

# Задачник по курсу «Программирование на Java»

Глускер А.И.

3 марта 2026 г.

# Содержание

<b>1</b>	<b>Предварительные сведения</b>	<b>3</b>
<b>2</b>	<b>Семинар №1 «Введение»</b>	<b>4</b>
2.1	Условный оператор if . . . . .	4
2.2	Оператор switch . . . . .	5
2.3	Оператор do...while . . . . .	7
2.4	Цикл for . . . . .	10
2.5	Цикл while . . . . .	12
<b>3</b>	<b>Семинар 2</b>	<b>13</b>
3.1	Задание 1: Манипуляции со строками с помощью StringBuilder . . . . .	13
3.2	Задание 2: Операции с HashMap без упрощений . . . . .	14
3.3	Задание 3: Множества с HashSet и ручными операциями . . . . .	16
3.4	Задание 4: Списки с ArrayList и ручными манипуляциями . . . . .	17
<b>4</b>	<b>Семинар 3 (простейшее ООП)</b>	<b>18</b>
<b>5</b>	<b>Семинар 4 (методы Object)</b>	<b>20</b>
<b>6</b>	<b>Семинар 5 (списки, inner, static классы и Iterator)</b>	<b>22</b>
<b>7</b>	<b>Семинар 6 (generic)</b>	<b>24</b>
<b>8</b>	<b>Семинар 7</b>	<b>24</b>
<b>9</b>	<b>Семинар 8 (sealed class, unit tests)</b>	<b>24</b>
<b>10</b>	<b>Семинар 9 (файлы)</b>	<b>29</b>
<b>11</b>	<b>Семинар 10 (многопоточность)</b>	<b>30</b>
<b>12</b>	<b>Семинар 11 (Spring Boot, REST API)</b>	<b>32</b>
<b>13</b>	<b>Семинар 12 (JWT server)</b>	<b>34</b>
<b>14</b>	<b>Семинар 13 (микросервисы)</b>	<b>34</b>
<b>15</b>	<b>Семинар 14 (kafka)</b>	<b>34</b>
<b>16</b>	<b>Семинар 15</b>	<b>34</b>
<b>17</b>	<b>Семинар 16</b>	<b>34</b>

# 1 Предварительные сведения

## Требования к качеству кода

- **Инкапсуляция:** Все поля классов должны быть `private` с публичными геттерами/сеттерами только при необходимости
- **Валидация данных:** Обязательная проверка корректности входных данных с оповещением пользователя об ошибках (в заданиях не всегда указаны требования к входным данным, их вы должны определить самостоятельно)
- **Обработка исключений:** Использование `try-catch` блоков для обработки возможных ошибок выполнения
- **Документирование:** Javadoc-комментарии для всех классов, методов и нетривиальных полей (начиная с семинара «ООП»)
- **Стиль кода:** Соблюдение соглашений об именовании Java (`camelCase` для методов/переменных, `PascalCase` для классов)

## Требования к функциональности

- **Граничные случаи:** Обработка всех возможных граничных ситуаций (пустые коллекции, нулевые значения, переполнение)
- **Производительность:** Оптимизация алгоритмов до приемлемой сложности
- **Тестируемость:** Разделение логики и ввода/вывода для возможности `unit`-тестирования (начиная с соответствующего семинара)

## Требования к оформлению

- **Читаемость:** Разумное форматирование кода с отступами и пробелами
- **Минимальная сложность:** Упрощение условий и избегание глубокой вложенности
- **Константы вместо магических чисел:** Использование именованных констант для всех значений
- **Разделение ответственности:** Каждый метод должен выполнять одну логическую задачу (начиная с семинара «ООП»)

## Требования к сдаче работ

- Автор должен быть способен объяснить любой фрагмент кода, а также незначительно его модифицировать
- Автор должен знать теоретический материал, относящийся к работе

## 2 Семинар №1 «Введение»

На этом занятии вы познакомитесь с основными управляющими конструкциями языка Java: условными операторами (`if`, `switch`) и циклами (`for`, `while`, `do_while`).

Задачи предназначены для усвоения работы с простейшими операторами, потому некоторые задачи могут быть решены проще с помощью библиотеки, но это делать запрещено... Цель – просто освоить синтаксис и базовые навыки создания программ на Java.

### 2.1 Условный оператор `if`

#### Задачи

Решите следующие задачи, используя условные операторы `if`. Обеспечьте полную проверку ввода данных и корректную обработку всех математических случаев (деление на ноль, вырожденные интервалы, отрицательные значения и т.п.).

1. Напишите программу, которая решает неравенство  $(x - a)(x - b) > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
2. Напишите программу, которая решает неравенство  $(x - a)(x - b) < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
3. Напишите программу, которая решает неравенство  $(x - a)(x - b) \geq 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
4. Напишите программу, которая решает неравенство  $(x - a)(x - b) \leq 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
5. Напишите программу, которая решает неравенство  $(x + a)(x + b) > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
6. Напишите программу, которая решает неравенство  $(x + a)(x + b) < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
7. Напишите программу, которая решает неравенство  $(a - x)(b - x) > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
8. Напишите программу, которая решает неравенство  $(a - x)(b - x) < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
9. Напишите программу, которая решает неравенство  $(x - a)^2 > 0$ , где  $a$  — вещественное число, вводимое пользователем.
10. Напишите программу, которая решает неравенство  $(x - a)^2 \geq 0$ , где  $a$  — вещественное число, вводимое пользователем.
11. Напишите программу, которая решает неравенство  $\frac{x-a}{x-b} > 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
12. Напишите программу, которая решает неравенство  $\frac{x-a}{x-b} < 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.

13. Напишите программу, которая решает неравенство  $\frac{x-a}{x-b} \geq 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
14. Напишите программу, которая решает неравенство  $\frac{x-a}{x-b} \leq 0$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
15. Напишите программу, которая решает неравенство  $|x-a| > b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
16. Напишите программу, которая решает неравенство  $|x-a| < b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
17. Напишите программу, которая решает неравенство  $|x-a| \geq b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
18. Напишите программу, которая решает неравенство  $|x-a| \leq b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
19. Напишите программу, которая решает неравенство  $(x-a)(x-b)(x-c) > 0$ , где  $a, b, c$  — вещественные числа, вводимые пользователем.
20. Напишите программу, которая решает неравенство  $(x-a)(x-b)(x-c) < 0$ , где  $a, b, c$  — вещественные числа, вводимые пользователем.
21. Напишите программу, которая определяет, принадлежит ли точка  $x$  интервалу  $(a; b)$ , где  $a, b, x$  — вещественные числа, вводимые пользователем.
22. Напишите программу, которая определяет, принадлежит ли точка  $x$  отрезку  $[a; b]$ , где  $a, b, x$  — вещественные числа, вводимые пользователем.
23. Напишите программу, которая определяет, лежит ли число  $x$  вне отрезка  $[a; b]$ , где  $a, b, x$  — вещественные числа, вводимые пользователем.
24. Напишите программу, которая решает систему неравенств  $x > a$  и  $x < b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.
25. Напишите программу, которая решает совокупность неравенств  $x < a$  или  $x > b$ , где  $a$  и  $b$  — вещественные числа, вводимые пользователем.

## 2.2 Оператор switch

### Задачи

Решите следующие задачи, используя оператор `switch`. Запрещено использовать стандартные классы для работы с датами — дата задаётся тремя целыми числами: день, месяц, год. Обеспечьте полную проверку корректности ввода (существование даты, високосный год, допустимые диапазоны).

1. По введённой дате определите дату следующего дня. Выведите её и проверьте, совпадают ли количество дней в месяце исходной даты с количеством дней в месяце полученной даты.

2. По введённой дате определите дату предыдущего дня. Выведите её и проверьте, совпадает ли количество дней в месяце исходной даты с количеством дней в месяце полученной даты.
3. По введённой дате определите дату, которая наступит ровно через месяц (прибавить 1 к месяцу, при необходимости корректируя год). Если в следующем месяце нет дня с таким же числом (например, 31 апреля), то возьмите последний день следующего месяца. Выведите полученную дату и проверьте, является ли она последним днём месяца.
4. По введённой дате определите дату, которая была ровно месяц назад (вычесть 1 из месяца, при необходимости корректируя год). Если в предыдущем месяце нет дня с таким же числом, возьмите последний день предыдущего месяца. Выведите полученную дату и проверьте, является ли она первым днём месяца.
5. По введённой дате определите дату, которая наступит через 2 месяца (прибавить 2 к месяцу, корректируя год). Корректировка дня, как в предыдущих задачах. Выведите полученную дату и проверьте, находится ли она в том же квартале года, что и исходная дата. (Кварталы: 1-3, 4-6, 7-9, 10-12)
6. По введённой дате определите дату, которая была 3 месяца назад. Выведите полученную дату и проверьте, находится ли она в том же году, что и исходная дата.
7. По введённой дате определите дату, которая наступит через 1 год (прибавить 1 к году). Учтите високосность года для февраля. Если исходная дата - 29 февраля, то в следующем невисокосном году возьмите 28 февраля. Выведите полученную дату и проверьте, является ли она високосным днём (29 февраля).
8. По введённой дате определите дату, которая была 1 год назад. Выведите полученную дату и проверьте, была ли исходная дата високосным днём (29 февраля), а полученная - нет.
9. По введённой дате определите дату, которая наступит через 100 дней. Выведите её и проверьте, является ли полученная дата последним днём месяца.
10. По введённой дате определите дату, которая была 100 дней назад. Выведите её и проверьте, является ли полученная дата первым днём месяца.
11. По введённой дате определите дату, которая наступит через 1 неделю (7 дней). Выведите её и проверьте, находится ли полученная дата в том же месяце, что и исходная.
12. По введённой дате определите дату, которая была 1 неделю назад. Выведите её и проверьте, находится ли полученная дата в том же году, что и исходная.
13. По введённой дате определите дату, которая наступит через 2 месяца. Выведите её и проверьте, является ли день полученной даты последним днём месяца.
14. По введённой дате определите дату, которая была 2 месяца назад. Выведите её и проверьте, является ли день полученной даты первым днём месяца.
15. По введённой дате определите дату, которая наступит через 6 месяцев. Выведите её и проверьте, находится ли полученная дата во второй половине года (месяц с июля по декабрь).

16. По введённой дате определите дату, которая была 6 месяцев назад. Выведите её и проверьте, находится ли полученная дата в первом полугодии (месяц с января по июнь).
17. По введённой дате определите дату, которая наступит через 1 месяц и 1 день (сначала прибавить месяц, затем день). Корректировка дня, как в задаче 3. Выведите полученную дату и проверьте, является ли она первым днём месяца.
18. По введённой дате определите дату, которая была 1 месяц и 1 день назад (сначала вычесть месяц, затем день). Выведите полученную дату и проверьте, является ли она последним днём месяца.
19. По введённой дате определите дату, которая наступит через 2 года. Выведите её и проверьте, является ли год полученной даты високосным.
20. По введённой дате определите дату, которая была 2 года назад. Выведите её и проверьте, был ли год полученной даты високосным.
21. По введённой дате определите дату, которая наступит через 1 квартал (3 месяца). Выведите её и проверьте, является ли полученная дата последним днём квартала (31 марта, 30 июня, 30 сентября, 31 декабря).
22. По введённой дате определите дату, которая была 1 квартал назад. Выведите её и проверьте, является ли полученная дата первым днём квартала (1 января, 1 апреля, 1 июля, 1 октября).
23. По введённой дате определите дату, которая наступит через 1 год и 1 месяц. Выведите её и проверьте, является ли день полученной даты первым числом месяца.
24. По введённой дате определите дату, которая была 1 год и 1 месяц назад. Выведите её и проверьте, является ли день полученной даты последним числом месяца.
25. По введённой дате определите дату, которая наступит через 366 дней. Выведите её и проверьте, является ли полученная дата високосным днём (29 февраля).

## 2.3 Оператор `do...while`

### Задачи

Решите следующие задачи, используя цикл `do...while`. Все задачи предполагают последовательный ввод чисел, оканчивающийся нулём. Нулевое значение является признаком окончания ввода и в вычислениях **не участвует**. Обеспечьте корректную обработку граничных случаев: пустая последовательность (только 0), отсутствие подходящих чисел, деление на ноль, извлечение корня из отрицательного числа и т.п. При необходимости выводите сообщения об ошибках.

**Указание.** Для целочисленных операций:

1. Остаток при делении  $a$  на  $b$ :  $a \% b$ .
2. Целая часть частного:  $a / b$  (при целочисленном делении).
3. Последняя цифра числа  $n$ :  $n \% 10$ .

4. Предпоследняя цифра:  $(n / 10) \% 10$ .
1. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите максимальное число и количество чисел, больших 5 (кроме завершающего нуля).
  2. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите минимальное число и количество чисел, у которых последняя цифра равна 0 (кроме завершающего нуля).
  3. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите сумму синусов всех чисел и третье число последовательности (если чисел меньше трёх — вывести сообщение об ошибке).
  4. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите сумму всех нечётных чисел и количество чисел, делящихся на 3 (кроме завершающего нуля).
  5. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите количество двузначных натуральных чисел и минимальную последнюю цифру среди всех введённых чисел (кроме завершающего нуля).
  6. Последовательно вводятся натуральные числа, оканчивающиеся нулём. Выведите количество трёхзначных палиндромов (чисел, которые читаются одинаково слева направо и справа налево, например, 121, 343) (кроме завершающего нуля).
  7. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите сумму всех чисел и предпоследнее число последовательности (если чисел меньше двух — вывести сообщение об ошибке).
  8. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите произведение всех чисел (кроме завершающего нуля) и второе число последовательности (если чисел меньше двух — вывести сообщение об ошибке).
  9. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите среднее арифметическое всех чисел и максимум модуля введённых чисел (кроме завершающего нуля).
  10. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее геометрическое всех чисел (кроме завершающего нуля) и минимум модуля введённых чисел.  
**Примечание:** среднее геометрическое определено только для положительных чисел. Если есть неположительные — вывести сообщение об ошибке.  
 Формула:  $(a_1 a_2 \dots a_n)^{1/n}$ .
  11. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее квадратическое всех чисел (кроме завершающего нуля) и минимум квадрата введённых чисел.  
 Формула:  $\sqrt{\frac{a_1^2 + a_2^2 + \dots + a_n^2}{n}}$ .

12. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее гармоническое всех чисел (кроме завершающего нуля) и максимум квадрата введённых чисел.

**Примечание:** среднее гармоническое не определено, если есть нули или числа разных знаков. Проверьте знаменатель.

Формула:  $\frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \dots + \frac{1}{a_n}}$ .

13. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее арифметическое модулей всех чисел и максимум синусов введённых чисел (кроме завершающего нуля).

14. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее гармоническое модулей всех чисел (кроме завершающего нуля) и минимум синусов введённых чисел.

**Примечание:** модули положительны — среднее гармоническое определено, если только не все числа нулевые.

15. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее квадратическое модулей всех чисел (кроме завершающего нуля) и минимум косинусов введённых чисел.

16. Последовательно вводятся вещественные числа, оканчивающиеся нулём. Выведите среднее геометрическое модулей всех чисел (кроме завершающего нуля) и максимум косинусов введённых чисел.

**Примечание:** модули неотрицательны — если есть ноль, среднее геометрическое = 0.

17. Последовательно вводятся натуральные числа, оканчивающиеся нулём. Выведите среднее арифметическое квадратов всех чисел (кроме завершающего нуля) и максимальную последнюю цифру среди всех чисел.

18. Последовательно вводятся натуральные числа, оканчивающиеся нулём. Выведите среднее геометрическое квадратов всех чисел (кроме завершающего нуля) и минимальную последнюю цифру среди всех чисел.

19. Последовательно вводятся натуральные числа, оканчивающиеся нулём. Выведите среднее квадратическое квадратов всех чисел (кроме завершающего нуля) и максимальную предпоследнюю цифру среди всех чисел.

20. Последовательно вводятся натуральные числа, оканчивающиеся нулём. Выведите среднее гармоническое квадратов всех чисел (кроме завершающего нуля) и минимальную предпоследнюю цифру среди всех чисел.

21. Последовательно вводятся натуральные числа от 1 до 999, оканчивающиеся нулём. Выведите максимальную сумму цифр в числах и среднее арифметическое сумм цифр (кроме завершающего нуля).

22. Последовательно вводятся натуральные числа от 1 до 999, оканчивающиеся нулём. Выведите минимальную сумму цифр в числах и среднее гармоническое сумм цифр (кроме завершающего нуля).

23. Последовательно вводятся натуральные числа от 1 до 999, оканчивающиеся нулём. Выведите минимальную сумму количества сотен и единиц в числах и среднее геометрическое сумм цифр (кроме завершающего нуля).

**Пример:** для числа 347: сотни = 3, единицы = 7, сумма = 10.

24. Последовательно вводятся натуральные числа от 1 до 999, оканчивающиеся нулём. Выведите максимальную сумму количества сотен и единиц в числах и среднее квадратическое сумм цифр (кроме завершающего нуля).

25. Последовательно вводятся целые числа, оканчивающиеся нулём. Выведите среднее геометрическое всех чётных чисел (кроме завершающего нуля) и максимум среди нечётных чисел.

**Примечание:** если чётных чисел нет — вывести сообщение об ошибке. Учтите, что среднее геометрическое требует положительных значений.

## 2.4 Цикл for

### Задачи

Решите следующие задачи, используя цикл `for`. Все задачи должны использовать именно `for` (не `while` или `do...while`). Обеспечьте корректную обработку граничных случаев: деление на ноль, отрицательные числа, пустые диапазоны и т.п.

1. Найдите количество трёхзначных чисел в диапазоне  $[100; 999]$ , в которых вторая цифра равна сумме первой и третьей цифры.
2. Найдите количество трёхзначных чисел в диапазоне  $[100; 999]$ , в которых сумма первых двух цифр равна третьей цифре.
3. Найдите количество трёхзначных чисел в диапазоне  $[100; 999]$ , в которых сумма последних двух цифр равна первой цифре.

4. Найдите все натуральные числа в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 999$ ), которые равны сумме квадратов своих цифр.

**Пример:**  $1^2 + 3^2 + 0^2 = 10$  — не подходит;  $1^2 + 6^2 + 3^2 = 46$  — не подходит.

5. Найдите все натуральные числа в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 999$ ), которые равны сумме кубов своих цифр.

**Пример:**  $153 = 1^3 + 5^3 + 3^3$  — подходит.

6. Найдите все натуральные числа в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 999$ ), которые равны сумме своих цифр.

**Пример:**  $18 = 1 + 8 = 9$  — не подходит;  $1 = 1$  — подходит.

7. Найдите все натуральные делители числа  $n \in \mathbb{N}$  ( $n > 0$ ). Выведите их в порядке возрастания.

8. Определите, является ли число  $n \in \mathbb{N}$  ( $n > 1$ ) простым. Выведите «Да» или «Нет».

9. Найдите все натуральные числа в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 999$ ), которые делятся на свою последнюю цифру.

**Примечание:** если последняя цифра — 0, число не учитывается (деление на ноль).

10. Напечатайте таблицу перевода двоичных чисел от  $1_2$  до  $11111_2$  (т.е. от 1 до 31 в десятичной) в десятичную систему счисления.
11. Напечатайте таблицу перевода восьмеричных чисел от  $1_8$  до  $777_8$  (т.е. от 1 до 511 в десятичной) в десятичную систему счисления.
12. Напечатайте таблицу умножения (от  $1 \times 1$  до  $10 \times 10$ ).
13. Напечатайте первые 20 чисел Фибоначчи ( $f_1 = 1, f_2 = 1, f_n = f_{n-1} + f_{n-2}$  для  $n > 2$ ).
14. Найдите все трёхзначные числа в диапазоне  $[100; 999]$ , которые при зачёркивании средней цифры уменьшаются в 7 раз.  
**Пример:** число  $357 \rightarrow$  зачёркиваем 5  $\rightarrow$  получаем  $37; 357/37 = 9.648$  — не подходит.
15. Найдите сумму всех натуральных делителей числа  $n \in \mathbb{N}$  ( $n > 0$ ).
16. Вычислите  $a^n$ , где  $a \in \mathbb{R}, n \in \mathbb{Z}, n \geq 0$ .  
**Примечание:** если  $n < 0$ , вывести сообщение об ошибке. Используйте только умножение (не `Math.pow`).
17. Найдите сумму всех нечётных натуральных чисел в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 1000$ ).
18. Найдите сумму всех чётных натуральных чисел в диапазоне  $[m; n]$  ( $1 \leq m \leq n \leq 1000$ ).
19. Найдите все общие делители натуральных чисел  $n$  и  $m$  ( $n > 0, m > 0$ ). Выведите их в порядке возрастания.
20. Найдите все натуральные числа в диапазоне  $[m; n]$  ( $10 \leq m \leq n \leq 999$ ), которые делятся на свою предпоследнюю цифру.  
**Примечание:** если предпоследняя цифра — 0, число не учитывается.
21. Напечатайте все трёхзначные палиндромы (числа, которые читаются одинаково слева направо и справа налево, например, 121, 343) в диапазоне  $[100; 999]$ .
22. Найдите все трёхзначные числа в диапазоне  $[100; 999]$ , которые кратны числу, составленному из второй и третьей цифр.  
**Пример:** число 135  $\rightarrow$  вторая и третья цифры = 35;  $135/35 = 3.857$  — не целое  $\rightarrow$  не подходит.  
**Уточнение:** пропорциональны = делятся без остатка.
23. Найдите все четырёхзначные числа в диапазоне  $[1000; 9999]$ , в которых сумма первых двух цифр равна сумме последних двух цифр.
24. Найдите все четырёхзначные числа в диапазоне  $[1000; 9999]$ , в которых сумма крайних цифр равна сумме средних цифр.
25. Найдите все четырёхзначные числа в диапазоне  $[1000; 9999]$ , в которых сумма первой и третьей цифр равна сумме второй и четвёртой цифр.

## 2.5 Цикл `while`

### Задачи

Решите следующие задачи, используя цикл `while`. Использование `for` или `do...while` не допускается. Все задачи предполагают, что количество итераций заранее неизвестно и определяется в процессе выполнения. Обеспечьте обработку граничных случаев: нули, единицы, отрицательные числа, переполнения.

1. Дано натуральное число  $n$ . Найдите сумму его цифр, используя `while`.
2. Дано натуральное число  $n$ . Найдите количество его цифр, используя `while`.
3. Дано натуральное число  $n$ . Найдите произведение его цифр, используя `while`.
4. Дано натуральное число  $n$ . Определите, является ли оно палиндромом (читается одинаково слева направо и справа налево), используя `while`.  
**Указание:** постройте зеркальное число и сравните.
5. Дано натуральное число  $n$ . Удалите из него все чётные цифры и выведите результат (если получилось пустое число — вывести 0). Используйте `while`.
6. Дано натуральное число  $n$ . Проверьте, является ли оно факториалом какого-либо натурального числа. Если да — выведите это число, иначе — сообщение «Не является факториалом».  
**Пример:**  $120 = 5! \rightarrow$  вывести 5.
7. Дано натуральное число  $n$ . Найдите наименьшее  $k$ , такое что  $k! \geq n$ . Используйте `while`.
8. Дано натуральное число  $n$ . Разложите его на простые множители и выведите их в порядке возрастания (с повторениями). Используйте `while`.
9. Даны два натуральных числа  $a$  и  $b$ . Найдите их наибольший общий делитель (НОД) с помощью алгоритма Евклида, используя `while`.
10. Даны два натуральных числа  $a$  и  $b$ . Найдите их наименьшее общее кратное (НОК), используя `while` и НОД.
11. Дано натуральное число  $n$ . Переведите его в двоичную систему счисления, используя `while`. Выведите результат как число (не строку).
12. Дано натуральное число  $n$ . Переведите его в восьмеричную систему счисления, используя `while`. Выведите результат как число.
13. Дано натуральное число  $n$ . Определите, сколько раз в нём встречается цифра 7, используя `while`.
14. Дано натуральное число  $n$ . Найдите максимальную цифру в числе, используя `while`.
15. Дано натуральное число  $n$ . Найдите минимальную цифру в числе, используя `while`.
16. Дано натуральное число  $n$ . Определите, содержит ли оно хотя бы одну цифру, равную 0, используя `while`.

17. Дано натуральное число  $n$ . Определите, все ли его цифры нечётные, используя `while`.
18. Дано натуральное число  $n$ . Найдите число, составленное из его цифр в обратном порядке (зеркальное отражение), используя `while`.
19. Дано натуральное число  $n$ . Определите, является ли оно степенью двойки (т.е.  $n = 2^k$  для некоторого  $k \geq 0$ ), используя `while`.
20. Дано натуральное число  $n$ . Определите, является ли оно степенью тройки, используя `while`.
21. Дано натуральное число  $n$ . Найдите сумму всех его делителей, используя `while`.
22. Дано натуральное число  $n$ . Определите, является ли оно совершенным (т.е. сумма его собственных делителей равна самому числу), используя `while`.
23. Дано натуральное число  $n$ . Найдите количество нулей в его двоичном представлении, используя `while`.
24. Дано натуральное число  $n$ . Найдите количество единиц в его двоичном представлении, используя `while`.
25. Дано натуральное число  $n$ . Определите, можно ли его представить в виде суммы двух квадратов натуральных чисел, используя `while`.  
**Пример:**  $25 = 3^2 + 4^2 \rightarrow$  можно.

## 3 Семинар 2

### 3.1 Задание 1: Манипуляции со строками с помощью `StringBuilder`

**Описание:** Напишите программу, которая читает с клавиатуры целое число  $n$  ( $1 \leq n \leq 1000$ ), затем  $n$  строк (каждая длиной до 100 символов). Используя только `StringBuilder` для сборки результата, обработайте каждую строку согласно условию варианта (например, инвертируйте, удалите символы и т.д.), затем добавьте её в результат, если она удовлетворяет фильтру варианта (например, длина, наличие символов). Между добавленными строками вставьте фиксированный разделитель — ". В конце добавьте общее количество символов в результате. Выводите на экран. Запрещено использовать `String` methods вроде `reverse`, `replaceAll`, `format`, `join`; все операции вручную через циклы и `append/insert/delete`.

1. Обработка: инвертировать строку; Фильтр: длина  $> 5$
2. Обработка: удалить все гласные; Фильтр: начинается с согласной
3. Обработка: удвоить каждый символ; Фильтр: содержит цифру
4. Обработка: перевести в верхний регистр; Фильтр: заканчивается на 'а'
5. Обработка: удалить пробелы; Фильтр: длина четная
6. Обработка: добавить '!' в конец; Фильтр: не содержит 'е'
7. Обработка: заменить 'а' на '@'; Фильтр: больше 3 гласных

8. Обработка: удалить дубликаты символов; Фильтр: все символы уникальны
9. Обработка: отсортировать символы по алфавиту (вручную); Фильтр: длина  $< 10$
10. Обработка: добавить индекс в начало; Фильтр: содержит специальный символ
11. Обработка: обернуть в скобки; Фильтр: начинается с цифры
12. Обработка: удалить последние 2 символа; Фильтр: длина  $\geq 3$
13. Обработка: повторить строку twice; Фильтр: не пустая
14. Обработка: заменить пробелы на ' \_ '; Фильтр: содержит пробел
15. Обработка: удалить все цифры; Фильтр: была хотя бы одна цифра
16. Обработка: инвертировать регистр; Фильтр: смешанный регистр
17. Обработка: добавить длину в конец; Фильтр: длина делится на 3
18. Обработка: удалить гласные в начале; Фильтр: заканчивается гласной
19. Обработка: удвоить гласные; Фильтр: только гласные
20. Обработка: заменить согласные на '\*'; Фильтр: больше согласных
21. Обработка: добавить 'prefix-'; Фильтр: не начинается с 'r'
22. Обработка: удалить середину (если длина  $> 2$ ); Фильтр: длина нечетная
23. Обработка: циклический сдвиг влево; Фильтр: длина  $> 1$
24. Обработка: циклический сдвиг вправо; Фильтр: содержит 'z'
25. Обработка: удалить все кроме букв; Фильтр: была не-буква

### 3.2 Задание 2: Операции с HashMap без упрощений

**Описание:** Напишите программу, которая читает целое число  $m$  ( $1 \leq m \leq 500$ ), затем  $m$  пар: строка-ключ (до 50 символов) и целое значение (-1000..1000). Используя `HashMap<String, Integer>`, сохраните (при дубликатах ключей суммируйте значения). Затем прочитайте  $k$  ( $1 \leq k \leq 100$ ) запросов, каждый — строка. Для каждого запроса выполните операцию варианта над значениями, чьи ключи соответствуют условию варианта (например, начинаются с запроса, содержат и т.д.), и выведите результат. Если ничего не найдено, выведите 0. Запрещено использовать `streams`, `Collectors`, `computeIf`; все через циклы по `keySet` или `entrySet`, `contains`, `get`, `put`.

1. Операция: сумма значений; Условие: ключ начинается с запроса
2. Операция: максимум значения; Условие: ключ заканчивается запросом
3. Операция: количество ключей; Условие: ключ содержит запрос
4. Операция: минимум значения; Условие: ключ равен запросу (`equals`)

5. Операция: среднее значение (int); Условие: длина ключа = длине запроса
6. Операция: произведение значений; Условие: ключ лексикографически > запрос
7. Операция: сумма квадратов; Условие: ключ имеет подстроку запрос реверс
8. Операция: количество положительных; Условие: ключ в нижнем регистре содержит запрос
9. Операция: максимум по модулю; Условие: ключ без гласных содержит запрос
10. Операция: сумма только четных; Условие: ключ с цифрами содержит запрос
11. Операция: количество уникальных значений; Условие: ключ короче запроса
12. Операция: минимум среди отрицательных; Условие: ключ длиннее запроса
13. Операция: сумма абсолютных; Условие: ключ стартует с реверса запроса
14. Операция: произведение нечетных; Условие: ключ в верхнем регистре = запрос
15. Операция: количество нулей; Условие: ключ содержит запрос дважды
16. Операция: максимум среди четных; Условие: ключ без пробелов = запрос
17. Операция: сумма делимых на 3; Условие: ключ с удаленными цифрами содержит запрос
18. Операция: минимум по квадрату; Условие: ключ инвертированный содержит запрос
19. Операция: количество > среднего; Условие: ключ с удвоенными символами содержит запрос
20. Операция: произведение положительных; Условие: ключ без последних 2 символов = запрос
21. Операция: сумма первых цифр значений; Условие: ключ с префиксом "a" содержит запрос
22. Операция: максимум разницы с min; Условие: ключ циклически сдвинутый содержит запрос
23. Операция: количество пар значений; Условие: ключ с заменой 'a' на 'b' содержит запрос
24. Операция: сумма факториалов (малых); Условие: ключ только буквы содержит запрос
25. Операция: минимум среди делимых на 5; Условие: ключ с добавленным суффиксом содержит запрос

### 3.3 Задание 3: Множества с HashSet и ручными операциями

**Описание:** Напишите программу, которая читает  $p$  ( $1 \leq p \leq 800$ ), затем  $p$  целых чисел ( $1..10000$ ). Используя `HashSet<Integer>`, сохраните уникальные. Затем прочитайте  $q$  ( $1 \leq q \leq 200$ ) запросов, каждый — целое число. Для каждого выполните действие варианта: например, если есть, удалите и добавьте трансформацию (квадрат, удвоение и т.д.), если трансформация уже есть, пропустите. В конце выведите элементы в порядке возрастания (сортируйте вручную в массив, без `TreeSet` или `sorted`). Запрещено использовать `containsAll`, `addAll`; все через `add`, `remove`, `contains`, `iterator`.

1. Действие: удалить и добавить квадрат
2. Действие: удалить и добавить удвоенное
3. Действие: удалить и добавить  $+1$
4. Действие: удалить и добавить факториал (малый)
5. Действие: удалить и добавить корень (int)
6. Действие: удалить и добавить обратное ( $1/x$  если  $!=0$ )
7. Действие: удалить и добавить модуль
8. Действие: удалить и добавить сумму цифр
9. Действие: удалить и добавить произведение цифр
10. Действие: удалить и добавить реверс цифр
11. Действие: удалить и добавить  $+100$
12. Действие: удалить и добавить  $-50$
13. Действие: удалить и добавить куб
14. Действие: удалить и добавить  $\log_2$  (int)
15. Действие: удалить и добавить fib next (простой fib)
16. Действие: удалить и добавить prime next
17. Действие: удалить и добавить делимое на 3
18. Действие: удалить и добавить битовый сдвиг
19. Действие: удалить и добавить XOR 42
20. Действие: удалить и добавить AND 255
21. Действие: удалить и добавить кол-во бит 1
22. Действие: удалить и добавить row 2
23. Действие: удалить и добавить div 2
24. Действие: удалить и добавить mul 3
25. Действие: удалить и добавить mod 100

### 3.4 Задание 4: Списки с ArrayList и ручными манипуляциями

**Описание:** Напишите программу, которая читает  $r$  ( $1 \leq r \leq 600$ ), затем  $r$  целых ( $-5000..5000$ ). Используя `ArrayList<Integer>`, сохраните. Затем прочитайте  $s$  ( $1 \leq s \leq 150$ ) операций, каждая в формате строки (парсите вручную без `split` упрощений). Операция варианта: например, "add X Y добавить X в Y, но с условием; "remove Z удалить Z если условие; "swap A B swap если разница  $>k$ . После операций выведите список по условию варианта (реверс, только четные и т.д.) через цикл, без `Collections.reverse/sort`. Запрещено использовать `subList`, `sort`, `reverse`; все `get/set/add/remove` вручную.

1. Операция: add если  $X > 0$ , в позицию  $Y \bmod \text{size}$
2. Операция: remove если  $Z$  четный
3. Операция: swap если  $A+B$  even
4. Операция: add X в начало если  $X$  odd
5. Операция: remove последний если  $> 0$
6. Операция: swap первый и последний если  $\text{size} > 1$
7. Операция: add X в конец если not contains
8. Операция: remove по значению если exists
9. Операция: swap если  $|A-B| > 10$
10. Операция: add если  $X$  prime
11. Операция: remove если divisible 5
12. Операция: swap random (но fixed seed)
13. Операция: add в середину
14. Операция: remove дубликаты (ручной)
15. Операция: swap соседние
16. Операция: add сумму соседей
17. Операция: remove min
18. Операция: swap max и min
19. Операция: add среднее
20. Операция: remove  $> \text{average}$
21. Операция: swap если both positive
22. Операция: add квадрат last
23. Операция: remove first negative
24. Операция: swap every other
25. Операция: add 0 в позиции multiples 3

## 4 Семинар 3 (простейшее ООП)

Напишите программу в соответствии с заданием, используя объектно-ориентированный стиль программирования. В программе должны быть отражены свойства объектно-ориентированного программирования: инкапсуляция, наследование и полиморфизм (наряду с этим в некоторых вариантах нужно реализовывать абстрактные классы и методы).

Обращаем внимание, что каждый класс следует поместить в отдельный файл. Все программы должны работать в консольном режиме (без графического интерфейса) и использовать только стандартную библиотеку Java.

1. Программа работы со списком работников. Каждый работник определяется фамилией, именем и отчеством, должностью (преподаватель и лаборант). Для преподавателя указывается количество часов в год, а для лаборанта – количество ставок. Дополнительно в программу вводится стоимость одного часа и стоимость ставки (за год). После ввода необходимо вывести на экран список работников в порядке возрастания оплаты за год, при этом в списке должны быть указаны ФИО, должность, количество часов/количество ставок и «стоимость» работника.
2. Программа работы со списком учебных заведений (школ и ВУЗов). Школа определяется номером, количеством учащихся и специализацией (физ-мат, гуманитарный); ВУЗ – названием, количеством студентов, наличием магистратуры, наличием аспирантуры. Программа должна предоставлять возможность ввести информацию о ВУЗах и школах, после чего вывести информацию о школах/ВУЗах в порядке убывания количества учащихся (вне зависимости от типа учебного заведения). В списке должна выводиться вся информация, что была введена.
3. Программа суммирования последовательностей двух типов:  $\frac{n}{1!} + \frac{n+1}{2!} + \dots + \frac{n+m}{(m+1)!}$  и  $\frac{n}{2!} + \dots + \frac{n+m}{2^{m+1}}$ .  $n$  и  $m$  вводятся с клавиатуры. Реализовать вычисление через иерархию классов.
4. Программа нахождения интеграла методом прямоугольников для функций двух видов:  $ax^3 + bx^2 + cx + d$  и  $a \sin x + be^x$ . Реализовать функции в виде отдельных классов.
5. Программа решения уравнений двух видов методом дихотомии:  $ax^3 + bx^2 + cx + d = 0$  и  $a \sin x + be^x = c$ . Реализовать функции в виде отдельных классов.
6. У игрока может быть несколько предметов экипировки (до 10): бластеры (с индикатором количества заряда и уровня бластера от 1 до 5), витамины (с количеством оставшихся таблеток), плащи (характеризуются уровнем защиты). Написать программу, которая вводит с клавиатуры информацию об имеющихся игровых принадлежностях, после чего выводит информацию на экран. Данная программа (в части вывода данных) может быть фрагментом игры.
7. Пользователь задает простейший тест, состоящий из вопросов двух видов: с выбором варианта ответа и с вводом верного ответа; после чего компьютер тестирует (другого) пользователя по введенному тесту (в консольном режиме).
8. Создайте программу сортировки массива натуральных чисел, которая сортирует по выбору пользователя: 1) по возрастанию; 2) по убыванию; 3) по возрастанию сумм цифр в числе; 4) по убыванию сумм цифр в числе.

9. Создайте программу вывода всех элементов заданного массива по выбору пользователя: 1) в прямом порядке; 2) в обратном порядке; 3) в случайном порядке; 4) в челночном порядке (первый-последний-второй-предпоследний и т. д.).
10. Напишите программу-игру «чет-нечет». Один игрок загадывает «чет» или «нечет», а второй угадывает. За один раунд идет 10 угадываний. Пользователь выбирает в начале работы программы ее режим работы: пользователь-компьютер, компьютер-пользователь, компьютер-компьютер или пользователь-пользователь.
11. Пользователь имеет несколько счетов трех видов: первый вид характеризуется тем, что за его использование с него списывается 1 рубль в месяц, второй – тем, что количество денег на нем увеличивается на 1% в месяц, третий – тем, что с вероятностью 50% количество денег на нем за месяц не меняется, с вероятностью 50% – увеличивается на 2%. Пользователь задает список своих счетов с указанием количества денег на них. После чего программа должна вывести таблицу изменения сумм, размещенных на указанных счетах, в течение года.
12. Пользователь задает информацию о своих контактных сведениях (хранение в памяти программы): он может задать один или несколько телефонов, один или несколько адресов и т. д.: телефон (код города + сам телефон), адрес (город, улица, дом, корпус, квартира), tg, e-mail; после чего он может выводить список контактов, изменять информацию по контактному сведению любого вида, добавлять и удалять контакт.
13. Программа нахождения производной для функций двух видов:  $ax^3 + bx^2 + cx + d$  и  $a \sin x + be^x$  (результат – формула). Реализовать функции в виде отдельных классов.
14. Программа библиотеки (хранение в памяти), в которой хранятся книги (описываются автором, названием, количеством страниц) и CD-диски (название CD, производитель, количество треков). Программа должна позволять добавлять в библиотеку книги и CD-диски, а также выводить на экран содержимое библиотеки.
15. Конфигуратор компьютеров. Пользователь выбирает конфигурацию компьютера: процессор (марка, быстродействие), один или несколько жестких дисков (марка, емкость), клавиатуру, мышь, принтеры (не обязательно, марка, тип). После чего ему выводится на экран полная информация о компьютере (включая стоимость).
16. Формирование заказа в магазине: пользователь выбирает тип товара: рубашка (указывается ее размер), ткань (указывается длина и ширина), нитки (выбирается цвет и длина). После чего ему выводится полная информация о заказе (включая стоимость).
17. Пользователь магазина выбирает конфигурацию велосипеда, который будет собран для него: тип рамы (обычная, женская, изогнутая); колеса (размер - 24, 26 или 28); велокомпьютер (может отсутствовать, если есть, то выбирается беспроводной он или нет); амортизатор (может отсутствовать, если есть, то – одноподвес или двухподвес). После чего ему выводится на экран полная информация о получившемся велосипеде.
18. Программа работы с геометрическими фигурами (окружность, прямоугольник, треугольник). Для каждой фигуры вводятся параметры (радиус, стороны и т.д.). Программа должна вычислять площадь и периметр каждой фигуры, а затем выводить список фигур в порядке убывания площади (без графического отображения).

19. Программа учета транспортных средств (легковой автомобиль, грузовик, автобус). Для каждого ТС указываются марка, расход топлива на 100 км и вместимость. Программа должна позволять рассчитать стоимость поездки заданной длины с учетом цены топлива.
20. Программа имитации файловой системы. Элементы системы делятся на файлы (имеют размер) и папки (содержат другие элементы в памяти). Программа должна выводить структуру папок и файлов текстом, а также общий размер каждой папки (сумма размеров вложенных файлов).
21. Программа бронирования номеров в отеле. Номера бывают трех видов: стандарт, люкс, апартаменты (различаются стоимостью за сутки и набором услуг). Пользователь задает список забронированных номеров и количество дней, после чего программа выводит счет для каждого клиента.
22. Программа службы доставки. Виды доставки: почта, курьер, экспресс (различаются стоимостью за кг и скоростью). Пользователь вводит параметры посылок (вес, расстояние), программа выбирает оптимальный способ доставки или рассчитывает стоимость для каждого вида.
23. Программа записи к врачу в клинике. Врачи специализации: терапевт, хирург, стоматолог (различаются стоимостью приема и длительностью). Программа позволяет добавить записи в расписание и вывести список приемов с указанием времени, врача и стоимости.
24. Программа имитации отправки уведомлений. Типы уведомлений: SMS, e-mail, push-сообщение (различаются длиной текста, стоимостью отправки и скоростью доставки). Программа формирует очередь уведомлений и выводит отчет об отправке (статус, стоимость) в консоль.
25. Программа управления устройствами «умного дома» (консольная имитация). Устройства: лампа (вкл/выкл, яркость), термостат (температура, режим), камера (запись/отдых). Программа позволяет изменять состояние устройств и выводить сводку по всем устройствам в доме.

## 5 Семинар 4 (методы Object)

Создайте класс в соответствии с вашим вариантом, реализуйте корректно в нём методы `clone`, `equals`, `hashCode`, `toString`.

В случае некорректности данных для инициализации, вызывайте исключение.

Замечание: это учебное задание, мы дублируем логику `record`.

1. Класс «Точка в 3D»: поля `x`, `y`, `z` (тип `double`).
2. Класс «Цвет в RGB»: поля `red`, `green`, `blue` (тип `int`, значения от 0 до 255).
3. Класс «Время суток»: поля `hour` (0–23), `minute` (0–59), `second` (0–59) (все — `int`).
4. Класс «Геокоординаты с высотой»: поля `latitude`, `longitude` (`double`), `altitude` (`int`, в метрах).

5. Класс «Дробь с единицей измерения»: поля `numerator`, `denominator` (`int`, знаменатель  $\neq 0$ ), `unit` (`String`).
6. Класс «Комплексное число с меткой»: поля `re`, `im` (`double`), `label` (`String`).
7. Класс «Размер экрана с ориентацией»: поля `width`, `height` (`int`), `orientation` (`String`: "portrait" или "landscape").
8. Класс «Погодные данные»: поля `temp` (температура в °C, `double`), `humidity` (влажность в %, `int`), `pressure` (давление в мм рт.ст., `int`).
9. Класс «Скорость в 3D»: поля `vx`, `vy`, `vz` (тип `double`).
10. Класс «Пиксель»: поля `x`, `y` (`int`), `brightness` (`int`, 0–255).
11. Класс «Дата»: поля `day` (1–31), `month` (1–12), `year` (`int`,  $\geq 1900$ ).
12. Класс «Валютная пара с курсом»: поля `base`, `quote` (`String`), `rate` (`double`).
13. Класс «Углы Эйлера»: поля `yaw`, `pitch`, `roll` (в градусах, `double`).
14. Класс «Физическая величина»: поля `value` (`double`), `unit` (`String`), `precision` (`int`, знаков после запятой).
15. Класс «Настройки звука»: поля `left`, `right`, `master` (`int`, 0–100).
16. Класс «Коэффициенты плоскости»: поля `a`, `b`, `c` (`double`, уравнение  $ax + by + cz = 0$ ).
17. Класс «Шахматная позиция»: поля `file` (`char`, от 'a' до 'h'), `rank` (`int`, 1–8), `piece` (`String`, например "king").
18. Класс «Интервал с единицей измерения»: поля `start`, `end` (`double`), `unit` (`String`).
19. Класс «Разрешение с частотой»: поля `width`, `height` (`int`), `refreshRate` (`int`, в Гц).
20. Класс «Именованный вектор»: поля `x`, `y`, `z` (`double`), `name` (`String`).
21. Класс «Позиция с меткой времени»: поля `x`, `y` (`double`), `timestamp` (`long`).
22. Класс «Параметры изображения»: поля `brightness`, `contrast`, `saturation` (`int`, 0–100).
23. Класс «Квадратный трёхчлен»: поля `a`, `b`, `c` (`double`,  $a \neq 0$ ).
24. Класс «GPS-точка»: поля `lat`, `lon` (`double`), `accuracy` (`float`, в метрах).
25. Класс «Финансовая операция»: поля `amount` (`double`), `currency` (`String`, например "RUB"), `date` (`String` в формате "YYYY-MM-DD").

## 6 Семинар 5 (списки, inner, static классы и Iterator)

Реализуйте список (самостоятельно, без использования библиотеки) согласно вашему варианту. В необходимых случаях используйте static и inner-классы.

1. 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.
2. Реализуйте класс `MyLinkedList` с методом `add(int value)`, добавляющим элемент в начало. Метод `remove(int index)` удаляет элемент по индексу. Метод `print()` выводит все элементы. Итератор возвращает элементы в порядке хранения.
3. 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.
4. 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.
5. 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.
6. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeLast()` удаляет последний элемент. Метод `print()` выводит все элементы. Итератор возвращает все элементы.
7. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет только уникальные значения. Метод `remove(int value)` удаляет все вхождения. Метод `print()` выводит все элементы. Итератор возвращает элементы без дубликатов (в порядке первого вхождения).
8. Реализуйте класс `MyLinkedList` с методом `addAll(int[] values)`, добавляющим все элементы массива в конец. Метод `remove(int value)` удаляет все вхождения. Метод `print(int k)` выводит первые  $k$  элементов. Итератор возвращает все элементы.
9. Реализуйте класс `MyLinkedList`, в котором `add(int value)` вставляет элемент перед первым значением, большим `value` (иначе — в конец). Метод `removeAbove(int threshold)` удаляет все элементы  $> \text{threshold}$ . Метод `print()` выводит все элементы. Итератор возвращает оставшиеся элементы.
10. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeFirst()` удаляет первый элемент. Метод `printEven()` выводит только чётные элементы. Итератор возвращает только чётные значения.

11. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в начало. Метод `removeBelow(int threshold)` удаляет все элементы  $< \text{threshold}$ . Метод `printOdd()` выводит только нечётные элементы. Итератор возвращает только нечётные значения.
12. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeZeros()` удаляет все нули. Метод `printPositive()` выводит только положительные элементы. Итератор возвращает только положительные числа.
13. Реализуйте класс `MyLinkedList` с методом `add(int index, int value)`. Метод `removeEvenIndices` удаляет элементы с чётными индексами. Метод `printOddIndices()` выводит элементы с нечётными индексами. Итератор возвращает элементы с нечётными индексами.
14. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в начало. Метод `removeDivisibleBy(int n)` удаляет все элементы, делящиеся на  $n$ . Метод `print()` выводит оставшиеся элементы. Итератор возвращает элементы, не делящиеся на  $n$ .
15. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeSmall(int minAbs)` удаляет элементы с  $|x| < \text{minAbs}$ . Метод `printLarge(int minAbs)` выводит элементы с  $|x| \geq \text{minAbs}$ . Итератор возвращает такие элементы.
16. Реализуйте класс `MyLinkedList` с ограниченной ёмкостью  $N$ : `add(int value)` добавляет в конец, при переполнении удаляется первый элемент. Метод `remove(int value)` удаляет первое вхождение. Метод `print()` выводит все элементы. Итератор возвращает все элементы в порядке хранения.
17. Реализуйте класс `MyLinkedList`, в котором `add(int value)` вставляет с сохранением невозрастающего порядка. Метод `removeMax()` удаляет первое вхождение максимального элемента. Метод `print()` выводит все элементы. Итератор возвращает элементы по порядку.
18. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeDuplicates()` оставляет только первые вхождения. Метод `printUnique()` выводит уникальные элементы. Итератор возвращает элементы без повторений.
19. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в начало. Метод `removeNegative()` удаляет все отрицательные элементы. Метод `printNonNegative()` выводит оставшиеся. Итератор возвращает только неотрицательные числа.
20. Реализуйте класс `MyLinkedList` с методом `addAfterZero(int value)`, добавляющим `value` после каждого нуля. Метод `removeZeroPairs()` удаляет все пары  $(0, x)$ . Метод `print()` выводит результат. Итератор возвращает элементы, не следующие непосредственно за 0.
21. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeIndexEqualsValue()` удаляет элементы, у которых значение равно их индексу. Метод `print()` выводит все элементы. Итератор возвращает элементы, у которых значение  $\neq$  индекс.

22. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в начало. Метод `removeAverage()` удаляет все элементы, равные округлённому среднему арифметическому списка. Метод `print()` выводит все элементы. Итератор возвращает все элементы.
23. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeLocalMinima()` удаляет локальные минимумы (элементы, меньшие обоих соседей). Метод `print()` выводит оставшиеся элементы. Итератор возвращает все элементы в исходном порядке.
24. Реализуйте класс `MyLinkedList`, в котором `add(int value)` вставляет с сохранением неубывающего порядка. Метод `removeRange(int a, int b)` удаляет все элементы в диапазоне  $[a, b]$ . Метод `print()` выводит оставшиеся элементы. Итератор возвращает элементы вне диапазона.
25. Реализуйте класс `MyLinkedList`, в котором `add(int value)` добавляет в конец. Метод `removeEverySecond()` удаляет каждый второй элемент (индексы 1, 3, 5, ...). Метод `print()` выводит оставшиеся элементы. Итератор возвращает элементы с чётными индексами (0, 2, 4, ...).

## 7 Семинар 6 (generic)

Реализуйте задачу предыдущего семинара, но реализуемый класс должен быть обобщённым (Generic).

## 8 Семинар 7

Проверочная работа по материалам прошлых семинаров

## 9 Семинар 8 (sealed class, unit tests)

Реализуйте решение задачи с возвратом результата в виде Sealed-классов, консольный интерфейс для решения задачи и Unit-тесты (согласно всем принципам).

Если в задании какие-то случаи не учтены, дополните.

- 1 **Анализ линейного уравнения** Реализуйте класс `LinearEquationSolver` с методом `static LinearResult solve(double a, double b)`, решающим уравнение  $ax + b = 0$ . Возвращаемый sealed-класс `LinearResult` имеет следующие подклассы:

- `NoSolution` — без полей; соответствует случаю  $a = 0, b \neq 0$ .
- `InfiniteSolutions` — без полей; случай  $a = 0, b = 0$ .
- `UniqueSolution` — содержит поле `double x`; случай  $a \neq 0$ , решение  $x = -b/a$ .

- 2 **Система двух линейных уравнений** Класс `LinearSystemSolver`, метод `static SystemResult solve(double a1, double b1, double c1, double a2, double b2, double c2)` для системы

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Sealed-класс `SystemResult`:

- `NoSolution` — определитель  $\Delta = a_1b_2 - a_2b_1 = 0$ , но система несовместна.
- `InfiniteSolutions` — все коэффициенты пропорциональны:  $a_1/a_2 = b_1/b_2 = c_1/c_2$ .
- `UniqueSolution` — содержит `double x, y`;  $\Delta \neq 0$ , решение по формулам Крамера.

3 **Пересечение прямых  $y = k_1x + b_1$  и  $y = k_2x + b_2$**  Класс `LineIntersectionAnalyzer`, метод `static IntersectionResult intersect(double k1, double b1, double k2, double b2)`. Sealed-класс `IntersectionResult`:

- `ParallelDistinct` —  $k_1 = k_2, b_1 \neq b_2$ .
- `Coincident` —  $k_1 = k_2, b_1 = b_2$ .
- `IntersectAtPoint` — содержит `double x, y`;  $k_1 \neq k_2$ , точка пересечения.

4 **Площадь треугольника по трём сторонам** Класс `HeronTriangle`, метод `static AreaResult computeArea(double a, double b, double c)`. Sealed-класс `AreaResult`:

- `InvalidTriangle` — содержит `String reason`; стороны  $\leq 0$  или  $a + b \leq c, a + c \leq b, b + c \leq a$ .
- `Degenerate` — содержит `double area = 0.0`; когда  $a + b = c, a + c = b$  или  $b + c = a$ .
- `ValidTriangle` — содержит `double area`; вычисляется по формуле Герона.

5 **Классификация треугольника** Класс `TriangleClassifier`, метод `static Classification classify(double a, double b, double c)`. Sealed-класс `Classification`:

- `NotTriangle` — нарушено неравенство треугольника.
- `Equilateral` — содержит `double side`;  $a = b = c$ .
- `IsoscelesRight` — содержит `double leg, double hypotenuse`; две равные стороны и выполняется  $2 \cdot leg^2 = hypotenuse^2$ .
- `Isosceles` — содержит `double equalSide, double base`; две равные стороны, но не прямоугольный.
- `ScaleneAcuteOrObtuse` — содержит `double a, double b, double c`; все стороны разные, и ни один угол не прямой.

6 **Проверка простоты числа** Класс `PrimeChecker`, метод `static Primality check(int n)`. Sealed-класс `Primality`:

- `NotNatural` — содержит `int value`;  $n \leq 1$ .
- `Prime` — содержит `int value`;  $n \geq 2$  и простое.
- `Composite` — содержит `int value, int smallestDivisor`;  $n \geq 4$  и составное.

7 **Разложение на простые множители** Класс `PrimeFactorizer`, метод `static Factorization factorize(int n)`. Sealed-класс `Factorization`:

- `InvalidInput` — содержит `int n`;  $n < 2$ .
- `PrimeNumber` — содержит `int prime`;  $n$  простое.

- `FactorizationResult` — содержит `List<Integer> factors`; список простых множителей.
- 8 **НОД и НОК двух целых чисел** Класс `GcdLcmCalculator`, метод `static GcdLcmResult compute(long a, long b)`. Sealed-класс `GcdLcmResult`:
- `BothZero` — без полей;  $a = b = 0$ .
  - `AtLeastOneZero` — содержит `long gcd`, `long lcm`; один ноль,  $\text{НОД} = |a + b|$ ,  $\text{НОК} = 0$ .
  - `BothNonZero` — содержит `long gcd`, `long lcm`; НОД и НОК вычисляются для  $|a|, |b|$ .
- 9 **Уравнение  $\sin x = a$  на отрезке  $[0, 2\pi]$**  Класс `SineEquationSolver`, метод `static SineResult solve(double a)`. Sealed-класс `SineResult`:
- `InvalidA` — содержит `double a`;  $|a| > 1$ .
  - `OneSolution` — содержит `double x`;  $a = 1 \Rightarrow x = \pi/2$ ,  $a = -1 \Rightarrow x = 3\pi/2$ .
  - `TwoSolutions` — содержит `double x1, x2`;  $|a| < 1$ ,  $x_1 = \arcsin(a)$ ,  $x_2 = \pi - \arcsin(a)$ .
- 10 **Расстояние между двумя точками** Класс `PointDistance`, метод `static DistanceResult compute(Point p1, Point p2)`. Sealed-класс `DistanceResult`:
- `IdenticalPoints` — содержит `Point p`;  $p_1 = p_2$ .
  - `PositiveDistance` — содержит `Point p1`, `Point p2`, `double distance`; расстояние  $> 0$ .
- 11 **Взаимное расположение двух окружностей** Класс `CircleRelationAnalyzer`, метод `static CircleRelation analyze(double x1, double y1, double r1, double x2, double y2, double r2)`. Sealed-класс `CircleRelation`:
- `InvalidInput` — содержит `double r1, r2`;  $r_1 \leq 0$  или  $r_2 \leq 0$ .
  - `Disjoint` —  $d > r_1 + r_2$ .
  - `TangentExternally` — содержит `Point touchPoint`;  $d = r_1 + r_2$ .
  - `TangentInternally` — содержит `Point touchPoint`;  $d = |r_1 - r_2| > 0$ .
  - `Intersecting` — содержит `Point p1, p2`;  $|r_1 - r_2| < d < r_1 + r_2$ .
  - `OneInsideOther` —  $d < |r_1 - r_2|$ .
  - `Coincident` —  $d = 0$  и  $r_1 = r_2$ .
- 12 **Факториал с проверкой переполнения** Класс `SafeFactorial`, метод `static FactorialResult compute(int n)`. Sealed-класс `FactorialResult`:
- `NegativeInput` — содержит `int n`;  $n < 0$ .
  - `Overflow` — содержит `int n`;  $n \geq 21$ .
  - `Value` — содержит `long result`;  $0 \leq n \leq 20$ .

- 13 **Анализ неравенства**  $ax^2 + bx + c > 0$  Класс `QuadraticInequalityAnalyzer`, метод `static InequalityResult analyze(double a, double b, double c)`. Sealed-класс `InequalityResult`:
- `NotQuadratic` — содержит `String message`;  $a = 0$ .
  - `AlwaysTrue` —  $a > 0$  и  $D < 0$ .
  - `AlwaysFalse` —  $a < 0$  и  $D < 0$ .
  - `TrueOutsideInterval` — содержит `double leftRoot, rightRoot`;  $a > 0, D \geq 0$ .
  - `TrueInsideInterval` — содержит `double leftRoot, rightRoot`;  $a < 0, D \geq 0$ .
- 14 **Положение точки относительно окружности** Класс `PointCirclePosition`, метод `static PositionResult check(double x0, double y0, double r, double x, double y)`. Sealed-класс `PositionResult`:
- `InvalidCircle` — содержит `double r`;  $r \leq 0$ .
  - `Inside` — содержит `Point p, double distance`; расстояние  $< r$ .
  - `OnBoundary` — содержит `Point p`; расстояние  $= r$ .
  - `Outside` — содержит `Point p, double distance`; расстояние  $> r$ .
- 15 **Уравнение**  $|x - a| = b$  Класс `AbsoluteEquation`, метод `static AbsResult solve(double a, double b)`. Sealed-класс `AbsResult`:
- `InvalidB` — содержит `double b`;  $b < 0$ .
  - `OneSolution` — содержит `double x`;  $b = 0$ .
  - `TwoSolutions` — содержит `double x1, x2`;  $b > 0$ .
- 16 **Уравнение**  $|x - a| + |x - b| = c$  Класс `SumOfAbsEquation`, метод `static SumAbsResult solve(double a, double b, double c)`. Sealed-класс `SumAbsResult`:
- `InvalidC` — содержит `double c`;  $c < 0$ .
  - `NoSolution` — содержит `double a, b, c`;  $c < |a - b|$ .
  - `InfiniteSolutions` — содержит `double left, right`;  $c = |a - b|$ .
  - `TwoSolutions` — содержит `double x1, x2`;  $c > |a - b|$ ,  $x_1 = \frac{a+b-c}{2}$ ,  $x_2 = \frac{a+b+c}{2}$ .
- 17 **День недели по дате** Класс `WeekdayFinder`, метод `static WeekdayResult find(int year, int month, int day)`. Sealed-класс `WeekdayResult`:
- `InvalidDate` — содержит `int year, month, day`; некорректная дата.
  - `Valid` — содержит `java.time.DayOfWeek weekday`; корректная дата.
- 18 **Високосный год** Класс `LeapYearChecker`, метод `static LeapResult check(int year)`. Sealed-класс `LeapResult`:
- `NonPositiveYear` — содержит `int year`;  $year \leq 0$ .
  - `LeapYear` — содержит `int year`; делится на 4, но не на 100, либо делится на 400.
  - `CommonYear` — содержит `int year`; иначе.

- 19 **Сумма арифметической прогрессии** Класс `ArithmeticProgression`, метод `static SumResult sum(int n, double first, double diff)`. Sealed-класс `SumResult`:
- `NegativeTermCount` — содержит `int n`;  $n < 0$ .
  - `ZeroTerms` —  $n = 0$ .
  - `ValidSum` — содержит `int n`, `double sum`;  $n > 0$ .
- 20 **Проверка возможности построить треугольник** Класс `TriangleFeasibility`, метод `static FeasibilityResult check(double a, double b, double c)`. Sealed-класс `FeasibilityResult`:
- `NonPositiveLength` — содержит `double a, b, c`; стороны  $\leq 0$ .
  - `CannotFormTriangle` — содержит `double a, b, c`; нарушено неравенство треугольника.
  - `CanFormTriangle` — содержит `double a, b, c`;  $a + b > c$ ,  $a + c > b$ ,  $b + c > a$ .
- 21 **Цилиндр: объём и площадь поверхности** Класс `CylinderCalculator`, метод `static CylinderResult compute(double r, double h)`. Sealed-класс `CylinderResult`:
- `InvalidInput` — содержит `double r, h`;  $r \leq 0$  или  $h \leq 0$ .
  - `Metrics` — содержит `double volume, surfaceArea`;  $V = \pi r^2 h$ ,  $S = 2\pi r(r + h)$ .
- 22 **Пересечение двух отрезков** Класс `SegmentIntersection`, метод `static IntersectionResult intersect(Point a1, Point a2, Point b1, Point b2)`. Sealed-класс `IntersectionResult`:
- `NoIntersection` — отрезки не пересекаются.
  - `PointIntersection` — содержит `Point p`; пересечение в одной точке.
  - `Overlap` — содержит `Point start, Point end`; общий отрезок.
- 23 **Параллелограмм по четырём точкам** Класс `ParallelogramChecker`, метод `static ParallelogramResult check(Point p1, Point p2, Point p3, Point p4)`. Sealed-класс `ParallelogramResult`:
- `Degenerate` — содержит `Point[] points`; точки вырождены.
  - `NotParallelogram` — содержит `Point[] points`; не параллелограмм.
  - `IsParallelogram` — содержит `Point[] points`; параллелограмм.
- 24 **Показательное уравнение  $a^x = b$**  Класс `ExponentialEquation`, метод `static ExpResult solve(double a, double b)`. Sealed-класс `ExpResult`:
- `InvalidBase` — содержит `double a`;  $a \leq 0$ .
  - `NoSolution` — содержит `double a, double b`;  $a = 1, b \neq 1$ .
  - `InvalidArgument` — содержит `double b`;  $a > 0, a \neq 1, b \leq 0$ .
  - `InfiniteSolutions` — содержит `double a, double b`;  $a = 1, b = 1$ .
  - `UniqueSolution` — содержит `double x`;  $a > 0, a \neq 1, b > 0$ .
- 25 **Анализатор сетевых адресов** Класс `NetworkAnalyzer`, метод `static AnalysisResult analyze(String address)`. Sealed-класс `AnalysisResult`:

- `ValidIPv4` — содержит массив 4 байтов
- `ValidIPv6` — содержит массив 8 чисел (по 32 бита)
- `InvalidAddress` — содержит `String error`
- `ReservedAddress` — содержит `String type` (например, `"private "loopback"`)

## 10 Семинар 9 (файлы)

В работе реализуются известные утилиты, входящие в поставки Unix/Linux (упрощенные).

1. Реализуйте утилиту `mytac`, выводящую содержимое текстового файла с конца (последняя строка, предпоследняя строка и так далее). Имя файла читается из аргументов командной строки.
2. Реализуйте утилиту `mywc`, выводящую количество слов, символов и строк в текстовом файле. Имя файла читается из аргументов командной строки.
3. Реализуйте утилиту `mytail`, выводящую последние  $n$  строк файла (или все, если строк меньше). Имя файла и  $n$  — аргументы командной строки.
4. Реализуйте утилиту `mygrep`, ищущую и выводящую строки, содержащие заданную подстроку. Имя файла и подстрока — аргументы командной строки.
5. Реализуйте утилиту `mysort`, сортирующую и выводящую строки файла в алфавитном порядке. Имя файла читается из аргументов командной строки.
6. Реализуйте утилиту `myuniq`, выводящую уникальные строки файла (удаляя последовательные дубликаты). Имя файла читается из аргументов командной строки.
7. Реализуйте утилиту `mycut`, выводящую заданный столбец из строк файла (разделитель — пробел; третий аргумент — номер столбца, начиная с 1). Имя файла и номер столбца — аргументы командной строки.
8. Реализуйте утилиту `mynl`, нумерующую и выводящую строки файла (номера слева). Имя файла читается из аргументов командной строки.
9. Реализуйте утилиту `mypaste`, объединяющую строки из двух файлов построчно (второй аргумент — второй файл). Имена двух файлов — аргументы командной строки.
10. Реализуйте утилиту `myrev`, инвертирующую и выводящую каждую строку файла (символы в обратном порядке). Имя файла читается из аргументов командной строки.
11. Реализуйте утилиту `mytr`, заменяющую символы в файле (третий аргумент — символ для замены, четвертый — на что заменить). Имя файла, символ для замены и заменяющий символ — аргументы командной строки.
12. Реализуйте утилиту `myfold`, разбивающую длинные строки на строки фиксированной длины (третий аргумент — максимальная длина строки) и выводящую результат. Имя файла и максимальная длина — аргументы командной строки.

13. Реализуйте утилиту `myod`, выводящую содержимое файла в hex-формате (по 16 байт на строку). Имя файла читается из аргументов командной строки.
14. Реализуйте утилиту `mytee`, копирующую содержимое файла в `stdout` и в другой файл (второй аргумент — файл для копии). Имена входного и выходного файлов — аргументы командной строки.
15. Реализуйте утилиту `mysed`, заменяющую первое вхождение слова в каждой строке (третий аргумент — слово для поиска, четвертый — на что заменить). Имя файла, слово для поиска и заменяющее слово — аргументы командной строки.
16. Реализуйте утилиту `mysortm`, сравнивающую два отсортированных файла и выводящую уникальные/общие строки (второй аргумент — второй файл). Имена двух файлов — аргументы командной строки.
17. Реализуйте утилиту `mydiff`, выводящую построчные различия между двумя файлами (второй аргумент — второй файл; упрощенная версия, только указание разных строк). Имена двух файлов — аргументы командной строки.
18. Реализуйте утилиту `myjoin`, объединяющую два файла по общему полю (третий аргумент — номер поля для `join`, начиная с 1; второй аргумент — второй файл). Имена двух файлов и номер поля — аргументы командной строки.
19. Реализуйте утилиту `myexpand`, заменяющую табуляции на пробелы (третий аргумент — ширина табуляции, по умолчанию 4) и выводящую результат. Имя файла и ширина табуляции — аргументы командной строки.
20. Реализуйте утилиту `myrg`, форматирующую файл для печати (добавляющую номера страниц, заголовки) и выводящую результат. Имя файла читается из аргументов командной строки.
21. Реализуйте утилиту `myshuf`, перемешивающую и выводящую строки файла в случайном порядке. Имя файла читается из аргументов командной строки.
22. Реализуйте утилиту `mycksum`, вычисляющую простой контрольный код файла: сумму всех байтов по модулю  $2^{32}$  и размер файла в байтах. Результат выводится в формате `checksum size filename`. Имя файла читается из аргументов командной строки.
23. Реализуйте утилиту `myrot13`, заменяющую все латинские буквы в файле по алгоритму ROT13 и выводящую результат. Имя файла читается из аргументов командной строки.

## 11 Семинар 10 (многопоточность)

Реализуйте потокобезопасную систему банковских переводов. Создайте класс `BankAccount` с полями `id` (уникальный идентификатор) и `balance` (баланс). Метод `boolean transfer(BankAccount to, int amount)` должен выполнять атомарный перевод средств между счетами с учётом ограничений варианта.

### Обязательные требования:

- Гарантированная потокобезопасность: отсутствие гонок данных при одновременной работе множества потоков

- Отсутствие дедлоков при любых последовательностях переводов
- Сохранение глобального инварианта: сумма балансов всех счетов остаётся постоянной
- Корректная обработка прерываний потоков (`InterruptedException`)
- Минимизация времени удержания блокировок для повышения пропускной способности
- Предоставление метода `getTotalBalance()` для проверки инварианта

**Тестовая сценария:** В методе `main` создайте 5 счетов с начальным балансом 1000. Запустите 10 потоков, каждый из которых выполняет 100 случайных переводов между случайными счетами в диапазоне сумм от 1 до 100. После завершения всех потоков проверьте выполнение глобального инварианта.

1. Перевод разрешён только если после операции баланс обоих счетов остаётся не менее 100 единиц.
2. Перевод разрешён только если после операции разница между максимальным и минимальным балансом не превышает 1500 единиц.
3. Перевод разрешён только если сумма перевода не превышает 30% от текущего баланса отправителя.
4. Перевод разрешён только если получатель имеет минимальный баланс среди всех счетов на момент начала операции.
5. Перевод разрешён только если отправитель имеет максимальный баланс среди всех счетов на момент начала операции.
6. Перевод разрешён только если после операции баланс получателя не превышает 2000 единиц.
7. Перевод разрешён только если сумма перевода кратна номеру отправителя (`amount % fromId == 0`).
8. Перевод разрешён только если после операции баланс отправителя остаётся чётным числом.
9. Перевод разрешён только если сумма перевода не превышает средний баланс всех счетов на момент начала операции.
10. Перевод разрешён только если номера отправителя и получателя имеют одинаковую чётность (`fromId % 2 == toId % 2`).
11. Перевод разрешён только если после операции разница балансов отправителя и получателя кратна 25.
12. Перевод разрешён только если отправитель и получатель не являются соседними по идентификатору (`|fromId - toId| ≠ 1`).
13. Перевод разрешён только если баланс получателя до перевода меньше 600 единиц.

14. Перевод разрешён только если баланс отправителя до перевода превышает 1200 единиц.
15. Перевод разрешён только если после операции оба баланса остаются в диапазоне [500; 2000].
16. Перевод разрешён только если сумма перевода является степенью двойки (1, 2, 4, 8, 16, 32, 64).
17. Перевод разрешён только если после операции балансы отправителя и получателя имеют разную чётность.
18. Перевод разрешён только если сумма перевода не превышает 10% от баланса получателя до перевода.
19. Перевод разрешён только если отправитель и получатель не являются крайними счетами (идентификаторы не 0 и не 4).
20. Перевод разрешён только если после операции сумма балансов крайних счетов (идентификаторы 0 и 4) составляет не менее 1800 единиц.
21. Перевод разрешён только если сумма перевода является палиндромом в десятичной записи (11, 22, 33, 44, 55, 66, 77, 88, 99).
22. Перевод разрешён только если баланс получателя до перевода меньше баланса отправителя.
23. Перевод разрешён только если после операции все балансы остаются в диапазоне [400; 2200].
24. Перевод разрешён только если сумма перевода делится без остатка на количество активных потоков в момент начала операции.
25. Перевод разрешён только если после операции медианный баланс всех счетов не изменился более чем на 50 единиц.

## 12 Семинар 11 (Spring Boot, REST API)

Разработайте с использованием Spring boot REST API сервис.

Сервис должен предоставлять весь CRUD сервис (Create, Read, Update, Delete), то есть достаточно реализовать 4 запроса, так же он должен валидировать передаваемые данные и сообщать об ошибках в достаточно удобном виде (посредством статусом и описания ошибок для каждого поля).

1. База данных студентов группы. Поля: фамилия, имя, отчество, пол, возраст.
2. База данных расходов семьи. Поля: товар, стоимость, количество, дата.
3. База данных загрузки аудиторий. Поля: дата и время, начала, дата и время конца, аудитория, преподаватель.
4. База данных учета доходов и расходов предпринимателя. Поля: дата, тип операции (доход/расход), объем операции, описание, корреспондент.

5. База данных велоклуба. Поля: ФИО, тип велосипеда (МТВ и др.), стаж участия в велоклубе.
6. База данных рейсов авиакомпании. Поля: дата и время вылета, аэропорт вылета, аэропорт прилета, дата и время прилета, марка самолета.
7. База данных автобусных маршрутов. Поля: номер маршрута, номер парка, времена начала и окончания движения, длина маршрута в км.
8. База данных электричек. Поля: вокзал, номер поезда, количество вагонов, тип (экспресс/обычный/спутник).
9. База данных товаров Интернет-магазина. Поля: название товара, категория, цена товара, описание товара.
10. База заказов Интернет-магазина. Поля: ФИО заказчика, стоимость заказа, скидка (в процентах), адрес доставки.
11. База данных выборов. Поля: участок, кандидат, количество голосов.
12. База данных практических работ. Поля: практическая работа, студент, номер варианта, номер уровня, дата сдачи, оценка.
13. База данных операторов и телеканалов. Поля: Название, тип (спутник, кабель, Интернет), охват (кол-во миллионов домохозяйств), минимальная стоимость подписки.
14. База данных тарифных планов оператора. Поля: название, тип вещания (обычный/HD), флаг общедоступности.
15. База данных незаконно огороженных берегов. Поля: водный объект, регион, GPS-координаты, длина недоступного участка берега, дата фиксации нарушения.
16. База данных временного прекращения движения в метро. Поля: дата и время начала прекращения движения, дата и время окончания прекращения движения, станция, станция (от какой до какой станции прекращено движение).
17. База данных проката фильмов. Поля: дата, время, кинотеатр, фильм, номер зала, тип сеанса (3D/Imax/обычный).
18. База данных эвакуированных автомобилей. Поля: улица, автостоянка, GPS-координаты, тип нарушения (стоянка на проезжей части в месте запрета, стоянка на тротуаре, стоянка на газоне), номер автомобиля, тип автомобиля (легковой/грузовой малой тонажности/-грузовой большой тонажности).
19. База данных средних специальных учебных учреждений. Поля: название, адрес, тип подчинения (федеральный/региональный), год основания, номер лицензии, номер аккредитации, дата окончания действия аккредитации.
20. База данных поселков. Поля: название, девелопер, площадь, количество жителей.
21. База данных сухопутной военной техники. Поля: название, модель, разработчик, предприятие, стоимость, тип.

22. База данных деревьев в городе. Поля: GPS-координаты, вид дерева, округ, год посадки.
23. База данных футбольных матчей. Поля: дата, команда, команда, счет, место проведения.
24. База данных обращений жителей. Поля: дата, время, объект, заявитель, содержание обращения (до 255 символов), дата ответа, ответ на обращение (до 255 символов).
25. База данных студентов колледжа. Поля: ФИО, группа, признак бюджетности, стипендия (нет/обычная/повышенная), флаг наличия социальной стипендии, дата рождения.

### 13 Семинар 12 (JWT server)

Реализуйте функционал регистрации, аутентификации и авторизации на REST API сервисе, реализованном на предыдущем семинаре.

В итоге каждый пользователь должен иметь доступ только к своим объектам в базе данных, работа сессий должна поддерживаться посредством JWT.

### 14 Семинар 13 (микросервисы)

«Распилите» ваше приложение на два микросервиса: регистрации и аутентификации; собственно основной работы. Каждый микросервис должен работать (естественно) со своей базой данных – postgresql, развернутой в другом контейнере.

Замечание: задача носит учебный характер (отрабатываются технические навыки и логика взаимодействия микросервисов на простейшем примере) и реализуется в рамках небольшого времени, отведенного на работу. Архитектурные вопросы выделения микросервисов мы не рассматриваем.

### 15 Семинар 14 (kafka)

Реализуйте разбиение микросервиса основной работы на два: приема запросов от клиентов с передачей событий в kafka/чтения готовой информации из базы данных; сервиса обработки событий с записью их в ту же базу данных; дополнительно реализуйте микросервис, формирующий файл с информацией обо всех операциях

Замечание: задача носит учебный характер (отрабатываются технические навыки и логика взаимодействия микросервисов) и реализуется в рамках небольшого времени, отведенного на работу. Архитектурные вопросы выделения микросервисов мы не рассматриваем.

### 16 Семинар 15

Проверочная работа по материалам предыдущих семинаров.

### 17 Семинар 16

В ходе семинара предполагается прием и защита работ, закрытие «долгов».