Boot: JPA, аннотации @Entity, @Id, CRUD репозитории
22. Spring Data: обработка ошибок, @ControllerAdvice, @ExceptionHandler
23. Spring Data: работа с базой данных H2, конфигурация
24. Rest API: тестирование REST API
25. Spring Boot: валидация данных, аннотации @Valid, @NotNull

# Практические задания

## Задание 1

Напишите консольную программу, которая решает неравенство  $(x-a)(x-b) > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте полную проверку ввода данных и обработку всех математических случаев. Создать Unit-тест для класса решения неравенства.

## Задание 2

Напишите консольную программу, которая по введённой дате (день, месяц, год) определяет дату следующего дня. Запрещено использовать стандартные классы для работы с датами. Обеспечьте проверку корректности ввода. Создать Unit-тест для класса, решающего задачу.

## Задание 3

Напишите JavaFX программу, которая решает неравенство  $(x - a)(x - b) > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте полную проверку ввода данных и обработку всех математических случаев.

## Задание 4

Напишите JavaFX программу, которая по введённой дате (день, месяц, год) определяет дату следующего дня. Запрещено использовать стандартные классы для работы с датами. Обеспечьте проверку корректности ввода.

## Задание 5

Напишите программу, которая читает целое число  $n$  ( $1 \leq n \leq 100$ ), затем  $n$  строк. Используя StringBuilder, обработайте каждую строку: инвертируйте её и добавьте в результат только если длина строки больше 5. Между добавленными строками вставьте разделитель — ".

## Задание 6

Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 100$ ), затем  $m$  пар: строка-ключ и целое значение. Используя HashMap<String,

`Integer>`, сохраните данные (при дубликатах ключей суммируйте значения). Затем для каждого запроса выведите сумму значений, чьи ключи начинаются с запроса.

### **Задание 7**

Напишите программу, которая читает  $p$  целых чисел ( $1 \leq p \leq 200$ ). Используя `HashSet<Integer>`, сохраните уникальные значения. Затем для каждого запроса удалите число из множества (если оно есть) и добавьте его квадрат. В конце выведите элементы в порядке возрастания.

### **Задание 8**

Напишите программу, которая читает  $r$  целых чисел ( $1 \leq r \leq 100$ ). Используя `ArrayList<Integer>`, сохраните их. Затем обработайте операции: "add X Y добавить X в позицию  $Y \bmod \text{size}$  (только если  $X > 0$ ). После всех операций выведите список в прямом порядке.

### **Задание 9**

Реализуйте иерархию классов для работников: базовый класс Сотрудник с полями ФИО и должность, производные классы Преподаватель (количество часов в год) и Лаборант (количество ставок). Реализуйте метод расчета годовой оплаты. Выведите список работников в порядке возрастания оплаты. Дополнительные необходимые параметры храните в отдельном класс Конфигурация.

### **Задание 10**

Создайте класс «Точка в 3D» с полями  $x$ ,  $y$ ,  $z$  (тип `double`). Реализуйте методы `clone`, `equals`, `hashCode` и `toString`. При некорректных данных для инициализации вызывайте исключение.

### **Задание 11**

Реализуйте класс `MyLinkedList`, содержащий внутренний класс `Node` и вложенный итератор. Метод `add(int value)` добавляет элемент в конец списка. Метод `remove(int value)` удаляет первое вхождение значения. Метод `print()` выводит все элементы. Итератор последовательно возвращает все элементы.

## **Задание 12**

Реализуйте класс LinearEquationSolver с методом static LinearResult solve(double a, double b), решающим уравнение  $ax + b = 0$ . LinearResult должен быть sealed-классом с подклассами NoSolution, InfiniteSolutions и UniqueSolution. Напишите unit-тесты для всех случаев.

## **Задание 13**

Реализуйте потокобезопасный класс Counter с методами increment(int value), decrement(int value) (только если результат не станет отрицательным) и getValue(). В main создайте экземпляр с начальным значением 500, запустите 10 потоков: 5 потоков по 100 раз вызывают increment(5), 5 потоков по 100 раз вызывают decrement(5). После завершения выведите итоговое значение.

## **Задание 14**

Реализуйте программу с графическим интерфейсом JavaFX для решения квадратного уравнения  $ax^2 + bx + c = 0$ . Обеспечьте ввод коэффициентов, проверку корректности данных и вывод результатов в удобном виде.

## **Задание 15**

Реализуйте программу с графическим интерфейсом JavaFX, которая динамически создаёт таблицу (максимум  $10 \times 10$ ) по заданным размерам. После заполнения таблицы числами программа рассчитывает сумму чисел в каждой строке, сумму чисел в каждом столбце и общую сумму всех чисел.

## **Задание 16**

Реализуйте программу с графическим интерфейсом JavaFX для имитации движения мяча на бильярдном столе. Учтите упругие отскоки от границ стола. Пользователь задаёт начальную скорость и направление движения, лунки не учитывайте.

## **Задание 17**

Создайте приложение на JavaFX для работы с базой данных заметок. Поля: заголовок, содержание. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, изменения записей.

## **Задание 18**

Создайте приложение на JavaFX для работы с базой данных заметок. Поля: заголовок, содержание. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, удаления записей.

## **Задание 19**

Разработайте REST API сервис на Spring Boot для управления заметками (поля – заголовок и содержание). Предоставьте эндпоинты для CRUD-операций. Используйте реляционную базу данных для хранения данных.

## **Задание 20**

Разработайте REST API сервис на Spring Boot для управления заметками (поля – заголовок и содержание). Реализуйте только создание и получение списка заметок. Создайте клиент на JavaFX для этого сервиса

## **Задание 21**

Реализуйте функционал регистрации и аутентификации пользователей в REST API сервисе с использованием JWT. Реализуйте middleware для проверки JWT-токенов. Реализуйте JavaFX-клиент, который осуществляет регистрацию и аутентификацию пользователей с помощью сервиса.

## **Задание 22**

Напишите консольное приложение для решения неравенства  $\frac{x-a}{x-b} > 0$ , где  $a$  и  $b$  – вещественные числа, вводимые пользователем. Обеспечьте проверку деления на ноль и вывод правильного решения. Реализуйте UNIT-тестирование модели.

## **Задание 23**

Напишите JavaFX приложение для решения неравенства  $\frac{x-a}{x-b} > 0$ , где  $a$  и  $b$  – вещественные числа, вводимые пользователем. Обеспечьте проверку деления на ноль и вывод правильного решения. Реализуйте UNIT-тестирование модели.

## **Задание 24**

Напишите консольную программу, которая по введённой дате определяет дату предыдущего дня. Запрещено использовать стандартные классы для работы с датами. Учтите високосные годы и корректное количество дней в месяцах. Проверяйте корректность входных данных. Реализуйте UNIT-тестирование модели.

## **Задание 25**

Напишите JavaFX программу, которая по введённой дате определяет дату предыдущего дня. Запрещено использовать стандартные классы для работы с датами. Учтите високосные годы и корректное количество дней в месяцах. Проверяйте корректность входных данных. Реализуйте UNIT-тестирование модели.

## **Задание 26**

Напишите JavaFX программу, которая находит наименьшее  $k$ , такое что  $k! \geq n$  для заданного натурального  $n$ . Используйте цикл while. Реализуйте UNIT-тестирование модели.

## **Задание 27**

Напишите программу, которая читает целое число  $n$  ( $1 \leq n \leq 100$ ), затем  $n$  строк. Используя StringBuilder, обработайте каждую строку: удалите все гласные и добавьте в результат только если строка начинается с согласной. Между добавленными строками вставьте разделитель — ".

## **Задание 28**

Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 100$ ), затем  $m$  пар: строка-ключ и целое значение. Используя  $\text{HashMap}<\text{String}, \text{Integer}>$ , сохраните данные. Затем для каждого запроса выведите максимальное значение среди тех, чьи ключи заканчиваются на запрос.

## **Задание 29**

Напишите программу, которая читает  $p$  целых чисел ( $1 \leq p \leq 200$ ). Используя  $\text{HashSet}<\text{Integer}>$ , сохраните уникальные значения. Затем для каждого запроса удалите число из множества (если оно есть) и добавьте

его удвоенное значение. В конце выведите элементы в порядке возрастания.

### **Задание 30**

Напишите программу, которая читает  $r$  целых чисел ( $1 \leq r \leq 100$ ). Используя `ArrayList<Integer>`, сохраните их. Затем обработайте операции: "remove Z удалить число Z из списка, только если оно чётное. После всех операций выведите оставшиеся элементы.

### **Задание 31**

Реализуйте иерархию классов для геометрических фигур: базовый абстрактный класс Фигура с абстрактным методом `площадь()`, производные классы Круг (радиус), Прямоугольник (длина, ширина), Треугольник (стороны). Реализуйте корректный ввод данных о любом количестве фигур (по выбору пользователя) с клавиатуры (консоль). Выведите на экран информацию о всех фигурах и сумму площадей всех фигур.

### **Задание 32**

Создайте класс «Время суток» с полями `hour` (0–23), `minute` (0–59), `second` (0–59) (все — int). Реализуйте методы `clone`, `equals`, `hashCode` и `toString`. При некорректных данных для инициализации вызывайте исключение.

### **Задание 33**

Реализуйте класс `MyLinkedList` с методом `add(int value)`, добавляющим элемент в начало списка. Метод `remove(int index)` удаляет элемент по индексу. Метод `print()` выводит все элементы. Итератор возвращает элементы в порядке хранения.

### **Задание 34**

Реализуйте класс `SystemSolver` с методом `static SystemResult solve(double a1, double b1, double c1, double a2, double b2, double c2)` для решения системы двух линейных уравнений. `SystemResult` должен быть sealed-классом с подклассами `NoSolution`, `InfiniteSolutions` и `UniqueSolution`. Напишите unit-тесты.

## **Задание 35**

Реализуйте потокобезопасный класс SafeWallet с методами addMoney(int amount), spendMoney(int amount) (возвращает true, если хватает средств) и getBalance(). В main инициализируйте кошелёк с 2000 единицами, запустите 10 потоков: 5 потоков по 80 раз добавляют по 15, 5 потоков по 80 раз тратят по 15. После завершения выведите баланс.

## **Задание 36**

Реализуйте программу с графическим интерфейсом JavaFX для решения неравенства  $ax + b > 0$ . Обеспечьте ввод коэффициентов, проверку корректности данных и вывод результатов в удобном виде. Реализуйте UNIT-тестирование модели.

## **Задание 37**

Реализуйте программу с графическим интерфейсом JavaFX, которая позволяет пользователю указать размерность матрицы (максимум  $5 \times 5$ ), ввести элементы матрицы, а затем найти минимальный элемент в каждой строке и максимальное значение среди этих минимальных элементов.

## **Задание 38**

Создайте приложение на JavaFX для работы с базой данных доходов. Поля: дата, количество рублей. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, изменения записей.

## **Задание 39**

Создайте приложение на JavaFX для работы с базой данных доходов. Поля: дата, количество рублей. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, удаления записей.

## **Задание 40**

Разработайте REST API сервис на Spring Boot для управления информацией о доходах (поля – дата и количество рублей). Предоставьте эндпоинты для CRUD-операций. Используйте реляционную базу данных для хранения данных. Реализуйте валидацию входных данных.

## **Задание 41**

Разработайте REST API сервис на Spring Boot для управления информацией о доходах (поля – дата и количество рублей). Предоставьте энд-поинты для добавления дохода и получения всех доходов. Используйте реляционную базу данных для хранения данных. Реализуйте валидацию входных данных. Создайте JavaFX клиент для этого сервиса.

## **Задание 42**

Разработайте консольную программу решения неравенства  $|x - a| > b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте проверку входных данных и вывод правильного решения. Разработайте UNIT-тесты для модели.

## **Задание 43**

Разработайте JavaFX программу решения неравенства  $|x - a| > b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте проверку входных данных и вывод правильного решения. Разработайте UNIT-тесты для модели.

## **Задание 44**

Напишите консольную программу, которая по введённой дате определяет дату, которая наступит через месяц. Если в следующем месяце нет дня с таким же числом, возьмите последний день следующего месяца. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность входных данных. Разработайте UNIT-тесты для модели.

## **Задание 45**

Напишите JavaFX программу, которая по введённой дате определяет дату, которая наступит через месяц. Если в следующем месяце нет дня с таким же числом, возьмите последний день следующего месяца. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность входных данных. Разработайте UNIT-тесты для модели.

## **Задание 46**

Напишите программу, которая читает целое число  $n$  ( $1 \leq n \leq 100$ ), затем  $n$  строк. Используя `StringBuilder`, обработайте каждую строку: удвойте каждый символ и добавьте в результат только если строка содержит цифру. Между добавленными строками вставьте разделитель — ".

## **Задание 47**

Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 100$ ), затем  $m$  пар: строка-ключ и целое значение. Используя `HashMap<String, Integer>`, сохраните данные. Затем для каждого запроса выведите количество ключей, которые содержат запроc как подстроку.

## **Задание 48**

Напишите программу, которая читает  $p$  целых чисел ( $1 \leq p \leq 200$ ). Используя `HashSet<Integer>`, сохраните уникальные значения. Затем для каждого запроса удалите число из множества (если оно есть) и добавьте число +1. В конце выведите элементы в порядке возрастания.

## **Задание 49**

Напишите программу, которая читает  $r$  целых чисел ( $1 \leq r \leq 100$ ). Используя `ArrayList<Integer>`, сохраните их. Затем обработайте операции: "swap A B"— поменять местами элементы с индексами A и B, только если сумма значений этих элементов чётная. После всех операций выведите список.

## **Задание 50**

Реализуйте иерархию классов для транспортных средств: базовый класс Транспортное средство с полями марка, скорость, производные классы Автомобиль (количество дверей, тип кузова) и Мотоцикл (объем двигателя, наличие коляски). Реализуйте метод расчета времени в пути для заданного расстояния. Выведите информацию обо всех транспортных средствах.

## **Задание 51**

Создайте класс «Цвет в RGB» с полями red, green, blue (тип `int`, значения от 0 до 255). Реализуйте методы `clone`, `equals`, `hashCode` и `toString`. При

некорректных данных для инициализации вызывайте исключение.

### **Задание 52**

Реализуйте класс MyLinkedList, в котором add(int value) вставляет элемент с сохранением неубывающего порядка. Метод remove(int value) удаляет все вхождения значения. Метод print() выводит все элементы. Итератор возвращает элементы по порядку.

### **Задание 53**

Реализуйте класс LineIntersectionAnalyzer с методом static IntersectionResult intersect(double k1, double b1, double k2, double b2) для анализа пересечения прямых. IntersectionResult должен быть sealed-классом с подклассами ParallelDistinct, Coincident и IntersectAtPoint. Напишите unit-тесты.

### **Задание 54**

Реализуйте потокобезопасный класс ScoreBoard с методами addPoints(int points), removePoints(int points) (возвращает false, если недостаточно очков) и getScore(). В main создайте объект с начальным счётом 0, запустите 10 потоков: 5 потоков по 120 раз добавляют по 8 очков, 5 потоков по 120 раз убирают по 8. После завершения выведите итоговый счёт.

### **Задание 55**

Реализуйте программу с графическим интерфейсом JavaFX для перевода чисел из 10-ой системы счисления в 16-ую, 8-ую и 2-ую. Обеспечьте ввод числа, проверку корректности данных и вывод результатов в удобном виде.

### **Задание 56**

Реализуйте программу с графическим интерфейсом JavaFX для создания графического редактора. Пользователь может выбирать фигуры (прямоугольник, круг), вводить координаты и размещать их на холсте. Удаление фигур — клик по фигуре.

## **Задание 57**

Реализуйте программу с графическим интерфейсом JavaFX для имитации движения луча света. Пользователь задаёт координаты источника света, направление луча. Границы рисунка — идеальные зеркала. Программа отображает траекторию луча с сохранением следа.

## **Задание 58**

Создайте приложение на JavaFX для работы с базой данных самолетов авиакомпании. Поля: марка самолета, количество мест. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, изменения записей.

## **Задание 59**

Создайте приложение на JavaFX для работы с базой данных самолетов авиакомпании. Поля: марка самолета, количество мест. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, удаления записей.

## **Задание 60**

Создайте приложение на JavaFX для работы с базой данных самолетов авиакомпании. Поля: марка самолета, количество мест. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, фильтрации записей по первому полю.

## **Задание 61**

Разработайте REST API сервис на Spring Boot для хранения информации о самолетах авиакомпании (марка самолета и количество мест). Предоставьте эндпоинты для CRUD-операций. Используйте реляционную базу данных для хранения данных. Реализуйте валидацию входных данных.

## **Задание 62**

Разработайте REST API сервис на Spring Boot для хранения информации о самолетах авиакомпании (марка самолета и количество мест).

Предоставьте эндпоинты для создания и получения всех записей. Используйте реляционную базу данных для хранения данных. Реализуйте клиент сервиса на JavaFX.

### **Задание 63**

Напишите консольную программу, которая решает неравенство  $(x-a)^2 \geq 0$ , где  $a$  — вещественное число, вводимое пользователем. Обеспечьте полную проверку ввода данных и корректную обработку всех математических случаев. Разработайте UNIT-тесты.

### **Задание 64**

Напишите JavaFX программу, которая решает неравенство  $(x - a)^2 \geq 0$ , где  $a$  — вещественное число, вводимое пользователем. Обеспечьте полную проверку ввода данных и корректную обработку всех математических случаев. Разработайте UNIT-тесты.

### **Задание 65**

Напишите консольную программу, которая по введённой дате определяет дату, которая была месяц назад. Если в предыдущем месяце нет дня с таким же числом, возьмите последний день предыдущего месяца. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность исходных данных. Разработайте UNIT-тесты

### **Задание 66**

Напишите JavaFX программу, которая по введённой дате определяет дату, которая была месяц назад. Если в предыдущем месяце нет дня с таким же числом, возьмите последний день предыдущего месяца. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность исходных данных. Разработайте UNIT-тесты

### **Задание 67**

Напишите JavaFX программу, которая проверяет, является ли натуральное число  $n$  факториалом какого-либо натурального числа. Если да — выведите это число, иначе — сообщение «Не является факториалом». Используйте цикл while. Покройте ее UNIT-тестами (в части модели)

## **Задание 68**

Напишите программу, которая читает целое число  $n$  ( $1 \leq n \leq 100$ ), затем  $n$  строк. Используя StringBuilder, обработайте каждую строку: переведите в верхний регистр и добавьте в результат только если строка заканчивается на 'а'. Между добавленными строками вставьте разделитель — ".

## **Задание 69**

Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 100$ ), затем  $m$  пар: строка-ключ и целое значение. Используя HashMap<String, Integer>, сохраните данные. Затем для каждого запроса выведите минимальное значение среди тех, чьи ключи точно равны запросу.

## **Задание 70**

Напишите программу, которая читает  $p$  целых чисел ( $1 \leq p \leq 200$ ). Используя HashSet<Integer>, сохраните уникальные значения. Затем для каждого запроса удалите число из множества (если оно есть) и добавьте его факториал (для чисел до 10). В конце выведите элементы в порядке возрастания.

## **Задание 71**

Напишите программу, которая читает  $r$  целых чисел ( $1 \leq r \leq 100$ ). Используя ArrayList<Integer>, сохраните их. Затем обработайте операции: "add X to begin" добавить число X в начало списка, только если X нечётное. После всех операций выведите список в обратном порядке.

## **Задание 72**

Реализуйте иерархию классов для документов: базовый класс Документ с полями номер, дата, производные классы Счёт (сумма, покупатель) и Приказ (текст приказа, исполнитель). Реализуйте метод вывода информации о документе. Выведите список документов в хронологическом порядке (с полной информацией о них).

## **Задание 73**

Создайте класс «Геокоординаты с высотой» с полями latitude, longitude (double), altitude (int, в метрах). Реализуйте методы clone, equals, hashCode

и `toString`. При некорректных данных для инициализации вызывайте исключение. `latitude` может быть от -90 до +90, `longitude` от -180 до 180.

### Задание 74

Реализуйте класс `MyLinkedList` с методом `add(int target, int value)`, добавляющим `value` после первого вхождения `target` (если `target` не найден — в конец). Метод `remove(int value)` удаляет первое вхождение. Метод `print()` выводит все элементы. Итератор возвращает все элементы.

### Задание 75

Реализуйте класс `HeronTriangle` с методом `static AreaResult computeArea(double a, double b, double c)` для вычисления площади треугольника по формуле Герона. `AreaResult` должен быть `sealed`-классом с подклассами `InvalidTriangle`, `Degenerate` и `ValidTriangle`. Напишите unit-тесты.

### Задание 76

Реализуйте потокобезопасный класс `SafeAccount` с методами `topUp(int sum)`, `pay(int sum)` (возвращает `true`, если оплата возможна) и `getAmount()`. В `main` создайте счёт с 1500, запустите 10 потоков: 5 потоков по 90 раз пополняют на 20, 5 потоков по 90 раз оплачивают по 20. После завершения выведите баланс.

### Задание 77

Реализуйте программу с графическим интерфейсом JavaFX, которая читает файл с описанием расположения `TextField`'ов (координаты). После выбора файла на форме динамически создаются `TextField`'ы согласно данным из файла. При изменении любого числа в `TextField`'ах, в `Label` автоматически обновляется сумма всех введённых чисел.

### Задание 78

Реализуйте программу с графическим интерфейсом JavaFX для имитации движения двух планет. Пользователь задаёт начальные координаты и векторы скоростей двух шаров. На каждый шар действует сила притяжения к другому, равная  $\frac{k}{r^2}$ . Движение прекращается при столкновении шаров.

## **Задание 79**

Создайте приложение на JavaFX для работы с базой данных автобусных маршрутов. Поля: номер маршрута, номер парка. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, изменения записей.

## **Задание 80**

Создайте приложение на JavaFX для работы с базой данных автобусных маршрутов. Поля: номер маршрута, номер парка. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, удаления записей.

## **Задание 81**

Создайте приложение на JavaFX для работы с базой данных автобусных маршрутов. Поля: номер маршрута, номер парка. Используйте SQLite для хранения данных. Реализуйте возможность добавления, чтения, фильтрации записей по второму полю.

## **Задание 82**

Разработайте REST API сервис на Spring Boot для управления автобусными маршрутами (номер маршрута и номер парка). Предоставьте эндпоинты для CRUD-операций. Используйте реляционную базу данных для хранения данных. Реализуйте валидацию входных данных.

## **Задание 83**

Разработайте REST API сервис на Spring Boot для управления автобусными маршрутами (номер маршрута и номер парка). Предоставьте эндпоинты для добавления и получения всех записей. Используйте реляционную базу данных для хранения данных. Разработайте клиент сервиса на JavaFX.

## **Задание 84**

Напишите консольную программу, которая решает неравенство  $(a-x)(b-x) < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте полную проверку ввода данных и корректную обработку всех математических случаев. Разработайте UNIT-тесты для модели.

## **Задание 85**

Напишите JavaFX программу, которая решает неравенство  $(a - x)(b - x) < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем. Обеспечьте полную проверку ввода данных и корректную обработку всех математических случаев. Разработайте UNIT-тесты для модели.

## **Задание 86**

Напишите консольную программу, которая по введённой дате определяет дату, которая наступит через 2 месяца. Учтите переход между годами и корректировку дня при отсутствии соответствующего дня в следующем месяце. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность исходных данных. Разработайте UNIT-тесты для модели.

## **Задание 87**

Напишите JavaFX программу, которая по введённой дате определяет дату, которая наступит через 2 месяца. Учтите переход между годами и корректировку дня при отсутствии соответствующего дня в следующем месяце. Запрещено использовать стандартные классы для работы с датами. Проверяйте корректность исходных данных. Разработайте UNIT-тесты для модели.

## **Задание 88**

Напишите программу, которая читает целое число  $n$  ( $1 \leq n \leq 100$ ), затем  $n$  строк. Используя StringBuilder, обработайте каждую строку: удалите пробелы и добавьте в результат только если длина строки чётная. Между добавленными строками вставьте разделитель — ".

## **Задание 89**

Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 100$ ), затем  $m$  пар: строка-ключ и целое значение. Используя  $\text{HashMap}<\text{String}, \text{Integer}>$ , сохраните данные. Затем для каждого запроса выведите среднее значение (целое) среди тех, чьи ключи имеют ту же длину, что и запрос.

## **Задание 90**

Напишите программу, которая читает  $p$  целых чисел ( $1 \leq p \leq 200$ ). Используя HashSet<Integer>, сохраните уникальные значения. Затем для каждого запроса удалите число из множества (если оно есть) и добавьте его целую часть квадратного корня. В конце выведите элементы в порядке возрастания.

## **Задание 91**

Напишите программу, которая читает  $r$  целых чисел ( $1 \leq r \leq 100$ ). Используя ArrayList<Integer>, сохраните их. Затем обработайте операции: "remove последний" удалить последний элемент списка, только если он положительный. После всех операций выведите список.

## **Задание 92**

Реализуйте иерархию классов для животных: базовый класс Животное с полями имя, возраст, производные классы Собака (порода, команды) и Кошка (окрас, характер). Реализуйте метод издавания звука (гав-гав для собаки, мяу для кошки). Выведите информацию обо всех животных.

## **Задание 93**

Создайте класс «Дробь с единицей измерения» с полями numerator, denominator (int, знаменатель  $\neq 0$ ), unit (String). Реализуйте методы clone, equals, hashCode и toString. При некорректных данных для инициализации вызывайте исключение.

## **Задание 94**

Реализуйте класс MyLinkedList с методом add(int index, int value), добавляющим элемент по индексу (с проверкой границ). Метод remove(int index) удаляет элемент по индексу. Метод print() выводит все элементы. Итератор возвращает элементы в прямом порядке.

## **Задание 95**

Реализуйте класс TriangleClassifier с методом static Classification classify(double a, double b, double c) для классификации треугольника. Classification должен быть sealed-классом с подклассами NotTriangle, Equilateral, IsoscelesRight, Isosceles, ScaleneRight, ScaleneAcuteOrObtuse. Напишите unit-тесты.

## **Задание 96**

Реализуйте потокобезопасный класс EnergyMeter с методами charge(int units), consume(int units) (возвращает true, если энергии хватает) и getLevel(). В main начальный уровень — 1000. Запустите 10 потоков: 5 потоков по 100 раз заряжают на 10, 5 потоков по 100 раз потребляют по 10. После завершения выведите итоговый уровень.

## **Задание 97**

Реализуйте программу с графическим интерфейсом JavaFX для поиска времени окончания интервала. Дано: часы и минуты начала интервала и количество минут, сколько он идет. Результат: часы и минуты окончания интервала. Обеспечьте ввод данных, проверку корректности и вывод результата.

## **Задание 98**

Реализуйте программу с графическим интерфейсом JavaFX, которая читает файл с описанием Label'ов и интервалов времени. После загрузки на форме появляются Label'ы с начальными значениями. Для каждого Label создаётся поток, который с указанным интервалом обновляет значение (увеличивает на 1).

## **Задание 99**

Реализуйте программу с графическим интерфейсом JavaFX для имитации движения спутника. Один шар неподвижен (центр масс), второй — спутник с заданной начальной скоростью. На спутник действует сила притяжения к центру:  $\frac{k}{r^2}$ .