Unfortunately, you can't participate in this project

Subscribe

Intensive Show all

AP1-Kt-T01

AP1-Kt-T02

AP1-Kt-P01

AP1-Kt-T03

Task

Project 02 — Kotlin_Bootcamp

Резюме: в этом проекте ты научишься применять ООП/процедурный/мультипарадигмальный подход в **Kotlin**, а также напишешь код, придерживаясь функциональной парадигмы.

💡 Нажми сюда, **чтобы поделиться с нами обратной связью на этот проект**. Это анонимно и поможет нашей команде сделать обучение лучше. Рекомендуем заполнить опрос сразу после выполнения проекта.

Содержание

Chapter I

Инструкция

1. На протяжении всего курса тебя будет сопровождать чувство неопределенности и острого дефицита информации — это нормально. Не забывай, что информация в репозитории и Google всегда с тобой. Как и пиры, и Rocket.Chat. Общайся. Ищи. Опирайся на здравый смысл. Не бойся ошибиться.
2. Будь внимателен к источникам информации. Проверяй. Думай. Анализируй. Сравнивай.
3. Внимательно читай задания. Перечитай несколько раз.
4. Читать примеры тоже лучше внимательно. В них может быть что-то, что не указано в явном виде в самом задании.
5. Тебе могут встретиться несоответствия, когда что-то новое в условиях задачи или примере противоречит уже известному. Если встретилось такое — попробуй разобраться. Если не получилось — запиши вопрос в открытые вопросы и выясни в процессе работы. Не оставляй открытые вопросы неразрешенными.
6. Если задание кажется непонятным или невыполнимым — так только кажется. Попробуй его декомпозировать. Скорее всего, отдельные части станут понятными.
7. На пути тебе встретятся самые разные задания. Те, что помечены звездочкой (*) — подходят для более дотошных. Они повышенной сложности и необязательны к выполнению. Но если ты их сделаешь, то получишь дополнительный опыт и знания.
8. Не пытайся обмануть систему и окружающих. В первую очередь ты обманешь себя.
9. Есть вопрос? Спроси соседа справа. Если это не помогло — соседа слева.

10. Когда пользуешься помощью — всегда разбирайся до конца: почему, как и зачем. Иначе помощь не будет иметь смысла.
11. Всегда делай push только в ветку develop! Ветка master будет проигнорирована. Работай в директории src.
12. В твоей директории не должно быть иных файлов, кроме тех, что обозначены в заданиях.

Chapter II

Общая информация

Темы для изучения:

- ООП/процедурный/мультипарадигмальный подход в Kotlin;
- Отличия от C и C++;
- Функциональная парадигма;
- Асинхронное/параллельное программирование;

Chapter III

Проект: Pet Info

Рассматриваются ООП/процедурные/мультипарадигмальные подходы в Kotlin, написание кода в соответствии с функциональной парадигмой, а также асинхронное/параллельное программирование при помощи проекта, который представляет собой набор модулей для отображения информации о питомцах.

Внимание! Каждую задачу оформляй в качестве отдельного проекта. Например, `T01/src/exercise0` , `T01/src/exercise1` , ... , `T01/src/exerciseN-1` , где *N* количество задач. Если предыдущее задание необходимо для следующего, просто скопируй предыдущий проект в директорию следующего и продолжай разработку в нем.

Задание 1. Список питомцев

Разработай модуль, который составляет список питомцев и выводит информацию о каждом питомце.

- Создай абстрактный класс Animal с конструктором, который принимает два private поля: строка имени питомца, целочисленный возраст.
- Создай класс Dog, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Dog, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.

- Переопредели метод toString() в классе Dog, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца]».
- Создай класс Cat, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Cat, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Cat, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца]».
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна вывести информацию о каждом питомце.

Входные данные	Выходные данные
3 dog Snowball 12 dog Snowball2 10 dog Snowball3 9	Dog name = Snowball, age = 12 Dog name = Snowball2, age = 10 Dog name = Snowball3, age = 9
3 dog Snowball 12 cat Kitty 10 dog Balloon 9	Dog name = Snowball, age = 12 Cat name = Kitty, age = 10 Dog name = Balloon, age = 9
3 hamster cat Kitty -10	Incorrect input. Unsupported pet type Incorrect input. Age <=0 Cat name = Fura, age = 9

Входные данные	Выходные данные
cat Fura 9	

Задание 2. Определение количества корма питомцу

Разработай модуль, который по типу питомца определяет, сколько грамм корма ему необходимо на 1 порцию.

- Создай абстрактный класс Animal с конструктором, который принимает три private поля: строка имени питомца, целочисленный возраст, вещественный вес питомца.
- Объяви в абстрактном классе Animal метод getFeedInfoKg(), который возвращает вещественное количество корма.
- Создай класс Dog, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Dog, который принимает три параметра: строка имени питомца, целочисленный возраст, вещественный вес питомца и передаёт их в конструктор базового класса.
- Реализуй метод getFeedInfoKg() в классе Dog, который вычисляет количество необходимого корма по следующей формуле: кол-во корма = масса питомца * 0.3.
- Переопредели метод toString() в классе Dog, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца], mass = [масса_питомца], feed = [порция_питомца]».
- Создай класс Cat, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Cat, который принимает три параметра: строка имени питомца, целочисленный возраст, вещественный вес питомца и передаёт их в конструктор базового класса.
- Реализуй метод getFeedInfoKg() в классе Cat, который вычисляет количество необходимого корма по следующей формуле: кол-во корма = масса питомца * 0.1.
- Переопредели метод toString() в классе Cat, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца], mass = [масса_питомца], feed = [порция_питомца]».
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.

- Если ввели отрицательную или нулевую массу, то программа выводит: «Incorrect input. Mass <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна вывести информацию о каждом питомце и количество корма, которое ему необходимо.

Входные данные	Выходные данные
3 dog Snowball 12 5.0 dog Snowball2 10 10.0 dog Snowball3 9 9.0	Dog name = Snowball, age = 12, mass = 5.00, feed = 1.50 Dog name = Snowball2, age = 10, mass = 10.00, feed = 3.00 Dog name = Snowball3, age = 9, mass = 9.00, feed = 2.70
3 dog Snowball 12 5.0 cat Kitty 10 10.0 dog Balloon 9 9.0	Dog name = Snowball, age = 12, mass = 5.00, feed = 1.50 Cat name = Kitty, age = 10, mass = 10.00, feed = 1.00 Dog name = Balloon, age = 9, mass = 9.00, feed = 2.70
4 hamster cat Kitty -10 dog Balloon 9 -9 cat Fura 9 12.5	Incorrect input. Unsupported pet type Incorrect input. Age <=0 Incorrect input. Mass <= 0 Cat name = Fura, age = 9, mass = 12.50, feed = 1.25

Задание 3. Списки травоядных и всеядных питомцев

Разработай модуль, который сначала выводит только травоядных животных, а затем только всеядных животных.

- Создай абстрактный класс Animal с конструктором, который принимает два private поля: строка имени питомца, целочисленный возраст.
- Создай интерфейс Herbivore.
- Объяви в интерфейсе Herbivore метод chill(), который возвращает строку.
- Создай интерфейс Omnivore.
- Объяви в интерфейсе Omnivore метод hunt(), который возвращает строку.
- Создай класс Dog, который наследуется от абстрактного класса Animal и реализует интерфейс Omnivore.
- Реализуй конструктор для класса Dog, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- В классе Dog реализуй метод hunt() таким образом, чтобы формировалась следующая строка: «I can hunt for robbers».
- В классе Dog переопредели метод toString() таким образом, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца]. » + hunt().
- Создай класс Cat, который наследуется от абстрактного класса Animal и реализует интерфейс Omnivore.
- Реализуй конструктор для класса Cat, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- В классе Cat реализуй метод hunt() таким образом, чтобы формировалась следующая строка: «I can hunt for mice».
- В классе Cat переопредели метод toString() таким образом, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца]. » + hunt().
- Создай класс Hamster, который наследуется от абстрактного класса Animal и реализует интерфейс Herbivore.
- Реализуй конструктор для класса Hamster, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- В классе Hamster реализуй метод chill() таким образом, чтобы формировалась следующая строка: «I can chill for 8 hours».
- В классе Hamster переопредели метод toString() таким образом, чтобы формировалась следующая строка: «Hamster name = [имя_питомца], age = [возраст_питомца]. » + chill()
- Создай класс GuineaPig, который наследуется от абстрактного класса Animal и реализует интерфейс Herbivore.

- Реализуй конструктор для класса GuineaPig, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- В классе GuineaPig реализуй метод chill() таким образом, чтобы формировалась следующая строка: «I can chill for 12 hours».
- В классе GuineaPig переопредели метод toString() таким образом, чтобы формировалась следующая строка: «GuineaPig name = [имя_питомца], age = [возраст_питомца]. » + chill()
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat/hamster/guinea.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна сначала вывести информацию обо всех травоядных питомцах, а потом обо всех всеядных питомцах.

Входные данные	Выходные данные
4 dog Snowball 12 guinea Piggy 5 cat Snowball 9 hamster Wave 2	GuineaPig name = Piggy, age = 5. I can chill for 12 hours Hamster name = Wave, age = 2. I can chill for 8 hours Dog name = Snowball, age = 12. I can hunt for robbers Cat name = Snowball, age = 9. I can hunt for mice
2 dog Snowball 12 cat Kitty 10	Dog name = Snowball, age = 12. I can hunt for robbers Cat name = Kitty, age = 10. I can hunt for mice
3 turtle cat Kitty	Incorrect input. Unsupported pet type Incorrect input. Age <=0 GuineaPig name = Piggy. I can chill for 12 hours

Входные данные	Выходные данные
-10 guinea Piggy 3	

Задание 4. Увеличение возраста конкретных питомцев в функциональной парадигме

Разработай модуль, который увеличивает возраст питомцев старше 10 лет, придерживаясь функциональной парадигме.

- Создай абстрактный класс Animal с конструктором, который принимает два private поля: строка имени питомца, целочисленный возраст.
- Создай класс Dog, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Dog, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Dog, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца]».
- Создай класс Cat, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Cat, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Cat, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца]».
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна увеличить возраст всех питомцев старше 10 лет на 1 год.
- Программа должна вывести информацию о каждом питомце.
- Программа должна придерживаться функциональной парадигме.
- Операции увеличения возраста и вывода питомцев должны выполняться при помощи Sequences.

- Запрещается использовать любые операторы повторения.

Входные данные	Выходные данные
3 dog Snowball 12 dog Snowball2 8 dog Snowball3 10	Dog name = Snowball, age = 13 Dog name = Snowball2, age = 8 Dog name = Snowball3, age = 10
3 dog Snowball 8 cat Kitty 9 dog Balloon 9	Dog name = Snowball, age = 8 Cat name = Kitty, age = 9 Dog name = Balloon, age = 9
4 hamster cat Kitty -10 dog Balloon 10 cat Fura 9	Incorrect input. Unsupported pet type Incorrect input. Age <=0 Dog name = Balloon, age = 10 Cat name = Fura, age = 9

Задание 5. Отслеживание прогулок питомцев

Разработай модуль, который отслеживает время начала и завершения прогулки питомца.

- Создай абстрактный класс Animal с конструктором, который принимает два private поля: строка имени питомца, целочисленный возраст.
- Объяви в абстрактном классе Animal метод, который возвращает вещественное время goToWalk().
- Создай класс Dog, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Dog, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.

- Переопредели метод toString() в классе Dog, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца]».
- Переопредели метод goToWalk() в классе Dog следующим образом: метод вычисляет время прогулки в миллисекундах, вызывает TimeUnit.MILLISECONDS.sleep() для вычисленного времени, возвращает вычисленное время.
- Создай класс Cat, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Cat, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Cat, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца]».
- Переопредели метод goToWalk() в классе Dog следующим образом: метод вычисляет время прогулки в миллисекундах, вызывает TimeUnit.MILLISECONDS.sleep() для вычисленного времени, возвращает вычисленное время.
- Вычисление времени прогулки для класса Dog выполняется по следующей формуле: [время_прогулки] = [возраст_питомца] * 500.
- Вычисление времени прогулки для класса Cat выполняется по следующей формуле: [время_прогулки] = [возраст_питомца] * 250.
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна вызвать метод goToWalk() у каждого питомца.
- Каждый вызов метода goToWalk() должен выполняться асинхронно в отдельном потоке.
- Программа должна дождаться выполнения всех вызовов метода goToWalk(), перед тем как завершиться.
- При завершении прогулки, программа должна выводить на одной строке в консоль следующую информацию: информация о питомце, время старта прогулки в секундах, время конца прогулки в секундах
- Время старта прогулки и время конца прогулки должны вычисляться относительно времени старта программы.
- Разница времени старта прогулки у питомцев относительно друг друга не должна быть больше 1 секунды.

Входные данные	Выходные данные
3 dog Snowball 12 dog Snowball2 8 dog Snowball3 10	Dog name = Snowball2, age = 8, start time = 0.20, end time = 4.20 Dog name = Snowball3, age = 10, start time = 0.30, end time = 5.30 Dog name = Snowball, age = 12, start time = 0.10, end time = 6.10
3 dog Snowball 8 cat Kitty 9 dog Balloon 9	Cat name = Kitty, age = 9, start time = 0.20, end time = 2.45 Dog name = Snowball, age = 8, start time = 0.10, end time = 4.10 Dog name = Balloon, age = 9, start time = 0.30, end time = 4.80
4 hamster cat Kitty -10 dog Balloon 10 cat Fura 9	Incorrect input. Unsupported pet type Incorrect input. Age <=0 Cat name = Fura, age = 9, start time = 0.20, end time = 2.45 Dog name = Balloon, age = 11, start time = 0.10, end time = 5.60

Задание 6. Итератор питомцев

Разработай модуль, который реализует итератор питомцев.

- Создай абстрактный класс Animal с конструктором, который принимает два private поля: строка имени питомца, целочисленный возраст.
- Создай класс Dog, который наследуется от абстрактного класса Animal.
- Реализуй конструктор для класса Dog, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Dog, чтобы формировалась следующая строка: «Dog name = [имя_питомца], age = [возраст_питомца]».
- Создай класс Cat, который наследуется от абстрактного класса Animal.

- Реализуй конструктор для класса Cat, который принимает два параметра: строка имени питомца, целочисленный возраст и передаёт их в конструктор базового класса.
- Переопредели метод toString() в классе Cat, чтобы формировалась следующая строка: «Cat name = [имя_питомца], age = [возраст_питомца]».
- Создай интерфейс BaseIterator.
- Объяви в интерфейсе BaseIterator метод next(), который возвращает элемент типа T.
- Объяви в интерфейсе BaseIterator метод hasNext(), который возвращает boolean.
- Объяви в интерфейсе BaseIterator метод reset(), который ничего не возвращает.
- Создай класс AnimalIterator, который реализует интерфейс BaseIterator.
- В классе AnimalIterator объяви 2 private поля: список животных, целочисленный индекс текущего элемента списка.
- Реализуй конструктор для класса AnimalIterator, который принимает список животных.
- В классе AnimalIterator реализуй метод next() следующим образом: метод возвращает текущий элемент списка животных, а затем увеличивает индекс текущего элемента списка на единичку.
- В классе AnimalIterator реализуй метод hasNext() следующим образом: метод возвращает true, если индекс текущего элемента списка меньше количества элементов списка животных, иначе false.
- В классе AnimalIterator реализуй метод reset() следующим образом: метод сбрасывает значение индекса текущего элемента списка.
- Программа считывает количество питомцев.
- Программа считывает тип вводимого питомца: dog/cat.
- Каждый питомец добавляется в общий список pets.
- Если ввели неправильный тип питомца, то программа выводит: «Incorrect input. Unsupported pet type» и переходит к следующему вводу.
- Если ввели отрицательный или нулевой возраст, то программа выводит: «Incorrect input. Age <= 0» и переходит к следующему вводу.
- Программа не завершается с ошибкой при некорректных входных данных. Она выводит: «Couldn't parse a number. Please, try again» и повторяет попытку ввода.
- Программа должна вывести информацию о каждом питомце.
- Программа должна пройтись по списку питомцев pets с помощью итератора AnimalIterator.

Входные данные	Выходные данные
3 dog Snowball 12	Dog name = Snowball, age = 12 Dog name = Snowball2, age = 10 Dog name = Snowball3, age = 9

Входные данные	Выходные данные
dog Snowball2 10 dog Snowball3 9	
3 dog Snowball 12 cat Kitty 10 dog Balloon 9	Dog name = Snowball, age = 12 Cat name = Kitty, age = 10 Dog name = Balloon, age = 9
3 hamster cat Kitty -10 cat Fura 9	Incorrect input. Unsupported pet type Incorrect input. Age <=0 Cat name = Fura, age = 9