Общие требования ко всем вариантам

1. Реализовать решение на **Java**. Использовать `java.util.regex.Pattern/Matcher`. Работа со списками — `java.util.List`, `ArrayList`, `Stream/Collectors` (по желанию).

2. Каждый вариант должен использовать **несколько** приёмов с регулярными выражениями (например: разбор, валидация, замена, извлечение с группами, позитив/негатив lookaround, ленивые/жадные квантификаторы, named groups, non-capturing groups).

3. Обязательные операции со списками: фильтрация, трансформация (map), агрегация (grouping/сводка), сортировка, удаление дубликатов, слияние/разбиение списков.

Вариант 1 — «Нормализация международных телефонных номеров»

Описание: Есть текстовый файл с телефонными номерами в различных форматах (локальные, с пробелами, с символами `()`, с международным префиксом +, с кодом страны в скобках, с дефисами, с префиксом 8/0 и т.д.). Нужно извлечь, валидировать, нормализовать (в формат +<CC><NNN...>), сгруппировать по стране, удалить дубликаты и вывести частотную статистику.
Что сделать (пошагово):

1. Прочитать файл/строки, собрать все потенциальные номера.

2. С помощью регексов выделить валидные номера (поддержать разные форматы).

3. Преобразовать каждый номер в единый формат `+<country_code><subscriber_number>` (включая доп. правила для вариантов с 8 вместо +7, и т.п.).

4. Удалить дубликаты, отсортировать по частоте встречаемости.

5. Вывести: список уникальных нормализованных номеров + частота, и топ-5 стран по количеству номеров.
   Формат ввода/вывода: вход — текстовый файл; выход — таблица `номер, частота` + блок `страна, count`.
   Обязательные regex-приёмы: извлечение с группами, lookbehind для отделения кода, non-capturing группы для допустимых разделителей, ленивые квантификаторы.

Пример:
Вход (фрагмент):

```
8 (912) 345-67-89
+7 912 345 67 89
+1-202-555-0173
(202)5550173
0044 20 7946 0958
```

Выход (фрагмент):

```
+79123456789, 2
+12025550173, 2
+442079460958, 1

Countries:
RU, 2
US, 2
GB, 1
```

Вариант 2 — «Разбор и агрегация web-логов»

Описание: Даны строковые access-логи (формат похож на Common Log Format / Combined). Нужно выделить IP, дату/время, HTTP-метод, путь, статус, размер, user-agent; посчитать уникальных посетителей за час, 404-ошибки по URL, топ-10 запрашиваемых ресурсов.
Что сделать:

1. Разобрать строки логов с помощью regex, распарсить поля.

2. Перевести временные метки в `LocalDateTime` (учесть часовой пояс).

3. Сгруппировать записи по часу и посчитать уникальные IP (set).

4. Подсчитать частоты статусов (особенно 404) и вывести топ-10 URL по количеству запросов и топ-10 URL с наибольшим числом 404.
   Формат: вход — файл логов; выход — отчёт (таблицы).
   Обязательные regex-приёмы: жадность/ленивость для корректного извлечения `"user-agent"`, named groups для удобства, группировка для time-parse.
   Пример входа (1 строка):
   `127.0.0.1 - - [10/Oct/2024:13:55:36 +0300] "GET /index.html HTTP/1.1" 200 1024 "http://ref" "Mozilla/5.0..."`

Вывод (фрагмент):

```
Hour 2024-10-10 13: unique_ips=123
Top resources: /index.html (345), /product/12 (234), ...
Top 404: /missing.png (45), ...
```

Подсказки: использовать `Pattern.DOTALL`/`DOTALL` для multi-line, аккуратно обрабатывать отсутствующие поля.

Вариант 3 — «Парсер и нормализатор дат в свободном тексте»

Описание: В тексте встречаются даты в разных форматах: `dd.MM.yyyy`, `d MMM yyyy`, `yyyy/MM/dd`, `Oct 12, 2024`, `12th Oct 2024` и т.п. Задача — найти все даты, валидировать (учесть високосные), нормализовать в ISO `yyyy-MM-dd`, убрать дубликаты, отсортировать.
Что сделать:

1. С помощью набора regex-шаблонов найти все возможные даты.

2. Для найденных дат определить формат, распарсить, проверить валидность (например, 30 февраля — отклонить).

3. Нормализовать и вывести отсортированный список уникальных дат.
   Формат: вход — текст; выход — список `yyyy-MM-dd`.
   Обязательные приёмы: несколько шаблонов, использование `lookahead`/`lookbehind` для отделения суффиксов (`st`, `nd`, `th`), named groups для дня/месяца/года.
   Пример: вход: `"Event on 12th Oct 2024, rescheduled from 02.03.2022 and 2022/03/02."` → вывод: `2022-03-02`, `2024-10-12`.

Вариант 4 — «Категоризация транзакций в банковской выписке»

Описание: Текст вывода банковской выписки содержит описания транзакций (разные языки и сокращения). Нужно извлечь операции: дата, сумма (разные валюты), контрагент, назначение; с помощью правил/регексов отнести их к категориям (питание, транспорт, зарплата, интернет и т.д.), посчитать суммарные расходы/доходы по категориям.
Что сделать:

1. С помощью regEx извлечь строки транзакций, затем извлечь группы: дата, сумма (со знаком), валюта, описание.

2. Составить набор шаблонов для категоризации: regex-паттерны для ключевых слов (cafe|coffee → питание, taxi|uber → транспорт и т.д.). Категоризация должна поддерживать несколько совпадений (multi-label).

3. Подсчитать суммы по категориям (в одной базовой валюте — для простоты: использовать указанный курс из файла).
   Формат: вход — файл выписки + файл курсов; выход — CSV: `category, sum`.
   Обязательные regex-приёмы: извлечение чисел с разделителями, групповые шаблоны, case-insensitive, использование `\p{Sc}` (символ валюты) либо буквенных кодов.
   Пример:
   Вход: `12.10.2024; -1,234.56 RUB; CAFE "StarCoffee"` → Категория `Питание: -1234.56`.
   Подсказки: предусмотреть отрицательные/положительные суммы; использовать `BigDecimal` для сумм.

Вариант 5 — «Дедупликация и нормализация email-адресов»

Описание: Имеется большой список email-адресов (разные регистры, пробелы, опечатки, адреса с точками в локальной части, плюсовыми тегами `user+tag@domain`). Задача — собрать уникальные рабочие адреса по ряду правил (например, в Gmail: точки и `+tag` игнорировать; в других доменах — нет), сгруппировать по доменам, вывести статистику.
Что сделать:

1. Регексом извлечь все потенциальные email-адреса.

2. Нормализовать: trim, toLowerCase; применить domain-specific правила (gmail/googlemail).

3. Удалить некорректные (невалидные) адреса по RFC-подобному шаблону (упростить, но строго).

4. Сгруппировать по домену и вывести топ-5 доменов.
   Обязательные приёмы: сложный шаблон email (с группами), lookahead/lookbehind для `+tag` удаления, non-capturing группы.
   Пример: вход: `User.Name+test@GMAIL.com` → нормализация: `username@gmail.com`.
   Подсказки: реализовать правило, где для домена `gmail.com` и `googlemail.com` локальная часть нормализуется (точки и `+tag` убираются).

Вариант 6 — «Извлечение и исправление SKU/артикулов из каталога»

Описание: В текстах/описаниях встречаются артикулы в разных форматах (например: `SKU: AB-1234`, `арт. AB1234`, `AB_1234/56`). Нужно извлечь артикулы, нормализовать формат `ABC-1234` (правила: буквы верхний регистр, дефис между буквенной и цифровой частью, убрать постфиксы), подсчитать повторяющиеся артикула, собрать список «похожих» артикулов (по расстоянию Левенштейна — бонус).
Что сделать:

1. Регексами найти потенциальные SKU.

2. Нормализовать: выделить буквенную часть и числовую, привести к стандарту.

3. Удалить лишние суффиксы (например, `/56`), объединить варианты.

4. Сгруппировать и посчитать частоты.
   Формат: вход — текст; выход — `SKU, count, source_lines`.
   Обязательные приёмы: сложный шаблон для смешанных форм, использование non-capturing groups, альтернативы `|`.
   Пример: вход: `Артикул: AB_1234/56` → `AB-1234`.
   Подсказки: предусмотреть транслит/русские буквы (если встречаются), использовать `replaceAll` для нормализации.

Вариант 7 — «Очистка и извлечение ссылок из HTML/Markdown»

Описание: Даны страницы в Markdown/частично в HTML. Нужно извлечь все ссылки (включая изображения), аггрегировать по доменам, удалить внешние трекеры (URL с параметрами `utm_*`), нормализовать query-params (сортировка параметров), вывести список уникальных доменов и топ-10 внешних ссылок.
Что сделать:

1. Регексами найти ссылки в форматах Markdown `[text](url)` и HTML `<a href="...">`.

2. Извлечь домен (регекс) и параметры query.

3. Удалить UTM-параметры и сортировать оставшиеся параметры (для нормализации).

4. Удалить дубликаты и посчитать частоты.
   Обязательные приёмы: многошаговый парсинг ссылок, использование lookaround для безопасного извлечения `href`, замены `replaceAll` с группами.
   Пример: вход:
   `[Статья](https://example.com/page?utm_source=fb&a=1)` → нормализованный `https://example.com/page?a=1`.
   Подсказки: осторожно с кавычками/апострофами; на HTML блоки могут быть многострочными.

Вариант 8 — «Разбор химических формул (подсчёт атомов)»

Описание: В тексте встречаются химические формулы (H2O, C6H12O6, Fe(OH)3, (NH4)2SO4). Задача — извлечь формулы, распарсить их (учитывая скобочные множители), вернуть карту элемент→количество для каждой формулы, затем сгруппировать формулы с одинаковым составом (изомеры по составу).
Что сделать:

1. Регексами найти формулы (последовательности букв+чисел+скобок).

2. Для каждой формулы с помощью регекс-токенизации разбить на токены (`Element`, `Number`, `(`, `)`), затем вычислить итоговые множители (стек).

3. Сформировать `Map<String, Integer>` элементов и вывести отсортированно.

4. Группировать формулы с равным составом.
   Обязательные приёмы: составной разбор с несколькими regex-шаблонами, использование рекурсивного подхода (стек), извлечение групп `([A-Z][a-z]?)(\d*)`.
   Пример: `Fe(OH)3` → `Fe:1, O:3, H:3`.

Вариант 9 — «Анализ логов электронной почты (headers)»

Описание: Даны raw-email заголовки (From, To, Subject, Date, Message-ID, Received и т.д.). Нужно извлечь цепочку Received-хопов (IP адреса и временные метки), выделить отправителя и получателей, отфильтровать спам-подобные сообщения (по набору regex-правил), сгруппировать письма по доменам отправителей и посчитать среднюю длину subject.
Что сделать:

1. Регексами извлечь поля заголовков (мультистрочные поля).

2. Для `Received` извлечь IP и временную метку (и нормализовать).

3. Спам-фильтр: набор правил (подозрительные ключевые слова, много доменов в Reply-To и т.п.).

4. Вывести статистику: `domain → count`, средняя длина subject.
   Обязательные приёмы: multi-line matching, DOTALL, lookahead/behind для многострочных заголовков, named groups.
   Пример: `Received: from mail.example.com (mail.example.com [192.0.2.1]) by mx.google.com with ESMTPS id ...; Mon, 10 Oct 2024 13:55:36 +0300` → IP `192.0.2.1`.
   Подсказки: аккуратный парсинг многострочных заголовков (сложные folding).

Вариант 10 — «Парсинг структурированных конфигураций в свободном тексте»

Описание: В текстах встречаются конфигурационные блоки в формате `key: value`, `key = value`, JSON-like (`{key: value, ...}`), INI-like. Нужно извлечь все пары ключ-значение, нормализовать ключи (lowercase, replace spaces → _), объединить пары с одинаковым ключом (правила слияния: последнее значение имеет приоритет), и вывести итоговую конфигурацию в YAML-подобном виде.
Что сделать:

1. Найти все пары разных форматов via regex.

2. Нормализовать ключи и значения (trim, remove кавычки).

3. Выполнить merge с приоритетом последнего упоминания.

4. Вывести итог.
   Обязательные приёмы: разные шаблоны для `key:value`, `key = "value"`, key в кавычках; grouping и последовательная фильтрация.
   Пример: вход: `timeout: 30` и позже `Timeout = 60` → итог `timeout: 60`.
   Подсказки: учитывать многострочные значения (heredoc-подобные) — можно ограничить задачу, но указать на них.

Вариант 11 — «Токенизация математических выражений и перевод в ОПЗ (RPN)»

Описание: Пользователь вводит строки с математическими выражениями (с числами, переменными, функциями `sin`, скобками, унарным минус). Нужно токенизировать выражение с помощью regex (числа/идентификаторы/операторы/скобки), затем реализовать shunting-yard, получить RPN и вычислить значение при заданных

значениях переменных.
Что сделать:

1. Regex-токенизация (учесть вещественные числа, экспоненциальную форму, идентификаторы).

2. Реализовать shunting-yard (стек) с приоритетами и унарными операторами.

3. Преобразование в RPN и вычисление по входному map переменных.
   Формат: вход — строка выражения; вывод — RPN и вычисленный результат.
   Обязательные приёмы: сложные шаблоны токенизации, lookbehind для отличия унарного и бинарного минуса, использование списков для стека.
   Пример: `-3 + 4* (x - 2)` при x=5 → RPN `3 unary- 4 x 2 - * +` → result 9.
   Подсказки: аккуратно различать - унарный/бинарный с помощью предшествующего токена (regex + логика).

Вариант 12 — «Разбор сложных CSV с вложенными кавычками и перевод в JSON»

Описание: Файл CSV содержит поля с запятыми, кавычками, многострочные поля. Нужно корректно выделить записи с помощью regex, преобразовать в объекты и вывести JSON-массив (без внешних библиотек). Также — удалить дубликаты строк (по комбинации ключевых полей).
Что сделать:

1. Реализовать парсер строк CSV с поддержкой "escaped quotes" и многострочных полей (regex-шаблон с учётом кавычек).

2. Преобразовать записи в JSON строки (cescaping).

3. Удалить дубликаты, отсортировать по ключу.
   Формат: вход — CSV; выход — JSON array.
   Обязательные приёмы: regex для CSV токенизации (сложный шаблон с alternation), replaceAll для unescape.
   Пример:
   `"id","desc"\n1,"Line with, comma"\n2,"Quote ""inside"""` → JSON.
   Подсказки: можно реализовать regex для токена: `"(?:[^"]|"""")*"|[^,]+`.

Вариант 13 — «Выделение и нормализация юридических ссылок (цитат законов)»

Описание: В текстах встречаются ссылки на статьи законов (на русском): `ст. 123 УК РФ`, `статья 45 ГК РФ`, `§ 12`, `Article 5 of Law 200/12`. Нужно извлечь все правовые ссылки, нормализовать формат `LAW_TYPE LAW_NUMBER:ARTICLE` и сгруппировать по закону.
Что сделать:

1. Регексами найти различные формы юридических ссылок (рус/англ).

2. Нормализовать в единый вид.

3. Составить сводку: какой документ и какие статьи чаще упоминаются.
   Формат: вход — текст; выход — таблица `doc → [articles]`.
   Обязательные приёмы: многоальтернативный шаблон, case-insensitive, использование lookbehind для отделения контекста.
   Пример: `ст. 123 УК РФ` → `UK_RF:123`.
   Подсказки: учесть сокращения (УК, ГК, ФЗ) и латинские варианты.

Вариант 14 — «Извлечение сущностей «деньги/валюта» из текста»

Описание: Тексты содержат денежные суммы в разных форматах (`$1,234.56`, `1 234,56 ₽`, `EUR 1234`, `€1.234,56`). Задача: найти все суммы, распознать валюту и числовое значение, конвертировать – агрегировать суммы по валютам, найти предложения с наибольшими суммами.
Что сделать:

1. Regex-шаблонами найти суммы с символом валюты до/после числа либо кодом валюты.

2. Нормализовать число (разделители тысяч/десятичных).

3. Собрать агрегаты по валютам и вывести топ-5 предложений по суммам.
   Формат: вход — текст; выход — `currency -> total`.
   Обязательные приёмы: сложные шаблоны для чисел с разными локалями, группы для валюты/числа.
   Пример: `Paid €1.234,56 today and $1,000 yesterday` → `EUR: 1234.56; USD: 1000.00`.
   Подсказки: использовать `BigDecimal` и нормализатор разделителей.

Вариант 15 — «Извлечение хэштегов и упоминаний из чата и статистика по пользователям»

Описание: Есть экспорт чата (строки с временными метками и сообщениями). Нужно извлечь упоминания (`@user`), хэштеги `#topic`, агрегировать список уникальных упоминаний, подсчитать активность пользователей (сколько сообщений, сколько упоминаний), и вывести топ-3 самых упоминаемых тем.
Что сделать:

1. Регексами извлечь timestamp, username, message.

2. Внутри message найти @ и # элементы (учесть границы слов, знаки пунктуации).

3. На основании списка сообщений посчитать активность и частоты.
   Формат: вход — chat export; выход — таблицы user activity, top hashtags.
   Обязательные приёмы: word boundary `\b`, negative lookbehind (чтобы не лочить email).
   Пример: `12:01 Anna: Привет @Ivan! #home` → mentions `Ivan`, hashtag `home`.
   Подсказки: отличать @ в email от упоминания (lookbehind).

Вариант 16 — «Анализ исходного кода: извлечение сигнатур методов и TODOs»

Описание: Дана папка с Java-файлами. Нужно: найти все определения методов (с модификаторами, возвращаемым типом, именем, параметрами), посчитать число перегруженных методов, извлечь все комментарии `TODO:` и `FIXME:`, сгруппировать TODO по файлам и вывести сводку.
Что сделать:

1. Регексами пройти файлы и найти сигнатуры методов (учесть Generic-типы и аннотации).

2. Сформировать список методов (`class.method(signature)`), найти перегрузки (по имени).

3. Собрать TODO/FIXME из комментариев (однострочных и многострочных).
   Формат: путь → [method signatures], TODO list.
   Обязательные приёмы: сложные шаблоны для Java-сигнатур (арифметика с generics), DOTALL для многострочных комментариев, non-capturing groups.
   Пример: `public <T> List<T> findAll(Class<T> cls) throws Exception {` → захват `findAll`.
   Подсказки: учитывать аннотации `@Override` и модификаторы.

Вариант 17 — «Выделение и нормализация товарных описаний (NLP-подход, regex-подложка)»

Описание: Каталог содержит неструктурированные товарные описания: размеры, цвета, состав (хлопок 80%), размеры в разных нотациях (`S`, `M`, `L`, `42`, `42-44`), веса. Нужно извлечь структурированные атрибуты, нормализовать (размер → единый формат, состав → список), затем собрать список вариантов каждого товара (вариации по цвету/размеру).

Что сделать:

1. Regex-выражениями найти паттерны размеров, цветов, материалов, веса.

2. Нормализовать названия (рус/англ), собрать вариации.

3. Вывести для каждого `product_id` список вариантов.
   Формат: вход — CSV с `product_id,description`; вывод — JSON с variants.
   Обязательные приёмы: альтернативы, case-insensitive, named groups, replaceAll.
   Пример: `Размер: 42-44, Цвет: красный/Red, Состав: 80% хлопок` → `sizes:[42,43,44], color:red, materials:[cotton:80%]`.
   Подсказки: предусмотреть диапазоны размеров.

Вариант 18 — «Парсинг и нормализация URL API-запросов (query param analytics)»

Описание: Журнал API-запросов содержит URL с query-параметрами. Нужно извлечь все параметры, нормализовать порядок параметров в URL, выделить редкие параметры (встречающиеся в <1% случаев), сгруппировать по endpoint (path) и вывести статистику параметров по каждому path.

Что сделать:

1. Регексом отделить path и query.

2. Разобрать параметры `key=value`, нормализовать порядок (alphabetic).

3. Подсчитать частоты параметров и пометить редкие.
   Формат: вход — список URL; вывод — `path → {param:count}`.
   Обязательные приёмы: шаблон `([^?]+)\?(.*)`, парсинг пар `([^&=]+)=([^&]*)`.
   Пример: `/api/search?q=test&page=2&sort=asc` → normalized `/api/search?page=2&q=test&sort=asc`.
   Подсказки: URL-decode значений (по желанию).

Вариант 19 — «Обработка текстов медицинских назначений»

Описание: В медицинских записях встречаются назначения: `Амоксиклав 500 мг 1 таб 2 раза в день 7 дней`, `Paracetamol 1g PRN q4-6h`. Нужно извлечь лекарство, дозировку, форму, режим (частоту) и срок, нормализовать в структуру (drug, dose_mg, form, frequency_per_day, duration_days).
Что сделать:

1. Regex-шаблонами найти записи назначений.

2. Распознать дозировки (мг, г), единицы, формы (таб, капс, сусп), частоту (2 раза в день, q6h).

3. Нормализовать частоту в `times_per_day` (приблизительно) и длительность.
   Формат: вход — список назначений; выход — JSON объектов.
   Обязательные приёмы: шаблоны для доз и временных выражений, lookbehind для единиц.
   Пример: `Амоксиклав 500 мг 1 таб 2 раза в день 7 дней` → `{drug: "Амоксиклав", dose_mg:500, form:"таб", freq:2, duration:7}`.
   Подсказки: частоту в форме qXh перевести в `24/X`.

Вариант 20 — «Поиск и нормализация товарных ценников с акциями»

Описание: В маркетплейс-логе встречаются строки с ценой и промо: `Цена: 1 299,00 ₽ (скидка 15% − 1 104,15 ₽)`, `OldPrice: $29.99 -> New: $19.99`. Нужно извлечь обычную цену, цену со скидкой (если есть), и вычислить правильную скидку/проверить указанную скидку, пометить неконсистентные записи.
Что сделать:

1. Regex-шаблонами найти ценовые блоки, извлечь original/discount/after.

2. Нормализовать числа и валюту.

3. Вычислить реальную скидку и сравнить с указанной (если есть) с tolerance 0.5%.

4. Вывести список неконсистентных ценников и статистику по валютам.
   Формат: вход — текст; выход — список `item_id, original, discounted, claimed_discount, computed_discount, consistent`.
   Обязательные приёмы: шаблоны для чисел и процентов, группы для валют/чисел, replaceAll.
   Пример: `Цена 1299 -> 1104.15 (скидка 15%)` → computed 15.0% →

consistent=true.

Подсказки: учесть округления (до 2 знаков) и локализованные разделители.