Сборник заданий для семинарских занятий
по курсу
«Разработка мобильных приложений»

Содержание

1 Общие сведения

Сборник содержит задания для семинарских занятий по курсу «Разработка мобильных приложений». Задачник рассчитан на 48 часов семинарских занятий.

Структурно курс делится на две части:

1. изучение языка Kotlin;
2. изучение разработки на платформе Android.

В практических и лабораторных работах предполагается использование языка Kotlin в средах IntelliJ IDEA и Android Studio. При этом необходимо соблюдать Coding Conventions (требования к стилю кода).

Перед сдачей работы добейтесь, чтобы среда не выдавала предупреждений при запуске подпункта **Inspect Code** пункта меню **Analyze**.

Задачи в большей степени рассчитаны на освоение возможностей языка, а не на алгоритмические сложности, поэтому осуществляйте написание кода в соответствии с заданием, а не с целью, чтобы он просто работал.

Обратите внимание на то, что во всех заданиях необходимо проверять корректность входных данных: программа не должна «падать» ни в каких ситуациях.

2 Язык Kotlin

2.1 Семинар «Разработка первой программы на Kotlin» (2 часа)

Цель первых четырёх семинаров — обеспечить подготовку к выполнению заданий, нацеленных на освоение особенностей языка Kotlin. В первой работе необходимо написать небольшую программу, чтобы убедиться в понимании базовых конструкций языка (функция main, ветвления, циклы) и целого типа, а также обеспечить успешную настройку среды разработки.

При выполнении заданий обращайте внимание на использование специфических особенностей языка везде, где это возможно: if и when могут быть как операторами, так и частью выражений; фигурные скобки во многих случаях можно опускать; точки с запятой почти никогда не используются, корректно выбирайте, как помечать переменные: ключевым словом var или val.

При выполнении работ обеспечивайте **оптимальность** предлагаемой программы как по скорости, так и по памяти. В случае противоречия между двумя критериями, выбирайте алгоритм, который обеспечивает лучшее быстродействие.

В частности, это обозначает, что:

1. Ввод осуществляется как строка, которую необходимо преобразовать в целое число.
2. Далее вся работа должна вестись **исключительно с числом**, используя только арифметические операции (остаток от деления, целочисленное деление, циклы, условия).
3. Использование строковых методов (включая преобразование числа обратно в строку) **запрещено**.
4. Следует избегать сложных структур данных (списков, множеств, массивов).

Задание

Для данного неотрицательного целого числа (в пределах Int) найдите указанный результат. Осуществите проверку корректности ввода. Оформите программу и устраните все warning.

1. Сумму чётных цифр
2. Сумму нечётных цифр
3. Произведение чётных цифр
4. Произведение нечётных цифр
5. Максимальную чётную цифру
6. Минимальную чётную цифру
7. Максимальную нечётную цифру
8. Минимальную нечётную цифру
9. Сумму цифр, кратных трём
10. Сумму цифр, не кратных трём
11. Произведение цифр, кратных трём
12. Произведение цифр, не кратных трём
13. Максимальную цифру, кратную трём
14. Минимальную цифру, кратную трём
15. Максимальную цифру, не кратную трём
16. Минимальную цифру, не кратную трём
17. Сумму цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 4
18. Произведение цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 3

19. Максимальную цифру среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 3

20. Минимальную цифру среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 1

21. Сумму цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2

22. Произведение цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2

23. Максимальную цифру среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2

24. Минимальную цифру среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2

25. Максимальную цифру среди стоящих на позициях в числе, номера которых кратны четырём (если нумеровать цифры с конца): для числа 1234 — ответ 1

2.2 Практическая работа «Функции Kotlin» (2 часа)

Задание №1 (функции и лямбда-функции)

Разработайте программу в соответствии с вашим вариантом. При этом:

- Основной алгоритм (без ввода-вывода) оформите в виде функции.
- В функцию передавайте исходное число и условие отбора цифр в виде лямбда-функции (например, проверка на чётность, нечётность, кратность трём, не кратность трём — в зависимости от варианта).
- Функция должна корректно работать при замене условия отбора — то есть быть универсальной.

1. Сумму чётных цифр
2. Сумму нечётных цифр
3. Произведение чётных цифр
4. Произведение нечётных цифр
5. Максимальную чётную цифру
6. Минимальную чётную цифру
7. Максимальную нечётную цифру
8. Минимальную нечётную цифру
9. Сумму цифр, кратных трём
10. Сумму цифр, не кратных трём
11. Произведение цифр, кратных трём
12. Произведение цифр, не кратных трём
13. Максимальную цифру, кратную трём
14. Минимальную цифру, кратную трём
15. Максимальную цифру, не кратную трём
16. Минимальную цифру, не кратную трём
17. Сумму цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 4
18. Произведение цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 3
19. Максимальную цифру среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 3
20. Минимальную цифру среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 — ответ 1
21. Сумму цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2
22. Произведение цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2
23. Максимальную цифру среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2
24. Минимальную цифру среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 — ответ 2

25. Максимальную цифру среди стоящих на позициях в числе, номера которых кратны четырём (если нумеровать цифры с конца): для числа 1234 — ответ 1

Задание №2 (функции высшего порядка)

1. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию — сумму исходных. Для аргумента x результирующая функция возвращает сумму $f_1(x) + f_2(x) + \cdots + f_n(x)$.

2. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию — произведение исходных. Для аргумента x результирующая функция возвращает произведение $f_1(x) \cdot f_2(x) \cdot \cdots \cdot f_n(x)$.

3. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию — максимум исходных. Для аргумента x результирующая функция возвращает $\max(f_1(x), f_2(x), \ldots, f_n(x))$.

4. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию — минимум исходных. Для аргумента x результирующая функция возвращает $\min(f_1(x), f_2(x), \ldots, f_n(x))$.

5. Создайте функцию, которая по данной функции f : `Int` $\to$ `Int` и целому числу n возвращает функцию, вычисляющую f, применённую n раз: $f(f(f(\ldots f(x)\ldots)))$.

6. Создайте функцию, которая принимает произвольное количество функций без параметров, возвращающих `String`, и возвращает новую функцию без параметров, возвращающую конкатенацию результатов исходных функций.

7. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию с аргументом x типа `Int`, возвращающую **индекс первой** функции (начиная с 0), имеющей **максимальное** значение при подстановке x.

8. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию с аргументом x типа `Int`, возвращающую **индекс первой** функции, имеющей **минимальное** значение при подстановке x.

9. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию с аргументом x типа `Int`, возвращающую **индекс последней** функции, имеющей **максимальное** значение при подстановке x.

10. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Int` и возвращаемым значением типа `Int`, и возвращает новую функцию с аргументом x типа `Int`, возвращающую **индекс последней** функции, имеющей **минимальное** значение при подстановке x.

11. Создайте функцию, которая по двум функциям с параметром типа `Int` и возвращаемым значением типа `Int?` возвращает новую функцию — сумму исходных. Если хотя бы одна из функций возвращает `null`, результат также должен быть `null`.

12. Создайте функцию, которая по двум функциям с параметром типа `Int` и возвращаемым значением типа `Int?` возвращает новую функцию — произведение исходных. Если хотя бы одна из функций возвращает `null`, результат также должен быть `null`.

13. Создайте функцию, которая по двум функциям с параметром типа `Int` и возвращаемым значением типа `Int?` возвращает новую функцию — максимум исходных. Если хотя бы одна из функций возвращает `null`, результат также должен быть `null`.

14. Создайте функцию, которая по двум функциям с параметром типа `Int` и возвращаемым значением типа `Int?` возвращает новую функцию — минимум исходных. Если хотя бы одна из функций возвращает `null`, результат также должен быть `null`.

15. Создайте функцию, которая по двум функциям f и g (обе: `Int` $\to$ `Int?`) возвращает функцию, вычисляющую композицию $f(g(x))$. Если $g(x) =$ `null`, то результат — `null`.

16. Создайте функцию, которая по функции f : `Int` $\to$ `Int` и целому числу n возвращает новую функцию, принимающую массив из n элементов типа `Int` и возвращающую массив результатов применения f к каждому элементу.

17. Создайте функцию, которая по данному массиву целых чисел возвращает функцию без параметров, которая при каждом вызове последовательно возвращает элементы массива. После исчерпания элементов возвращает `null`.

18. Создайте функцию, которая по функции f : `Int` $\to$ `Int` возвращает функцию без параметров, которая при каждом вызове последовательно возвращает $f(1)$, $f(2)$, $f(3)$, и так далее.

19. Создайте функцию, которая по данному массиву целых чисел возвращает функцию без параметров, которая при каждом вызове последовательно возвращает элементы массива в обратном порядке. После исчерпания элементов возвращает `null`.

20. Создайте функцию, которая по данной строке возвращает функцию без параметров, которая при каждом вызове последовательно возвращает символы строки. После исчерпания символов возвращает `null`.

21. Создайте функцию, которая по данной строке возвращает функцию без параметров, которая при каждом вызове последовательно возвращает символы строки в обратном порядке. После исчерпания символов возвращает `null`.

22. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Float` и возвращаемым значением типа `Float`, и возвращает новую функцию — среднее арифметическое исходных: $\frac{f_1(x)+f_2(x)+\cdots+f_n(x)}{n}$.

23. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Float` и возвращаемым значением типа `Float`, и возвращает новую функцию — среднее квадратическое исходных: $\sqrt{\frac{f_1(x)^2+f_2(x)^2+\cdots+f_n(x)^2}{n}}$.

24. Создайте функцию, которая принимает произвольное количество функций с параметром типа `Float` и возвращаемым значением типа `Float`, и возвращает новую функцию — среднее геометрическое исходных: $\sqrt[n]{f_1(x) \cdot f_2(x) \cdot \cdots \cdot f_n(x)}$. Предполагается, что все значения положительны.

25. Создайте функцию, которая по функции f : `Float` $\to$ `Float` и числу x возвращает функцию без параметров, которая при каждом вызове последовательно возвращает $f(x)$, $f(f(x))$, $f(f(f(x)))$, и так далее.

2.3 Семинар «Оператор when» (2 часа)

В работе предполагается освоение оператора when.

При выполнении заданий обращайте внимание на использование специфических особенностей языка везде, где это возможно: if и when могут быть как операторами, так и частью выражений; фигурные скобки во многих случаях можно опускать; точки с запятой почти никогда не используются, корректно выбирайте, как помечать переменные: ключевым словом var или val.

При выполнении работ обеспечивайте **оптимальность** предлагаемой программы как по скорости, так и по памяти. В случае противоречия между двумя критериями, выбирайте алгоритм, который обеспечивает лучшее быстродействие.

В частности, это обозначает, что нельзя использовать дополнительные строки (в заданиях, кроме как для ввода, строки не нужны), следует избегать сложных структур (списков, множеств).

В задачах предполагается, что не используются специальные функции, предназначенные для работы с датами.

1. По введённой дате определите дату следующего дня. Выведите её и проверьте, совпадает 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 (чтобы перепрыгнуть через год). Выведите её и проверьте, является ли полученная дата високосным днём (29 февраля).

2.4 Семинары «Функции высшего порядка в Kotlin/Функциональный подход в Kotlin» (4 часа)

В данной работе требуется написать не самую оптимальную реализацию, а реализацию, которая наиболее полноценно использует функции над коллекциями, использующие функциональный подход, и строковые функции. В работе запрещено использовать mutable коллекции и var переменные.

Реализация должна состоять из одной строки с точечными вызовами, включая ввод и вывод, использовать рекурсию запрещено.

Примечание: данный способ реализации программы нужен исключительно в учебных целях, в дальнейшем разбивайте подобные решения на небольшие функции, которые удобно повторно использовать.

Не забывайте проверять все входные данные от пользователя полноценно и всесторонне.

Задание №1

Для данного неотрицательного целого числа (в пределах Int) найдите указанный результат. Осуществите проверку корректности ввода. Оформите программу и устраните все warning.

1. Сумма чётных цифр
2. Сумма нечётных цифр
3. Произведение чётных цифр
4. Произведение нечётных цифр
5. Максимальная чётная цифра
6. Минимальная чётная цифра
7. Максимальная нечётная цифра
8. Минимальная нечётная цифра
9. Сумма цифр, кратных трём
10. Сумма цифр, не кратных трём
11. Произведение цифр, кратных трём
12. Произведение цифр, не кратных трём
13. Максимальная цифра, кратная трём
14. Минимальная цифра, кратная трём
15. Максимальная цифра, не кратная трём
16. Минимальная цифра, не кратная трём
17. Сумма цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 ответ 4
18. Произведение цифр, стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 ответ 3
19. Максимальная цифра среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 ответ 3
20. Минимальная цифра среди стоящих на чётных позициях в числе (если нумеровать цифры с конца): для числа 1234 ответ 1
21. Сумма цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 ответ 2
22. Произведение цифр, стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 ответ 2

23. Максимальная цифра среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 ответ 2

24. Минимальная цифра среди стоящих на позициях в числе, номера которых кратны трём (если нумеровать цифры с конца): для числа 1234 ответ 2

25. Максимальная цифра среди стоящих на позициях в числе, номера которых кратны четырём (если нумеровать цифры с конца): для числа 1234 ответ 1

Задание №2

Можно предполагать, что все символам соответствует одно значение типа char (однако, если вы сделаете корректное решение (разумеется, вне пары), то это будет плюсом).

1. Найдите первый символ в первом максимально длинном слове с нечётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

2. Найдите последний символ в первом максимально длинном слове с нечётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

3. Найдите первый символ в последнем максимально длинном слове с нечётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

4. Найдите последний символ в последнем максимально длинном слове с нечётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

5. Найдите первый символ в первом самом коротком слове в строке с нечётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

6. Найдите последний символ в первом самом коротком слове в строке с нечётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

7. Найдите первый символ в последнем самом коротком слове в строке с нечётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

8. Найдите последний символ в последнем самом коротком слове в строке с нечётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

9. Найдите первый символ в первом максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

10. Найдите последний символ в первом максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

11. Найдите первый символ в последнем максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

12. Найдите последний символ в последнем максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

13. Найдите первый символ в первом самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

14. Найдите последний символ в первом самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

15. Найдите первый символ в последнем самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

16. Найдите последний символ в последнем самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

17. Найдите второй символ в первом максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

18. Найдите предпоследний символ в первом максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

19. Найдите второй символ в последнем максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

20. Найдите предпоследний символ в последнем максимально длинном слове с чётным числом символов в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

21. Найдите второй символ в первом самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

22. Найдите предпоследний символ в первом самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

23. Найдите второй символ в последнем самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

24. Найдите предпоследний символ в последнем самом коротком слове в строке с чётным числом символов (в строке указываются только слова, разделённые одним или несколькими пробелами).

25. Найдите первый символ в первом максимально длинном слове с числом символов, кратным трём, в строке (в строке указываются только слова, разделённые одним или несколькими пробелами).

Задание №3

1. С клавиатуры вводится описание массива из 10 элементов в виде:

   номер:*значение*

   однако, порядок указания элементов может быть любой. Выведите все элементы массива в порядке возрастания номеров.

2. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют в каждом числе?

3. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют хотя бы в двух числах?

4. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют ровно в одном числе?

5. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют ровно в двух числах?

6. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры отсутствуют ровно в двух числах?

7. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры отсутствуют ровно в одном числе?

8. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какие символы присутствуют в каждом слове?

9. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какие символы присутствуют хотя бы в двух словах?

10. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какие символы присутствуют ровно в одном слове?

11. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какие символы отсутствуют ровно в одном слове?

12. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какие символы отсутствуют ровно в двух словах?

13. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какое количество чисел удовлетворяет условию отсутствия повторяющихся цифр?

14. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какое количество чисел удовлетворяет условию наличия повторяющихся цифр?

15. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какое количество слов удовлетворяет условию отсутствия повторяющихся символов?

16. В строке указано несколько слов, разделённых пробелами (по одному пробелу между словами). Какое количество слов удовлетворяет условию наличия повторяющихся символов?

17. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). В каком количестве чисел присутствуют все цифры от 0 до 9?

18. Имеется некоторая последовательность цифр от 0 до 9. С клавиатуры вводится 9 строк следующего вида:

    цифра->цифра

    Каждая строка обозначает, что после цифры, стоящей до стрелки, в последовательности стоит цифра, стоящая после стрелки.

    Выведите исходную последовательность.

19. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют в каждом числе дважды?

20. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют хотя бы в одном числе дважды?

21. В строке указано несколько неотрицательных целых чисел, разделённых пробелами (по одному пробелу между числами). Какие цифры присутствуют ровно в одном числе дважды?

22. С клавиатуры вводится несколько строк, последняя строка — пустая (пустая строка — признак окончания ввода и дальше игнорируется). Выведите символы, что присутствуют в каждой строке.

23. С клавиатуры вводится несколько строк, последняя строка — пустая (пустая строка — признак окончания ввода и дальше игнорируется). Выведите символы, что присутствуют ровно в одной строке.

24. С клавиатуры вводится несколько строк, последняя строка — пустая (пустая строка — признак окончания ввода и дальше игнорируется). Выведите символы, что присутствуют ровно в двух строках.

25. С клавиатуры вводится несколько строк, последняя строка — пустая (пустая строка — признак окончания ввода и дальше игнорируется). Выведите символы, что отсутствуют ровно в двух строках.

Задание №4 С клавиатуры вводится несколько целых значений через пробел. Найдите (без учета тех чисел, где соответствующей цифры нет):

1. Побитовое И предпоследней цифры всех чисел

2. Побитовое ИЛИ предпоследней цифры всех чисел

3. Побитовое исключающее ИЛИ предпоследней цифры всех чисел

4. Побитовый штрих Шеффера последней цифры всех чисел (операции выполняются слева направо)
5. Побитовый штрих Шеффера предпоследней цифры всех чисел (операции выполняются слева направо)
6. Побитовая стрелка Пирса последней цифры всех чисел (операции выполняются слева направо)
7. Побитовая стрелка Пирса предпоследней цифры всех чисел (операции выполняются слева направо)
8. Побитовый штрих Шеффера последней цифры всех чисел (операции выполняются справа налево)
9. Побитовый штрих Шеффера предпоследней цифры всех чисел (операции выполняются справа налево)
10. Побитовая стрелка Пирса последней цифры всех чисел (операции выполняются справа налево)
11. Побитовая стрелка Пирса предпоследней цифры всех чисел (операции выполняются справа налево)
12. Побитовое И первой цифры всех чисел
13. Побитовое ИЛИ первой цифры всех чисел
14. Побитовое исключающее ИЛИ первой цифры всех чисел
15. Побитовое И второй цифры всех чисел
16. Побитовое ИЛИ второй цифры всех чисел
17. Побитовое исключающее ИЛИ второй цифры всех чисел
18. Побитовый штрих Шеффера первой цифры всех чисел (операции выполняются слева направо)
19. Побитовый штрих Шеффера второй цифры всех чисел (операции выполняются слева направо)
20. Побитовая стрелка Пирса первой цифры всех чисел (операции выполняются слева направо)
21. Побитовая стрелка Пирса второй цифры всех чисел (операции выполняются слева направо)
22. Побитовый штрих Шеффера первой цифры всех чисел (операции выполняются справа налево)
23. Побитовый штрих Шеффера второй цифры всех чисел (операции выполняются справа налево)
24. Побитовая стрелка Пирса первой цифры всех чисел (операции выполняются справа налево)
25. Побитовая стрелка Пирса второй цифры всех чисел (операции выполняются справа налево)

Задание №5

1. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх лучших студентах по среднему баллу. В случае, если у нескольких студентов средний балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя лучшими баллами). Вывод надо осуществлять в порядке убывания среднего балла, а для одинаковых средних баллов — в алфавитном порядке по фамилии и имени.
2. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх лучших студентах по максимальному баллу. В случае, если у нескольких студентов максимальный балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя лучшими баллами). Вывод надо осуществлять в порядке убывания максимального балла, а для одинаковых максимальных баллов — в алфавитном порядке по фамилии и имени.

3. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх лучших студентах по минимальному баллу. В случае, если у нескольких студентов минимальный балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя лучшими баллами). Вывод надо осуществлять в порядке убывания минимального балла, а для одинаковых минимальных баллов — в алфавитном порядке по фамилии и имени.

4. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх худших студентах по среднему баллу. В случае, если у нескольких студентов средний балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя худшими баллами). Вывод надо осуществлять в порядке возрастания среднего балла, а для одинаковых средних баллов — в алфавитном порядке по фамилии и имени.

5. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх худших студентах по максимальному баллу. В случае, если у нескольких студентов максимальный балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя худшими баллами). Вывод надо осуществлять в порядке возрастания максимального балла, а для одинаковых максимальных баллов — в алфавитном порядке по фамилии и имени.

6. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх худших студентах по минимальному баллу. В случае, если у нескольких студентов минимальный балл совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя худшими баллами). Вывод надо осуществлять в порядке возрастания минимального балла, а для одинаковых минимальных баллов — в алфавитном порядке по фамилии и имени.

7. С клавиатуры вводится информация об абитуриентах: фамилия, имя, а далее названия предметов и оценки ЕГЭ по ним. Выведите на экран информацию о трёх лучших абитуриентах по максимальному баллу за сумму трёх ЕГЭ. В случае, если у нескольких абитуриентов сумма баллов совпадает, то выведите большее число абитуриентов (пока не будут выведены все абитуриенты или не будут полностью исчерпаны абитуриенты с тремя лучшими баллами). Вывод надо осуществлять в порядке убывания суммы баллов за три ЕГЭ, а для одинаковых сумм — в алфавитном порядке по фамилии и имени.

8. С клавиатуры вводится информация об абитуриентах: фамилия, имя, а далее названия предметов и оценки ЕГЭ по ним. Выведите на экран информацию о трёх худших абитуриентах по максимальному баллу за сумму трёх ЕГЭ. В случае, если у нескольких абитуриентов сумма баллов совпадает, то выведите большее число абитуриентов (пока не будут выведены все абитуриенты или не будут полностью исчерпаны абитуриенты с тремя худшими баллами). Вывод надо осуществлять в порядке возрастания суммы баллов за три ЕГЭ, а для одинаковых сумм — в алфавитном порядке по фамилии и имени.

9. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наибольшим количеством пятёрок. В случае, если у нескольких студентов количество пятёрок совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя лучшими показателями). Вывод надо осуществлять в порядке убывания количества пятёрок, а для одинакового количества — в алфавитном порядке по фамилии и имени.

10. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наибольшим количеством двоек. В случае, если у нескольких студентов количество двоек совпадает, то выведите большее число студентов (пока не будут выведены все студенты или не будут полностью исчерпаны студенты с тремя худшими показателями). Вывод надо осуществлять в порядке убывания количества двоек, а для одинакового количества — в алфавитном порядке по фамилии и имени.

11. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наибольшим разбросом оценок (разница между максимальной и минимальной). В случае, если у нескольких студентов разброс совпадает, то выведите большее число студентов. Вывод в порядке убывания разброса, затем по алфавиту.

12. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наименьшим разбросом оценок. В случае, если у нескольких студентов разброс совпадает, то выведите большее число студентов. Вывод в порядке возрастания разброса, затем по алфавиту.

13. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наибольшей медианой оценок. В случае совпадения — выведите всех, сортируя по алфавиту.

14. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о трёх студентах с наименьшей медианой оценок. В случае совпадения — выведите всех, сортируя по алфавиту.

15. С клавиатуры вводится информация о студентах: фамилия, имя, курс, оценки. Выведите на экран информацию о лучших студентах по среднему баллу на каждом курсе (по одному с каждого курса). Если на курсе несколько лучших — выведите всех, сортируя по алфавиту.

16. С клавиатуры вводится информация о студентах: фамилия, имя, факультет, оценки. Выведите на экран информацию о лучших студентах по среднему баллу на каждом факультете (по одному с каждого факультета). Если на факультете несколько лучших — выведите всех, сортируя по алфавиту.

17. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых все оценки — пятёрки. Вывод в алфавитном порядке. Если таких больше трёх — выведите первых трёх по алфавиту.

18. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых есть хотя бы одна двойка. Отсортируйте по возрастанию среднего балла, затем по алфавиту. Выведите первых трёх.

19. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых средний балл выше 4.5. Отсортируйте по убыванию среднего балла, затем по алфавиту. Выведите первых трёх.

20. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых средний балл ниже 3.0. Отсортируйте по возрастанию среднего балла, затем по алфавиту. Выведите первых трёх.

21. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, имеющих наибольшее количество оценок (например, если у кого-то 5 оценок, а у других — 4). Выведите всех таких, но не более пяти, отсортировав по алфавиту.

22. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, имеющих наименьшее количество оценок. Выведите всех таких, но не более пяти, отсортировав по алфавиту.

23. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых мода оценок — 5 (наиболее часто встречающаяся оценка). Отсортируйте по среднему баллу (убывание), затем по алфавиту. Выведите первых трёх.

24. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых мода оценок — 2. Отсортируйте по среднему баллу (возрастание), затем по алфавиту. Выведите первых трёх.

25. С клавиатуры вводится информация о студентах: фамилия, имя, оценки. Выведите на экран информацию о студентах, у которых стандартное отклонение оценок минимально (наиболее стабильные). Выведите трёх лучших, сортируя по возрастанию отклонения, затем по алфавиту.

Задание №6

1. По номеру числа Фибоначчи найдите число Фибоначчи (не используйте факты, которые вы не можете доказать самостоятельно)

2. По числу Фибоначчи найдите его номер (не используйте факты, которые вы не можете доказать самостоятельно)
3. По натуральному числу найдите его факториал
4. По факториалу найдите исходное число
5. По данному числу найдите простое число с таким номером (если простые числа нумеровать в порядке возрастания)
6. По простому числу определите его номер в последовательности всех простых чисел, расположенных по возрастанию
7. По данному числу найдите все его простые делители
8. По числу n найдите n-ое совершенное число (не используйте факты, которые вы не можете доказать самостоятельно)
9. По совершенному числу найдите его номер в последовательности всех совершенных чисел, расположенных в порядке возрастания (не используйте факты, которые вы не можете доказать самостоятельно)
10. По натуральному числу найдите его двойной факториал
11. По двойному факториалу найдите исходное число
12. Рассмотрим все тройки натуральных чисел, удовлетворяющих уравнению $a^2 + b^2 = c^2$. Для данного n найдите такую тройку чисел a, b, c, что $a^2 + b^2 = c^2$, чтобы $a + b + c$ было меньше n и наиболее близко к n.
13. Рассмотрим все тройки натуральных чисел, удовлетворяющих уравнению $a^2 + b^2 = c^2$. Для данного n найдите такую тройку чисел a, b, c, что $a^2 + b^2 = c^2$, чтобы $a + b + c$ было больше n и наиболее близко к n.
14. По данному натуральному числу n найдите наименьшее простое число, большее n
15. По данному натуральному числу n найдите наибольшее простое число, меньшее n
16. По данному натуральному числу n найдите наименьший факториал, больший n
17. По данному натуральному числу n найдите наибольший факториал, меньший n
18. По данному натуральному числу n найдите наименьший двойной факториал, больший n
19. По данному натуральному числу n найдите наибольший двойной факториал, меньший n
20. По данному натуральному числу n найдите наименьшее число Фибоначчи, большее n (не используйте факты, которые вы не можете доказать самостоятельно)
21. По данному натуральному числу n найдите наибольшее число Фибоначчи, меньшее n (не используйте факты, которые вы не можете доказать самостоятельно)
22. По данному натуральному числу n найдите наименьшее совершенное число, большее n (не используйте факты, которые вы не можете доказать самостоятельно)
23. По данному натуральному числу n найдите наибольшее совершенное число, меньшее n (не используйте факты, которые вы не можете доказать самостоятельно)
24. Для данного натурального числа n найдите такое простое число p, что входит в разложение на простые множители числа n наибольшее число раз.
25. Для данного натурального числа n найдите такое простое число p, что входит в разложение на простые множители числа n наименьшее число раз.

2.5 Семинар «Оператор when с проверкой типов (is) и Unit-тестирование» (2 часа)

Цель работы — освоить использование оператора `when` для безопасной обработки значений разных типов.

При выполнении заданий используйте идиоматичный Kotlin:

- `if` и `when` могут быть выражениями (возвращать значение);
- фигурные скобки можно опускать для однострочных веток;
- точки с запятой не используются;
- переменные помечайте `val`, если их значение не меняется.

Разработайте функцию для вашего варианта задания, а также Unit-тесты для проверки её работы во всех случаях. Для написания Unit-тестов используйте библиотеку `kotlin.test`. Напишите минимум 3–5 тестов, покрывающих разные типы входных данных, включая `null` и неожиданные типы (при этом должно быть полное покрытие).

1. **toStringSafe**: Вернуть строковое представление значения. Для `Int`, `Double`, `Boolean` — стандартное `toString()`. Для `String` — саму строку. Для `List<Any?>` — строку вида "[элемент1, элемент2, ...]"(элементы тоже через `toStringSafe`). Для `null` — "null". Для всех остальных типов — пустую строку.
2. **toNumericValue**: Вернуть числовое значение. Для `Int`, `Double`, `Long` — само число как `Double`. Для `String`, содержащей число — преобразовать и вернуть. Для `Boolean` — 1.0 (true) или 0.0 (false). Для всех остальных — null.
3. **getSizeOrLength**: Вернуть "размер"значения. Для `String` — длина. Для `List`, `Array`, `Set` — количество элементов. Для `Map` — количество ключей. Для `Int`, `Double` — количество цифр в числе (игнорируя знак и точку). Для всех остальных — -1.
4. **isEmptyOrZero**: Вернуть `true`, если значение "пустое". Для `String` — если пустая или null. Для `List`, `Set`, `Array` — если пусто или null. Для `Int`, `Double` — если равно 0. Для `Boolean` — если false. Для всех остальных — true.
5. **toBooleanSafe**: Вернуть логическое значение. Для `Boolean` — само значение. Для `Int` — true, если не 0. Для `String` — true, если "true "1 "yes"(без учёта регистра). Для `List`, `Array` — true, если не пусто. Для всех остальных — false.
6. **getFirstElement**: Вернуть первый элемент или символ. Для `String` — первый символ (или null, если пусто). Для `List`, `Array` — первый элемент (или null). Для `Int` — первую цифру числа. Для всех остальных — null.
7. **getLastElement**: Вернуть последний элемент или символ. Для `String` — последний символ. Для `List`, `Array` — последний элемент. Для `Int` — последнюю цифру. Для всех остальных — null.
8. **toDisplayString**: Вернуть человекочитаемое описание. Для `String` — "Строка: <значение>". Для `Int` — "Целое число: <значение>". Для `Boolean` — "Логическое: <значение>". Для `List` — "Список из N элементов". Для всех остальных — "Неизвестный тип".
9. **countDigitsOrLetters**: Вернуть количество цифр (если число) или букв (если строка). Для `Int`, `Double` — количество цифр. Для `String` — количество букв (a-z, A-Z). Для `Boolean` — 4 (длина слова "true") или 5 ("false"). Для всех остальных — 0.
10. **toComparableValue**: Вернуть значение, пригодное для сравнения (в виде `Double`). Для `Int`, `Double` — само число. Для `String` — длину. Для `Boolean` — 1.0 или 0.0. Для `List`, `Array` — размер. Для всех остальных — 0.0.
11. **isValidInput**: Вернуть `true`, если значение может быть использовано как корректный ввод для калькулятора. Для `Int`, `Double` — true. Для `String`, содержащей число — true. Для `Boolean` — false. Для всех остальных — false.

12. **toSearchableString**: Вернуть строку, пригодную для поиска (в нижнем регистре, без лишних символов). Для `String` — в нижнем регистре. Для `Int`, `Double` — `toString().lowercase()`. Для `Boolean` — "true"/"false". Для `List` — конкатенация `toSearchableString` всех элементов через пробел. Для всех остальных — пустая строка.

13. **getHashCode**: Вернуть целочисленный хэш-код (упрощённый). Для `Int` — само число. Для `String` — сумму кодов всех символов. Для `Boolean` — 1 или 0. Для `List` — сумму хэшей всех элементов. Для всех остальных — 0.

14. **toLogString**: Вернуть строку для логирования в формате "[ТИП] значение". Для `String` — "[String] значение". Для `Int` — "[Int] значение". Для `null` — "[Null] null". Для всех остальных — "[Unknown] тип: неизвестен".

15. **canBeNegative**: Вернуть `true`, если тип значения теоретически может быть отрицательным. Для `Int`, `Double`, `Long` — true. Для `String`, `Boolean`, `List` — false. Для всех остальных — false.

16. **toPositiveValue**: Вернуть положительное числовое представление. Для `Int`, `Double` — модуль числа. Для `String` — длину. Для `Boolean` — 1 (true) или 0 (false). Для `List` — размер. Для всех остальных — 0.

17. **getSummary**: Вернуть краткую сводку. Для `String` — "Длина: N". Для `Int` — "Значение: N". Для `List` — "Элементов: N". Для `Boolean` — "Значение: true/false". Для всех остальных — "Тип: Unknown".

18. **isNumericType**: Вернуть `true`, если значение представляет число. Для `Int`, `Double`, `Long` — true. Для `String`, если можно распарсить как число — true. Для всех остальных — false.

19. **getDefaultName**: Вернуть имя по умолчанию. Для `String` — само значение. Для `Int` — "Item_N". Для `Boolean` — "Флаг". Для `List` — "Список_N_элементов". Для всех остальных — "Объект".

20. **toIdentifier**: Вернуть строку, пригодную для использования как идентификатор (латинские буквы, цифры, без пробелов). Для `String` — оставить только a-z, A-Z, 0-9, остальное удалить. Для `Int` — "id_N". Для `Boolean` — "flag_true"/"flag_false". Для всех остальных — "unknown_type".

21. **getByteSizeEstimate**: Вернуть примерный размер в байтах. Для `Int` — 4. Для `Double` — 8. Для `String` — длина * 2. Для `Boolean` — 1. Для `List` — сумму размеров элементов + 10. Для всех остальных — 0.

22. **toUserFriendlyString**: Вернуть строку, понятную пользователю. Для `null` — "Не задано". Для `String` — само значение. Для `Int` — форматированное число с разделителями тысяч. Для `Boolean` — "Да"/"Нет". Для всех остальных — "Системное значение".

23. **isPrimitiveLike**: Вернуть `true`, если значение похоже на примитив (число, строка, булево). Для `Int`, `Double`, `String`, `Boolean` — true. Для `List`, `Map`, `Array` — false. Для всех остальных — false.

24. **toSortKey**: Вернуть строку-ключ для сортировки. Для `String` — в нижнем регистре. Для `Int`, `Double` — дополнить нулями до 10 символов. Для `Boolean` — "0_false"/"1_true". Для всех остальных — "z_unknown".

25. **getDeepSize**: Вернуть "глубину"или сложность структуры. Для `Int`, `String`, `Boolean` — 1. Для `List`, `Array` — 1 + максимум `getDeepSize` элементов. Для всех остальных — 0.

2.6 Семинар «Поддержка ООП в Kotlin» (4 часа)

Напишите программу, осуществляющую ввод информации о сущностях, описанных в вашем варианте задания, и вывод на экран части из них. Количество вводимых сущностей не ограничено. При реализации обязательно используйте принципы объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм, а также следуйте принципам SOLID:

- **Принцип единственной ответственности**: класс не должен одновременно отвечать за хранение/бизнес-логику и за ввод-вывод. Разделяйте эти обязанности.
- **Принцип открытости/закрытости**: ограничивайте возможность некорректного использования классов — как при обычном применении, так и при наследовании и переопределении методов. В то же время обеспечьте гибкость: если наследование и переопределение позволяют адаптировать поведение под конкретные нужды, они должны быть разрешены.
- **Принцип подстановки Барбары Лисков**: если класс `A` наследует `B`, то `A` должен быть корректной реализацией `B`, и объект типа `A` должен корректно работать в любом контексте, где ожидается `B`.
- **Принцип разделения интерфейсов**: проектируйте небольшие, целенаправленные интерфейсы (`interface` в Kotlin), соответствующие конкретным потребностям клиентов. Классы могут реализовывать один или несколько таких интерфейсов. Избегайте «жирных» интерфейсов, заставляющих классы зависеть от методов, которые им не нужны.
- **Принцип инверсии зависимостей**: модули верхнего уровня (конкретные классы) должны зависеть от абстракций (интерфейсов или базовых классов), а не от других конкретных классов. Абстракции не должны зависеть от деталей; детали — от абстракций.

Выберите один из следующих вариантов задания:

1. **Товары интернет-магазина — книги и диски.** Все товары имеют цену. Книги — название, автора, количество страниц. Диски — название, количество треков. Выведите все товары со стоимостью менее 100 рублей.
2. **Преподаватели.** Определяются ФИО. У тех, кто имеет диссертацию, указывается её название; у остальных — стаж работы. Выведите всех преподавателей, чьё ФИО начинается на букву «А».
3. **Телефоны.** Определяются названием модели. Проводные телефоны — типом номеронабирателя (диск или кнопки); беспроводные — дальностью действия радиосигнала. Выведите все телефоны, название которых начинается на «А».
4. **Покатушки.** Определяются названием и расстоянием. Однодневные — плановым временем поездки (в часах). Многодневные — количеством дней и категорией сложности (от 1 до 6). Выведите все покатушки длиной более 100 км.
5. **Музыкальные композиции.** Определяются названием и композитором. Для песен дополнительно указывается автор стихов. Выведите информацию о всех композициях, у которых композитор начинается на «А».
6. **Олимпиады.** Определяются названием. Если олимпиада участвует в программе приёма в вузы — указывается уровень (1–3). Если это этап Всероссийской олимпиады — название этапа (школьный, окружной, региональный, всероссийский). В остальных случаях — размер призового фонда. Выведите все олимпиады, название которых начинается на «А».
7. **Проездные билеты.** Определяются стоимостью. Билет на количество поездок — количеством поездок. Билет на неограниченное количество поездок — сроком действия (1 день, 5, 10, 15 дней, месяц, 3 месяца, 6 месяцев, год). Выведите билеты стоимостью менее 300 рублей.
8. **Студенты.** Определяются ФИО. У студентов без автомата — балл за экзамен (2–5). У студентов с автоматом — основание (олимпиада или контрольные работы); если основание — контрольные, то также указывается средний балл за них. Выведите информацию о студентах с фамилией, начинающейся на буквы от «А» до «К».

9. **Сотовые телефоны.** Определяются названием. У смартфонов — операционная система; у остальных — наличие браузера. Выведите телефоны, название которых содержит слово «Nokia».

10. **Куртки.** Определяются названием модели и наличием капюшона. Мембранные куртки — степенью водонепроницаемости (в мм вод. ст.); остальные — наличием пропитки. Выведите информацию обо всех куртках с капюшоном.

11. **Жёсткие диски.** Определяются названием и ёмкостью. Внешние — наличием системы защиты от падения; внутренние — размером (2.5"или 3.5"). Выведите диски ёмкостью более 200 ГБ.

12. **Велосипеды.** Определяются названием модели. Горные — количеством скоростей; BMX — типом конструкции (фривил, кассетная, фрикостер). Выведите информацию обо всех велосипедах, в названии которых есть «Norco».

13. **Электронные книги.** Определяются названием и размером экрана. У E-Ink — поколение дисплея (Pearl, Vizplex); у LCD — количество поддерживаемых цветов. Выведите книги с экраном не менее 7 дюймов.

14. **GPS-навигаторы.** Определяются названием и диагональю экрана. Переносные — наличием велокрепления; автомобильные — поддержкой отображения пробок и наличием радар-детектора. Выведите GPS с экраном менее 7 дюймов.

15. **Пылесосы.** Определяются названием модели. Обычные — мощностью; роботы-пылесосы — площадью убираемого помещения и количеством виртуальных стен. Выведите пылесосы, в названии которых есть «Indesit».

16. **Туры.** Определяются названием. Пляжные — типом пляжа (галечный, песок); экскурсионные — количеством экскурсий. Выведите туры, содержащие слово «Египет».

17. **Языки программирования.** Определяются названием. Алгоритмические языки — поддержкой ООП (отсутствует, на классах, прототипная); остальные — типом (функциональный, логический, императивный и т.п.). Выведите языки, название которых начинается на «А».

18. **Контрагенты.** Определяются названием. ИП — наличием банковского счёта; юридические лица — формой организации (ООО, ОАО, ЗАО). Выведите контрагентов, название которых начинается на «А».

19. **Банковские счета.** Определяются номером. Текущие — платой за обслуживание; сберегательные — процентной ставкой и наличием капитализации. Выведите счета, номер которых начинается с «408178».

20. **Автомобильные дороги.** Определяются названием и длиной (в км). Бесплатные — статусом автомагистрали (да/нет); платные — стоимостью за км. Выведите дороги длиной менее 100 км.

21. **Офисные здания.** Определяются адресом. При наличии стоянки — количеством машиномест и стоимостью аренды в месяц. Выведите здания, в адресе которых есть слово «Тверская».

22. **Товары интернет-магазина — GPS-навигаторы и карты.** Все товары — ценой и названием. GPS — назначением (ручной, автомобильный) и возможностью загрузки карт; карты — размером (в МБ). Выведите товары стоимостью менее 4000 рублей.

23. **Товары интернет-магазина — чаи и кофе.** Все товары — ценой, названием и весом. Кофе — типом (растворимый, молотый, в зёрнах); чаи — типом (чёрный, зелёный). Выведите товары весом менее 150 г.

24. **Объекты в коттеджном посёлке — участки и дома.** Участки — площадью, стоимостью, наличием подряда. Дома — этажностью, площадью, стоимостью. Выведите объекты стоимостью менее 1 000 000 рублей.

25. **Вопросы теста.** Общие поля — формулировка и баллы за правильный ответ. Вопросы с вариантами — 4 варианта и номер правильного; остальные — формулировка правильного ответа. Выведите вопросы, оцениваемые в 10 баллов и выше.

26. **Слова.** Определяются самим словом. Существительные — родом; глаголы — спряжением. Выведите слова, начинающиеся на «А».

27. **Операционные системы.** Определяются названием. На базе Linux — названием менеджера пакетов; остальные — стоимостью лицензии. Выведите ОС, название которых начинается на «А».

28. **Рюкзаки.** Определяются названием модели и ёмкостью. Городские — наличием «вентилируемой спины»; походные — количеством отделений и креплением для трекинговых палок. Выведите рюкзаки, в названии которых есть «Trek».

29. **Автостоянки.** Определяются названием и количеством машиномест. Крытые — количеством этажей; открытые — наличием охраны. Выведите стоянки с более чем 20 местами.

30. **Политические партии.** Определяются названием. Финансируемые из бюджета — размером ассигнований; остальные — количеством депутатов в представительных органах. Выведите партии, название которых начинается на буквы от «А» до «К».

2.7 Семинар «Делегирование в Kotlin» (2 часа)

Теоретическое введение

В языке Kotlin делегирование (delegation) реализуется с помощью ключевого слова `by`. Этот механизм позволяет классу делегировать реализацию интерфейса другому объекту, что способствует повторному использованию кода без наследования. Делегирование особенно полезно при реализации паттерна *Decorator*, когда необходимо динамически расширять функциональность объекта.

Рассмотрим пример, в котором класс `LoggerPrinter` реализует интерфейс `Printer`, делегируя один метод и переопределяя другой:

```
interface Printer {
    fun print(message: String)
    fun log(message: String)
}

class BasicPrinter : Printer {
    override fun print(message: String) {
        println("Printing: $message")
    }

    override fun log(message: String) {
        println("Log: $message")
    }
}

class LoggerPrinter(printer: Printer) : Printer by printer {
    // Метод log делегируется базовому printer
    // Метод print переопределяется
    override fun print(message: String) {
        println("[SECURE] Printing: $message")
    }
}
```

В этом примере:

- метод `log` автоматически делегируется объекту `printer` (благодаря `by printer`);
- метод `print` переопределяется для добавления дополнительной логики.

Такой подход позволяет гибко комбинировать поведения без дублирования кода и без необходимости строить сложные иерархии наследования, особенно в условиях отсутствия множественного наследования.

Реализуйте несколько классов, демонстрирующих применение паттерна *Decorator* с использованием механизма делегирования, встроенного в Kotlin. Обратите внимание: приведённые задачи трудно или невозможно решить с помощью наследования без дублирования кода или изобретения избыточно сложных иерархий (особенно в языках без множественного наследования).

1. **Преподаватели.** Осуществите ввод информации о преподавателях и выведите тех, у кого средний балл учащихся выше 4. Преподаватель характеризуется ФИО, полом и средним баллом учащихся. Дополнительно:
   - если имеет категорию — дата последнего подтверждения и номер приказа;
   - если имеет кандидатскую степень — тема диссертации, научное направление, дата защиты.
2. **Студенты.** Введите информацию о студентах и выведите тех, у кого средний балл выше 4. Студент характеризуется ФИО, полом и средним баллом. Дополнительно:
   - если платник — номер договора и признак отсутствия задолженности;
   - если военнообязанный — номер приписного свидетельства/военного билета, название военкомата, признак прохождения службы.

3. **Остановочные пункты ЖД.** Введите данные об остановочных пунктах пригородного железнодорожного транспорта и выведите те, где пассажиропоток превышает 1000 чел./сутки. Пункт характеризуется названием, пассажиропотоком и расстоянием от вокзала. Дополнительно:
   - при наличии турникетного комплекса — количество павильонов и стоимость билета «на выход»;
   - если рядом со станцией метро — название станции и расстояние до неё.
4. **ВУЗы.** Введите информацию о вузах и выведите те, где обучается более 2000 студентов. ВУЗ характеризуется названием, числом студентов, адресом и номером лицензии. Дополнительно:
   - при наличии аккредитации — номер и дата окончания;
   - если выдаёт документы международного образца — стоимость выдачи и регион признания (например, «ЕС», «США», «весь мир»).
5. **Планшеты.** Введите данные о планшетах и выведите те, что стоят менее 15 000 рублей. Планшет характеризуется названием, производителем и стоимостью. Дополнительно:
   - при поддержке SIM-карты — поддержка 3G и LTE;
   - при поддержке Bluetooth — версия стандарта и возможность подключения клавиатуры.
6. **Смартфоны.** Введите информацию о смартфонах и выведите те, что стоят менее 10 000 рублей. Смартфон характеризуется моделью, производителем и ценой. Дополнительно:
   - при поддержке Wi-Fi — стандарт и максимальная скорость;
   - при поддержке геопозиционирования — поддержка GPS/ГЛОНАСС и количество каналов.
7. **Телевизоры.** Введите данные о телевизорах и выведите те, что стоят менее 10 000 рублей. Телевизор характеризуется маркой, производителем и диагональю. Дополнительно:
   - при поддержке Smart TV — название платформы и предустановленные приложения;
   - при поддержке многоканального звука — наличие оптического и поканальных аудиовыходов.
8. **Накопители.** Введите информацию о накопителях и выведите те, у которых ёмкость менее 100 ГБ. Накопитель характеризуется моделью, производителем и ёмкостью. Дополнительно:
   - при интерфейсе USB — версия стандарта и скорость передачи;
   - если это жёсткий диск — количество поверхностей и физический размер (2.5 3.5").
9. **Компьютеры.** Введите данные о компьютерах и выведите те, что стоят менее 20 000 рублей. Компьютер характеризуется производителем, моделью и стоимостью. Дополнительно:
   - если моноблок — диагональ дисплея;
   - при поддержке Wi-Fi — стандарт и максимальная скорость.
10. **Телеканалы.** Введите информацию о телеканалах и выведите те, чья аудитория превышает 1 000 000 зрителей. Телеканал характеризуется названием и размером аудитории. Дополнительно:
    - если эфирный — диапазон (МВ/ДМВ) и частота;
    - если государственный — объём финансирования.
11. **Реки.** Введите данные о реках и выведите те, длина которых превышает 20 км. Река характеризуется названием и длиной. Дополнительно:
    - если судоходная — ширина и глубина русла;
    - если используется для водоснабжения — суточный водозабор и название водопроводной станции.

12. **Памятники.** Введите информацию о памятниках и выведите те, у которых рейтинг выше 2.5. Памятник характеризуется названием, адресом и рейтингом. Дополнительно:
    - если посвящён одному человеку — ФИО и годы жизни;
    - если связан с военными действиями — название и годы войны.
13. **ПИФы.** Введите данные о паевых инвестиционных фондах и выведите те, что основаны до 2008 года. ПИФ характеризуется названием управляющей компании, названием фонда и годом основания. Дополнительно:
    - для интервального ПИФа — даты начала и окончания следующего интервала;
    - для ПИФа акций — эшелон акций (первый, второй, другие).
14. **Вклады.** Введите информацию о банковских вкладах и выведите те, у которых доходность выше 9%. Вклад характеризуется названием, банком и процентной ставкой. Дополнительно:
    - если разрешены пополнения — за сколько дней до окончания можно вносить средства и минимальный взнос;
    - если разрешено снятие — через сколько дней после открытия и размер неснижаемого остатка.
15. **Маршруты электричек.** Введите данные о маршрутах и выведите те, длина которых превышает 50 км. Маршрут характеризуется начальным и конечным пунктом, а также длиной. Дополнительно:
    - если экспресс — стоимость проезда и время в пути;
    - если межобластной — названия соответствующих областей.
16. **Холодильники.** Введите информацию о холодильниках и выведите те, высота которых превышает 1.5 м. Холодильник характеризуется габаритами (высота, ширина, глубина), моделью и производителем. Дополнительно:
    - при наличии морозильной камеры — её габариты;
    - при наличии зоны свежести — её габариты.
17. **Наушники.** Введите данные о наушниках и выведите те, что стоят более 500 рублей. Наушники характеризуются производителем, моделью, наличием активного шумоподавления и ценой. Дополнительно:
    - если беспроводные — радиус действия и признак цифрового сигнала;
    - если вставные — количество сменных амбушюров и наличие регулятора громкости.
18. **Электронные книги.** Введите информацию об электронных книгах и выведите те, что стоят более 3000 рублей. Книга характеризуется моделью, производителем и стоимостью. Дополнительно:
    - при дисплее на электронных чернилах — цветность (ч/б или цветной) и наличие подсветки;
    - при сенсорном интерфейсе — тип сенсора (ёмкостный, резистивный и т.д.).
19. **Устройства GPS.** Введите данные об устройствах GPS и выведите те, что стоят более 5000 рублей. Устройство характеризуется моделью, производителем, стоимостью и поддержкой ГЛОНАСС. Дополнительно:
    - если туристическое — водонепроницаемость, наличие магнитного компаса, тип батареи;
    - если цветной дисплей — количество поддерживаемых цветов.
20. **Туристические палатки.** Введите информацию о палатках и выведите те, у которых водостойкость дна превышает 7000 мм. Палатка характеризуется производителем, моделью, водостойкостью тента и дна. Дополнительно:
    - при наличии внутренней палатки — её ширина и высота;

- если многоместная — количество мест и входов.

21. **Обогреватели.** Введите данные об обогревателях и выведите те, что стоят более 1500 рублей. Обогреватель характеризуется производителем, моделью, стоимостью и мощностью. Дополнительно:
    - если инфракрасный — тип (галогенный, карбоновый, кварцевый) и наличие автоматического вращения;
    - если поддерживает программирование — возможность отсрочки старта и регулировки температуры.
22. **Автомобили.** Введите информацию об автомобилях и выведите те, у которых пробег менее 50 000 км. Автомобиль характеризуется маркой, моделью, годом выпуска и пробегом. Дополнительно:
    - если электромобиль — ёмкость батареи и запас хода;
    - если имеет систему помощи водителю — поддержка адаптивного круиз-контроля и автоматического торможения.
23. **Кофемашины.** Введите данные о кофемашинах и выведите те, что стоят более 10 000 рублей. Кофемашина характеризуется брендом, моделью и ценой. Дополнительно:
    - если с молочным резервуаром — объём резервуара и возможность регулировки пены;
    - если поддерживает капсулы — совместимые системы (Nespresso, Dolce Gusto и др.).
24. **Фотоаппараты.** Введите информацию о фотоаппаратах и выведите те, у которых разрешение матрицы выше 20 МП. Фотоаппарат характеризуется брендом, моделью и разрешением матрицы. Дополнительно:
    - если зеркальный — наличие оптического видоискателя и количество точек фокусировки;
    - если поддерживает 4K-видео — максимальная частота кадров и битрейт.
25. **Часы.** Введите данные о часах и выведите те, что стоят более 5000 рублей. Часы характеризуются брендом, типом (механические/кварцевые/умные) и ценой. Дополнительно:
    - если водонепроницаемые — глубина погружения и стандарт (ISO 6425 и др.);
    - если умные — поддержка ОС (Wear OS, watchOS) и время автономной работы.
26. **Книги.** Введите информацию о книгах и выведите те, у которых количество страниц превышает 300. Книга характеризуется названием, автором и количеством страниц. Дополнительно:
    - если учебник — предмет и класс;
    - если художественная — жанр и год первого издания.
27. **Фильмы.** Введите данные о фильмах и выведите те, чей рейтинг выше 7.0. Фильм характеризуется названием, режиссёром и рейтингом. Дополнительно:
    - если научная фантастика — наличие сиквелов и бюджет;
    - если анимационный — студия-производитель и возрастное ограничение.
28. **Рестораны.** Введите информацию о ресторанах и выведите те, у которых средний чек ниже 1000 рублей. Ресторан характеризуется названием, кухней и средним чеком. Дополнительно:
    - если доставка — минимальная сумма заказа и зона доставки;
    - если есть детское меню — возрастные категории и наличие игровой зоны.
29. **Спортивные залы.** Введите данные о спортивных залах и выведите те, у которых абонемент дешевле 3000 рублей/мес. Зал характеризуется названием, адресом и стоимостью месячного абонемента. Дополнительно:
    - если есть бассейн — длина дорожки и температура воды;
    - если круглосуточный — наличие охраны и системы видеонаблюдения.

30. **Музыкальные инструменты.** Введите информацию об инструментах и выведите те, что стоят менее 10 000 рублей. Инструмент характеризуется типом (гитара, синтезатор и т.д.), брендом и ценой. Дополнительно:

    - если электронный — количество клавиш и наличие MIDI-выхода;
    - если струнный — материал струн и наличие кейса.

2.8 Семинар «Корутины» (6 часов; подведение итогов по Kotlin)

Для работы с консольным интерфейсом необходимо подключить библиотеку Lanterna. Добавьте следующую зависимость в ваш `build.gradle.kts` (или аналогичный файл сборки):

```
implementation("com.googlecode.lanterna:lanterna:3.1.2")
```

Библиотека Lanterna позволяет управлять текстовым терминалом: позиционировать курсор, изменять цвета, читать ввод с клавиатуры и очищать экран. Важно помнить, что взаимодействие с терминалом должно происходить из одного потока выполнения, либо вы должны явно обеспечивать потокобезопасность (например, через синхронизацию или использование каналов корутин).

Ниже приведён минимальный шаблон инициализации терминала, совместимый с Windows, macOS и Linux:

```
val terminalFactory = DefaultTerminalFactory()
terminalFactory.setForceTextTerminal(false)
terminalFactory.setPreferTerminalEmulator(true)
val terminal = terminalFactory.createTerminal()
terminal.enterPrivateMode()
```

После выполнения программы обязательно вызовите:

```
terminal.exitPrivateMode()
```

Это восстановит исходное состояние терминала.

Примеры базовых операций:

- Установка позиции курсора и вывод строки:

  ```
  terminal.cursorPosition = TerminalPosition(10, 2)
  terminal.putString("Hello, World!")
  ```

- Очистка экрана:

  ```
  terminal.clearScreen()
  ```

- Чтение ввода:

  ```
  val key = terminal.readInput()
  ```

  Поле `key.keyType` содержит тип события (например, `KeyType.Escape`, `KeyType.Enter` и т.д.), а `key.character` — печатаемый символ (если применимо).

Все задания ниже должны быть реализованы с использованием корутин Kotlin (`kotlinx.coroutines`), Flow/Channel/StateFlow/SharedFlow по необходимости, и библиотеки Lanterna для консольного интерфейса. Интерфейс должен быть удобным, информативным и обновляться в реальном времени без миганий или артефактов.

Если какие-то символы сложно вывести, то вы можете заменить их на другие или рисовать с помощью ASCII ART.

1 **Многопользовательский Pomodoro-трекер.** Реализуйте приложение, позволяющее одновременно управлять несколькими Pomodoro-сессиями (например, для разных задач). Каждая сессия — отдельная корутина, генерирующая `Flow` или отправляющая события в `Channel` с текущим состоянием: *работа*, *отдых*, *пауза*, *остановлена*, а также оставшимся временем в секундах. Интерфейс должен отображать список всех сессий с возможностью запуска, паузы и остановки каждой из них. Поддержите стандартные интервалы (25/5 мин) и возможность их настройки.

2 **Мульти-секундомер для спорта.** Пользователь может создавать именованные секундомеры (например, «Иван», «Мария»). Каждый секундомер поддерживает: запуск, паузу, сброс и фиксацию кругов (laps). Все активные секундомеры отображаются в виде таблицы: имя, текущее время (с точностью до сотых секунды), количество кругов. Время должно обновляться в реальном времени. Каждый секундомер реализуется как независимая корутина, передающая обновления через `StateFlow` или `Channel`.

3 **Трекер параллельных задач с живым временем.** Пользователь добавляет задачи с названием и длительностью (в секундах). При запуске каждая задача становится корутиной, имитирующей выполнение: прогресс равномерно увеличивается от 0% до 100% за заданное время. Пользователь может приостанавливать, возобновлять или отменять любую задачу. Интерфейс отображает таблицу: название, статус (*ожидает*, *выполняется*, *приостановлена*, *завершена*), прогресс (в процентах и визуально, например, через прогресс-бар из символов), прошедшее время. Обновления передаются через `SharedFlow` или `StateFlow`.

4 **Чат с несколькими агентами.** Пользователь вводит команды вида `/погода`, `/курс`, `/шутка`. Каждая команда обрабатывается отдельной корутиной-агентом, имитирующей сетевой запрос (задержка 1–2 сек). Агенты возвращают текстовые ответы (например, «Сегодня +15°C», «USD = 92.5», «Почему программисты не ходят в лес? Там деревья!»). Все ответы поступают в общий поток и отображаются в хронологическом порядке в нижней части экрана (история сообщений). Верхняя часть содержит поле ввода. Взаимодействие между агентами и UI должно происходить через `Channel`.

5 **Анимированный Pomodoro-трекер с визуальным пульсом.** При активной сессии рядом с таймером пульсирует символ: $\bullet \to \circ \to \bullet$ с периодом 1 сек (500 мс на состояние). Анимация реализуется в отдельной корутине и синхронизируется с основной логикой таймера (например, при паузе анимация останавливается). Интерфейс отображает: текущий режим (работа/отдых), оставшееся время (мин:сек), анимированный индикатор. Все компоненты обмениваются данными через `Channel`.

6 **Таймер для интервальных тренировок (HIIT).** Пользователь задаёт последовательность фаз: название и длительность (например, «Бег — 40 сек», «Отдых — 20 сек»). После запуска приложение циклически проходит фазы. Каждая фаза управляется корутиной, которая отсчитывает время и отправляет обновления. Интерфейс отображает: текущую фазу, оставшееся время, номер цикла (если повторяется), общий прогресс (например, 2/5 циклов завершено). Поддерживается пауза и досрочное завершение. Обновления передаются через `Flow`.

7 **Симулятор нескольких печатающих машинок.** Пользователь создаёт машинки, задавая имя и текст. Каждая машинка посимвольно «печатает» свой текст в выделенной строке с задержкой (например, 100 мс на символ). Пользователь может остановить или возобновить печать любой машинки. Каждая машинка — отдельная корутина, отправляющая своё текущее состояние (напечатанная часть текста, позиция курсора) в UI через `Channel`. Интерфейс отображает все машинки одновременно.

8 **Менеджер фоновых загрузок (симуляция).** Пользователь добавляет загрузку, указывая имя, объём (например, 1000 условных единиц) и скорость (единиц/сек). Каждая загрузка — корутина, которая постепенно увеличивает прогресс. Интерфейс отображает таблицу: имя, прогресс (в процентах и визуально), текущая скорость, оставшееся время (расчётное). Любую загрузку можно отменить. Обновления передаются через `StateFlow` или `SharedFlow`.

9 **Живой лог-монитор с цветовой маркировкой.** Создайте три источника логов: «Сервер», «База данных», «Фронтенд». Каждый — отдельная корутина, генерирующая сообщения с разной частотой (например, сервер — каждые 500 мс, БД — каждые 1.2 сек, фронтенд — каждые 800 мс). Сообщения поступают в общий `BroadcastChannel` (или `SharedFlow`) и отображаются в UI как список последних 15 строк. Сообщения от разных источников визуально различаются: цветом текста или префиксом вида `[SRV]`, `[DB]`, `[FE]`. Пользователь может приостановить любой источник.

10 **Трекер времени для нескольких настольных игр за вечер.** Пользователь может запускать таймеры для разных игр («Шахматы», «Каркассон» и т.д.). Каждая игра может быть активна независимо от других. Для каждой игры отображается: название, статус (активна/-приостановлена), общее потраченное время. Каждая игра управляется отдельной корутиной,

передающей обновления через `Flow`. Поддержите одновременный запуск нескольких активных игр.

11 **Симулятор частиц огня.** Пользователь нажимает клавишу (например, `SPACE`), чтобы запустить новую частицу (ASCII-символ: `*`, `o`, `+`). Каждая частица движется вверх по экрану с индивидуальной скоростью (например, задержка 100–300 мс между шагами). При выходе за верхнюю границу частица удаляется. Пользователь может остановить все частицы. Каждая частица — отдельная корутина, отправляющая свою текущую позицию в UI через `Channel`. Интерфейс перерисовывает все частицы на каждом кадре.

12 **Симулятор нескольких аквариумов.** Каждый аквариум — отдельная строка, содержащая одну «рыбку» (например, `><>`). Рыбка плавает вправо-влево в пределах своей строки с постоянной скоростью. Пользователь может: создать аквариум (с именем), удалить существующий, приостановить движение конкретной рыбки. Каждая рыбка — отдельная корутина с циклической анимацией. Интерфейс отображает все аквариумы одновременно. Обновления позиций передаются через `Flow`.

13 **Таймер для языковой практики.** Пользователь запускает цикл из трёх фаз: «Говорение — 2 мин», «Письмо — 1 мин», «Отдых — 30 сек». После завершения фазы автоматически начинается следующая; после последней — цикл повторяется. Интерфейс отображает: текущую фазу, оставшееся время, номер цикла, общий прогресс. Поддерживается пауза и пропуск текущей фазы (переход к следующей). Каждая фаза управляется корутиной, смена фаз — через `Channel`.

14 **Мульти-таймер для готовки по рецепту.** Рецепт состоит из шагов, некоторые из которых помечены как параллельные. Например: «Варить пасту — 10 мин» (параллельно), «Готовить соус — 7 мин» (параллельно), «Смешать — 1 мин» (последовательно). При запуске все параллельные шаги начинают отсчёт одновременно. Интерфейс отображает каждый шаг с оставшимся временем, статусом (*ожидает*, *выполняется*, *завершён*) и индикатором параллельности. Каждый шаг — отдельная корутина, обновления передаются через `Flow`.

15 **Трекер времени в приложениях.** Пользователь может «включить» приложение (например, «Браузер», «IDE»). При включении запускается таймер использования; при выключении — останавливается, но накопленное время сохраняется. Интерфейс отображает список приложений с общим временем за текущую сессию и статусом (активно/неактивно). Каждое приложение управляется отдельной корутиной, которая отслеживает активное время и отправляет обновления через `StateFlow`.

16 **Симулятор нескольких будильников.** Пользователь создаёт будильник, указывая либо абсолютное время срабатывания (ЧЧ:ММ), либо интервал (например, каждые 5 минут). При срабатывании в UI появляется уведомление (например, мигающая строка). Пользователь может: «отложить» на 1 минуту или отключить. Каждый будильник — отдельная корутина, ожидающая события через `delay` или `withTimeout`. Уведомления передаются в UI через `Channel`.

17 **Трекер велопоездок с точками останова.** Поездка состоит из последовательности отрезков: «Дом → Парк — 15 мин», «Парк → Река — 20 мин» и т.д. После завершения отрезка поездка автоматически переходит в режим паузы. Пользователь может возобновить движение (начать следующий отрезок) или завершить поездку досрочно. Интерфейс отображает: текущий отрезок, общее пройденное время, статус (*в пути*, *пауза*, *завершена*). Каждый отрезок управляется корутиной.

18 **Живой генератор цитат.** Цитаты из заранее заданного списка появляются в UI одна за другой с интервалом (например, каждые 30 сек). Пользователь может: «сохранить» текущую цитату (она перемещается в отдельный блок истории), пропустить текущую или остановить генерацию. Генерация реализована в отдельной корутине, взаимодействие с UI — через `StateFlow`, содержащий текущую цитату и историю.

19 **Трекер медитаций с дыхательной анимацией.** Во время сессии в центре экрана отображается символ (например, ○), который плавно «расширяется» (вдох — 4 сек) и «сужается» (выдох — 6 сек). Анимация реализуется в отдельной корутине (через постепенное изменение символа или его окружения), а таймер сессии — в другой. Обе корутины синхронизированы через `Channel` (например, для паузы или остановки). Поддерживается пауза и остановка сессии.

20 **Pomodoro-трекер с историей и статистикой.** Пользователь запускает Pomodoro-сессии (работа/отдых). После завершения сессия сохраняется в историю (последние 10 записей с временем завершения и типом). Интерфейс разделён на две области: верхняя — активные сессии (с управлением), нижняя — история (только просмотр). Активные сессии реализованы как корутины, завершённые — хранятся в `SharedFlow` или `MutableStateFlow`.

21 **Симулятор мигающих огней на карте.** Пользователь добавляет «огонёк» в заданную строку экрана (например, вводит номер строки). Каждый огонёк мигает с индивидуальной частотой: `*` → пробел → `*` с заданным периодом (например, 800 мс). Каждый — отдельная корутина. Пользователь может удалить любой огонёк. Весь экран перерисовывается на основе актуальных состояний, передаваемых через `Channel`.

22 **Таймер для записи подкаста с главами.** Пользователь запускает запись — идёт общий таймер. В любой момент можно нажать клавишу (например, `C`), чтобы добавить «главу» — фиксируется временная метка (мин:сек). Интерфейс отображает: общий таймер и список глав с их временными метками. Запись (таймер) и обработка глав — разные корутины, взаимодействие через `Channel`.

23 **Трекер времени на улице.** Пользователь запускает «прогулку» — начинается отсчёт времени. При входе в помещение (магазин, дом) можно поставить на паузу; при выходе — возобновить. Можно вести несколько таких сессий в течение дня (например, «Утро», «Вечер»). Каждая сессия — отдельная корутина с состоянием (*активна*/*пауза*) и накопленным временем. UI отображает все сессии с возможностью управления.

24 **Симулятор нескольких сердец с разным ритмом.** Пользователь создаёт «сердце», задавая частоту (ударов в минуту, например, 60, 72, 80). Каждое сердце отображается как символ ♡, который мигает в заданном ритме (например, при 60 уд/мин — вспышка каждую секунду). Каждое — отдельная корутина с `delay`, рассчитанным по формуле delay = 60 000/частота мс. Пользователь может остановить любое сердце. Анимация обновляется через `Flow`.

25 **Таймер хода для настольных игр.** Пользователь задаёт список игроков и время на ход (например, 60 сек). После старта таймер идёт для текущего игрока. По истечении времени — звуковое или визуальное уведомление и автоматический переход к следующему игроку. Пользователь может продлить ход (добавить 30 сек) или пропустить игрока. Каждый игрок управляется логикой в корутине, обновления состояния (текущий игрок, оставшееся время) передаются через `StateFlow`.

26 **Мульти-таймер для йоги с визуальной подсказкой.** Пользователь выбирает последовательность поз (например, из списка: «Гора», «Дерево», «Собака мордой вниз»). Каждая поза отображается как упрощённая ASCII-схема (например, `/|\` для «горы»). Для каждой позы идёт таймер (например, 30 сек). После окончания можно: автоматически перейти к следующей или ждать подтверждения пользователя (нажатие клавиши). Каждая поза — корутина, UI обновляется по `Flow`.

27 **Консольный космос: спутники на орбите.** Пользователь добавляет спутник, задавая радиус орбиты (в символах) и угловую скорость (градусов/сек). Каждый спутник движется по круговой траектории в пределах ASCII-экрана с использованием упрощённых тригонометрических расчётов (например, `x = centerX + radius * cos(angle)`, `y = centerY + radius * sin(angle)`). Каждый спутник — отдельная корутина с `delay`, обновляющая свою позицию. Пользователь может удалить спутник. Положения всех спутников передаются в UI через `Channel` для перерисовки.

28 **Менеджер Pomodoro-сессий для команды.** Пользователь вводит имена участников (например, «Анна», «Борис»). Для каждого можно запустить независимую Pomodoro-сессию. Интерфейс отображает таблицу: имя, статус (*работа*/*отдых*/*пауза*), оставшееся время. Управление осуществляется клавишами: стрелки — выбор участника, `Enter` — запуск/пауза, `Del` — остановка. Каждая сессия — корутина, обновления — через `SharedFlow`.

3 Разработка на платформе Android

3.1 Семинар «Простейшие программы для Android» (3 часа)

Разработайте программу, работающую под управлением Android с использованием Jetpack Compose. Проверьте, что программа корректно работает с различными размерами экрана, а также при повороте экрана. Необходимо проверять корректность исходных данных и сообщать о проблемах.

Варианты для группы 411

1 Программа-календарь (по году, месяцу, числу – день недели) с удобным вводом (с выпадающим спис- ком). В работе запрещено использовать встроенные классы для даты и ввода даты.

2 Программа нахождения решения неравенства $ax + b$*знак*0 (удобный ввод с выпадающим списком знака неравенства).

3 Программа нахождения решения неравенства $ax^2 + bx + c$*знак*0 (удобный ввод с выпадающим списком знака неравенства).

4 Программа учета расходов и доходов по месяцам: в элементы управления вводятся доходы за январь- декабрь и расходы за январь-декабрь, программа выводит прибыль за январь-декабоь, общую сумму расходов, доходов и прибыли за год. Интерфейс должен напоминать таблицу.

5 Программа поиска обратной матрицы для матрицы 3×3. Запрещено использовать специальные библиотеки для матриц.

6 Программа нахождения центра и радиуса окружности по координатам трех точек, принадлежащих ей.

7 Напишите игру «Пятнашки» (не следует делать идеальный интерфейс)

8 Программа решения системы уравнений

$$\begin{cases} ax + by + c &= 0 \\ dx + ey + f &= 0 \end{cases}$$

где все представленные числа комплексные. Программа должна находить ответ в случае, если решение единственно, в противном случае выдавать диагностическое сообщение о невозможности решения. Ипользовать специальные классы для комплексных чисел, если они не реализованы вами, запрещено

9 Программа перевода даты (год, месяц, число) из григорианского календаря в юлианский и обратно с удобным вводом (с выпадающим списком). В работе запрещено использовать встроенные классы для даты и ввода даты.

10 Программа нахождения площади меньшей из частей круга, порождаемых пересечением круга и пря- мой, по координатам центра и радиусу круга и по координатам двух точек прямой.

11 Программа деления двух многочленов: не более, чем четвертой степени, на не более, чем второй.

12 Программа-игра чет-нечет, где человек задумывает, а программа отгадывает четность/нечетность загаданных чисел. Программа должна анализировать загадываемые человек числа с целью предсказания, выявляя закономерности длиной от 1 до 10 чисел: в случае если человек регулярно загадывает одинаковую последовательность чисел такой длины, то программа с некоторого момента должна их угадывать всегда. Если человек чаще всего после некой последовательности длины от 1 до 10 чисел загадывает чет (или нечет), то программа данное явление также должна выявлять.

Не следует придумывать слишком сложный алгоритм. Это не ИИ (в современном смысле слова).

13 Программа решения уравнения $ax^2 + bx + c = 0$, где все представленные числа – комплексные. Запрещено использовать классы для представления комплексных чисел, если они не реализованы вами.

14 Программа-калькулятор для матриц 2×2. Пользователю доступно задание матриц 2×2 и кнопки $+$, $-$, $*$, $/$, $=$, C

15 Программа перевода чисел в/из римской системы счисления

16 Программа перевода числа из одной системы счисления в другую (от 2 до 36) с удобным вводом: пользователь указывает исходную и целевую системы через выпадающие списки и вводит число. Запрещено использовать функции библиотеки для перевода из одной системы счисления в другую.

17 Программа нахождения точки пересечения двух отрезков на плоскости по координатам их концов. В случае отсутствия пересечения — вывод соответствующего сообщения.

18 Программа шифрования и дешифрования текста шифром Цезаря с возможностью выбора языка (русский/английский) и сдвига через интерфейс. Программа должна корректно обрабатывать регистр и неалфавитные символы.

19 Пользователь выбирает тип операции (`+`, `-`, `x`, `÷`), диапазон чисел и количество вопросов. Программа генерирует примеры, принимает ответы, ведёт счёт правильных/неправильных и в конце выводит статистику. Деление должно быть только нацело (делимое кратно делителю).

20 **Программа нахождения координат точки пересечения медиан треугольника (центроида)**. Пользователь вводит координаты трёх вершин треугольника. Программа вычисляет и выводит координаты центроида как среднее арифметическое координат вершин.

21 **Программа нахождения длины общей хорды двух пересекающихся окружностей**. Пользователь вводит центры и радиусы двух окружностей, которые гарантированно пересекаются в двух точках. Программа вычисляет и выводит длину их общей хорды (через геометрические соотношения в треугольниках, образованных центрами и точками пересечения).

22 **Программа нахождения координат центра и радиуса вписанной окружности треугольника.** Пользователь вводит координаты трёх вершин невырожденного треугольника. Программа вычисляет и выводит координаты центра вписанной окружности и её радиус.

23 **Программа нахождения расстояния между двумя непересекающимися отрезками на плоскости.** Пользователь вводит координаты концов двух отрезков. Программа определяет, пересекаются ли они; если да — выводит 0; если нет — вычисляет и выводит минимальное расстояние между ними (минимальное расстояние между точками, принадлежащими разным отрезкам).

24 **Программа нахождения уравнения общей касательной к двум окружностям.** Пользователь вводит центры и радиусы двух окружностей. Программа выводит коэффициенты уравнения прямой Ax+By+C=0 , являющейся одной из таких касательных (любой) или сообщает об отсутствии таковой.

25 **Программа нахождения площади пересечения двух прямоугольников со сторонами, параллельными осям координат.** Пользователь вводит координаты противоположных углов двух таких прямоугольников. Программа вычисляет площадь их пересечения (0, если не пересекаются).

26 **Программа нахождения координат точки пересечения высот треугольника.** Пользователь вводит координаты трёх вершин невырожденного треугольника. Программа вычисляет и выводит координаты точки пересечения трёх высот.

Варианты для группы 412

1. Программа решения квадратного уравнения
2. Программа решения неравенства вида $ax + b > 0$
3. Программа решения неравенства вида $ax + b < 0$
4. Программа решения неравенства вида $ax + b \geqslant 0$
5. Программа решения неравенства вида $ax + b \leqslant 0$

6. Программа поиска дня недели по числу и месяцу в текущем году
7. Программа перевода числа из 10-ой в 16-ую, 8-ую и 2-ую систем.
8. Программа поиска времени, когда окончится интервал. Дано: часы и минуты начала интервала и количество минут, сколько он идет. Результат: часы и минуты окончания интервала.
9. Программа поиска обратной матрицы для матрицы 3×3.
10. Программа поиска длины интервала. Дано: часы и минуты начала интервала и часы и минуты конца интервала. Результат: количество минут в интервале.
11. Программа умножения и деления двух комплексных чисел.
12. Программа нахождения площади треугольника по координатам вершин.
13. Программа нахождения углов треугольника по координатам вершин (проще всего это сделать по теореме косинусов).
14. Программа перевода числа из 16-ой, 8-ой и 2-ой системы в 10-ую систему счисления.
15. Программа нахождения количества денег на вкладе после окончания его срока по начальному взносу, проценту и срока в годах.
16. Программа нахождения степени комплексного числа. Исходные данные: действительная, мнимая часть числа и степень. Результат: действительная и мнимая часть результата.
17. Программа умножения и деления чисел, представленных в виде обыкновенных дробей (состоящих из целой части, числителя и знаменателя). Не забудьте выполнить сокращение дроби и приведение ее к правильному виду.
18. Программа сложения и вычитания чисел, представленных в виде обыкновенных дробей (состоящих из целой части, числителя и знаменателя). Не забудьте выполнить сокращение дроби и приведение ее к правильному виду.
19. Программа определения по дате (число и месяц) знака зодиака.
20. Программа определения по обыкновенной дроби (числителю и знаменателю) периода десятичной дроби.
21. Программа перевода комплесного числа из обычной формы в тригонометрическую и наоборот.
22. Программа-игра Баше. При реализации этого задания не требуется ничего рисовать, вся информация вводится и выводится в виде чисел в обычные элементы управления.
23. Программа разложения числа на простые множители.
24. Программа нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел.
25. Программа-тест по предмету «Разработка мобильных приложений». Создайте программу-тест из 10 вопросов с выбором вариантов ответов и показом результатов прохождения теста.
26. Программа генерации всех перестановок заданного набора различных символов (например, букв или цифр). Исходные данные: строка из k различных символов ($1 \leq k \leq 8$). Результат: список всех возможных перестановок в лексикографическом порядке.
27. Программа нахождения всех целых решений (пар (x, y)) линейного диофантова уравнения вида $ax + by = c$, где a, b, c — целые числа, в заданном диапазоне (например, $-100 \leq x, y \leq 100$). Исходные данные: коэффициенты a, b, c и границы диапазона поиска. Результат: список всех целочисленных пар (x, y), удовлетворяющих уравнению в указанном диапазоне, либо сообщение об отсутствии решений.
28. Программа решения системы двух линейных уравнений с двумя неизвестными методом Крамера. Дано: коэффициенты уравнений $a_1x + b_1y = c_1$, $a_2x + b_2y = c_2$. Результат: значения x и y, либо сообщение о том, что система не имеет единственного решения (определитель равен нулю).

3.2 Семинар «Автоматизация тестирования» (3 часа)

Разработайте автоматические тесты программы, разработанной на семинаре «Простейшие программы для Android». С этой целью: а) выделите вычислительные функции в отдельный класс; б) разработайте UNIT-тесты для данного класса; в) разработайте UI-тест для приложения.

Используйте возможности фреймворка при реализации данной работы

3.3 Семинар «Работа со списками» (6 часов)

В данной работе вам необходимо взять код с семинара «Объектно-ориентированное программирование» и разделить его на три части:

- модуль **core** (наведите курсор мыши на проект, нажмите правую кнопку мыши и создайте модуль Java/Kotlin library);
- модуль **console** (консольное приложение, которое работает как раньше и использует модуль core);
- модуль **app** (он создается мастером по умолчанию; Android-приложение, которое использует модуль core).

В последних двух модулях подключение core делается посредством строки:

```
implementation(project(":core"))
```

Реализуйте функциональность консольного приложения в Android-приложении, не забыв выделить отдельно View, ViewModel, а также Model, которая преимущественно реализуется модулем core. Используйте Navigation для переходов между экранами.

Дополните Android-приложение полной CRUD-функциональностью (создание, просмотр, обновление и удаление сущностей согласно вашему варианту).

3.4 Семинары «Использование библиотек для поддержки Dependency Injection и Clean Architecture» (6 часов)

Используйте Hilt для внедрения Dependency Injection в проект, созданный на прошлом семинаре. Приведите проект в соответствие принципам Clean Architecture (выделить View, ViewModel, UseCases, Model, Repositories, DataSources, DI).

Дополните Unit-тесты и UI-тесты для полного покрытия.

Для хранения информации в Android-приложении (и только в нем) используйте ROOM (соответственно у вас будет три репозитория – тот, что был реализован в консоли; ROOM-репозиторий и Mock).

3.5 Семинар «Работа с внешними сервисами RestAPI» (6 часов)

Общие требования ко всем заданиям

- Приложение должно быть реализовано в соответствии с принципами **чистой архитектуры** (Clean Architecture) и использовать паттерн **MVVM**.
- Для внедрения зависимостей обязателен **Hilt**.
- Для сетевых запросов используется библиотека **Retrofit** (в фоне, через `Coroutine`).
- Все сетевые ответы должны **кешироваться локально** (с помощью Room), чтобы обеспечивать работу приложения при отсутствии интернет-соединения.
- Обязательна реализация как **Unit-тестов** (для Repository, UseCase и т.д.), так и **UI-тестов**.
- Используйте Navigation для переходов между экранами.

Задания Реализуйте клиент для одного из следующих REST-API. Каждое задание предполагает:

- наличие хотя бы одного параметризованного запроса;
- отображение результата в UI;
- сохранение истории или кэша для офлайн-доступа.

Если сайт, указанный в вашем варианте, не работает или не доступен, замените его на другой аналогичный сайт, но при этом сайты-заменители у разных студентов совпадать не должны (выбирайте сайты-заменители полностью независимо друг от друга).

1. **RestCountries API** — поиск стран по языку (`lang=ru`). https://restcountries.com/
2. **RestCountries API** — поиск стран по валюте (`currency=USD`).
   https://restcountries.com/
3. **RestCountries API** — поиск стран по региону (`region=Europe`).
   https://restcountries.com/
4. **RestCountries API** — поиск стран по субрегиону (`subregion=Eastern Europe`).
   https://restcountries.com/
5. **RestCountries API** — поиск стран по региональному блоку (`regionalbloc=EU`).
   https://restcountries.com/
6. **Open Astronomy Catalog API** — поиск астрономических объектов в радиусе (в угловых секундах) от заданных координат RA/Dec.
   https://api.astrocats.space/
7. **Met.no Air Quality API** — получение данных о качестве воздуха по ID станции (например, `NO0057A`), отображение AQI, PM10, O3.
   https://api.met.no/weatherapi/airqualityforecast/0.1/
8. **Met.no Location Forecast API** — прогноз погоды (температура, осадки, ветер) по координатам из фиксированного списка.
   https://api.met.no/weatherapi/locationforecast/2.0/
9. **TVMaze API** — поиск телешоу по названию.
   https://www.tvmaze.com/api
10. **TVMaze API** — получение списка персонажей по ID шоу.
    https://www.tvmaze.com/api
11. **TVMaze API** — расписание эфиров по дате и стране.
    https://www.tvmaze.com/api
12. **Nager.Date API** — получение списка праздников по году и коду страны (например, `US`).
    https://date.nager.at/api/v3/

13. **Math.js API** — упрощение математического выражения (например, `2x + 3x`).
    https://api.mathjs.org/

14. **Tronald Dump API** — поиск цитат по ключевому фрагменту текста.
    https://api.tronalddump.io/

15. **Cat Facts API** — получение случайного факта о кошках с сохранением истории.
    https://catfact.ninja/

16. **AnimeChan API** — получение цитаты из указанного аниме.
    https://animechan.io/

17. **AnimeChan API** — получение цитат по имени персонажа.
    https://animechan.io/

18. **Open Library API** — поиск книг по тематике (ключевому слову).
    https://openlibrary.org/

19. **Open Library API** — список произведений по имени автора.
    https://openlibrary.org/

20. **Open Library API** — поиск книг по фрагменту названия.
    https://openlibrary.org/

21. **PoetryDB API** — получение стихотворений по имени автора (с начальными строками).
    https://poetrydb.org/

22. **Currency API** — получение курсов обмена относительно заданной валюты (например, `usd`).
    https://cdn.jsdelivr.net/npm/@fawazahmed0/currency-api@1/

23. **Nationalize.io API** — прогноз национальности по имени с вероятностями.
    https://api.nationalize.io/

24. **BoredAPI** — получение предложения занятия по типу и числу участников, с сохранением истории.
    https://bored-api.appbrewery.com/

25. **Public APIs API** — поиск публичных API по ключевому слову в названии.
    https://api.publicapis.org/

3.6 Последние семинары (две недели)

Последние семинары предназначены для подведения итогов: сдачи всех работ, проведения финального опроса и подведения итогов курса.

4 Проекты как замена заданиям семинаров

Студент по согласованию с преподавателем может реализовывать мобильное приложение, являющееся крупным проектом (его объём должен быть не меньше суммарного объёма заданий семинаров).

Проект может реализовываться на Android (Kotlin), iOS (Swift) или Flutter (Dart).

В ходе реализации проекта нужно использовать следующие принципы: покрытие unit- и UI-тестами, принципы Clean Architecture, MVVM, DI с использованием стандартных библиотек, SOLID, функциональный подход в программировании.

Используйте стандартные библиотеки Navigation или их аналоги на других платформах.

Код должен соответствовать стандартам кодирования, принятым в крупных организациях, занимающихся разработкой программного обеспечения.

5 Список литературы

1. Официальная документация по языку Kotlin: https://kotlinlang.org/docs/
2. Официальная документация по платформе Android: https://developer.android.com/docs
3. Быстрое введение в Kotlin от авторов языка: https://stepik.org/course/4222