#### 5.18 Домашняя работа

Задача

**Цель задания**

* поработать с классами, методами, свойствами;
* создать объекты классов, обратиться к их свойствам и вызвать их методы;
* поработать с наследованием, абстрактными классами, интерфейсами и анонимными классами.

**Что нужно сделать**

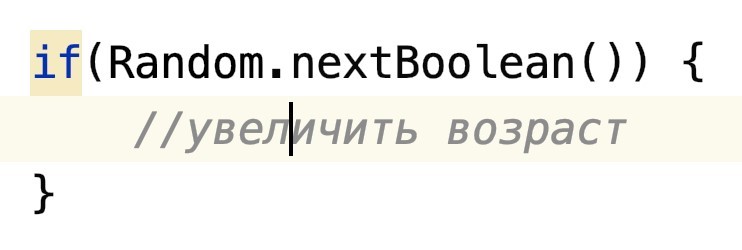
1. Создайте класс животного Animal.
2. Добавьте классу поля:
   * энергия — Int, изменяемое, приватный setter, публичный getter;
   * вес — Int, изменяемое, приватный сеттер, публичный getter;
   * возраст — Int, изменяемое, приватный сеттер, публичный getter, по умолчанию равен 0;
   * максимальный возраст — Int, неизменяемое, приватное;
   * имя — string, неизменяемое, публичное.

С помощью конструктора инициализируйте поля энергии, веса, максимального возраста, имени.

1. Добавьте вычисляемое свойство isTooOld — Boolean, неизменяемое, публичное. Оно должно сравнивать текущий возраст с максимальным и возвращать true, если текущий возраст равен или больше максимального.
2. Добавьте публичный метод sleep, который имитирует сон животного. Метод должен инкрементировать энергию на 5 единиц и возраст на 1 единицу, а также печатать «${имя животного} спит» в консоль.
3. Добавьте публичный метод eat, который имитирует потребление пищи животным. Метод должен увеличивать энергию на 3 единицы, вес на 1 единицу. Также во время еды может случайно увеличиться возраст животного на 1 единицу.

Метод должен печатать «${имя животного} ест» в консоль.

Случайное увеличение возраста реализуйте с помощью класса Random из пакета kotlin.random и метода.nextBoolean().

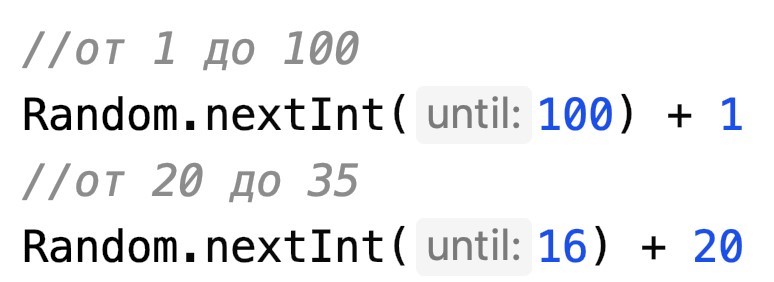
Пример использования:   


Вынесите случайное увеличение возраста в приватный метод — incrementAgeSometimes

1. Добавьте метод move, который имитирует движение животного. Энергия животного уменьшается на 5 единиц, вес на 1 единицу, случайно увеличивается возраст. Метод должен печатать «${имя животного} двигается» в консоль.
2. В методах sleep, eat, move добавьте предварительные проверки. Если животное слишком старое, если не хватает веса, энергии для совершения действия, то ничего не происходит — и метод завершается.
3. Добавьте метод makeChild, который имитирует рождение нового животного. Метод должен создавать новое животное, которое имеет имя и максимальный возраст, как у родителя. Энергия для рожденного животного устанавливается случайно в диапазоне от 1 до 10.

Вес для рожденного животного устанавливается случайно в диапазоне от 1 до 5.

Метод makeChild также печатает в консоль, какое животное было рождено, его параметры.

Пример получения случайного числа:   


1. Напишите программу, которая создаёт животное. Животное должно есть, двигаться и спать до того момента, пока не станет старым. После этого рождается новое животное, и оно продолжает этот цикл. Убедитесь по логам, что цикл продолжается бесконечно и постоянно происходит рождение новых животных.
2. Сделайте класс Animal открытым для наследования.
3. Создайте наследников класса: Bird, Fish, Dog. переопределите метод move. Для каждого наследника в методе move необходимо вызвать родительскую реализацию и дополнительно написать в консоль: для Bird — «летит», для Fish — «плывет», для Dog — «бежит».
4. Переопределите метод makeChild. Класс Fish в методе makeChild должен возвращать объект класса Fish. Аналогично с Bird и Dog.
5. Создайте абстрактный класс AgedAnimal. У этого класса должно быть только одно свойство — maxAge. Удалите у класса Animal свойство maxAge. Animal наследуется от AgedAnimal. Сделайте класс Animal абстрактным. Наследников Animal — Fish, Bird, Dog — сделайте конкретными и определите в каждом из этих классов максимальный возраст.
6. Перепишите функцию makeChild в классе Animal. Необходимо создать новое животное с помощью анонимного класса. В качестве maxAge необходимо использовать maxAge родителя. Для ссылки на maxAge родителя используйте конструкцию *this@Animal.maxAge*
7. Создайте интерфейс Soundable, у которого есть метод makeSound().
8. Классы Bird и Dog должны реализовывать этот интерфейс. Внутри метода makeSound должно писаться сообщение в консоль. Например, для собаки: «Гав-гав».
9. Создайте класс Zoo, который представляет из себя зоопарк со списком животных.
10. Инициализируйте зоопарк. Изначально в зоопарке должно быть пять птиц, три рыбы, две собаки и несколько обычных животных.
11. В цикле в течение ограниченного числа итераций N каждое животное делает одно случайное действие, которое позволяют сделать его свойства: ест, спит, двигается, рожает. Также, если животное реализует интерфейс Soundable, оно может издать звук. Когда у животного рождается детеныш, он добавляется в общий список животных зоопарка. В конце итерации цикла все животные, у которых возраст превысил максимальный, удаляются из зоопарка. По окончании цикла должно выводиться число животных в зоопарке, которые остались живы. Если все животные исчезли — цикл должен прерываться раньше, с выводом соответствующего сообщения.

**Критерии оценки**

* Названия переменных должны отражать суть данных, на которые они ссылаются.
* Переменные, которые не изменяются в программе, должны быть объявлены неизменяемыми.
* Различные выводы в консоль должны начинаться с новой строки.
* Методы должны иметь корректную область видимости. Например, если метод или свойство не должно быть доступно извне, то у них должен быть модификатор private.
* Названия классов и интерфейсов должны отражать их назначение.
* Классы должны называться в соответствии с соглашениями <https://kotlinlang.org/docs/reference/coding-conventions.html#naming-rules>.
* Классы должны быть открытыми для наследования (open) только в случае необходимости.
* Методы должны быть открытыми для переопределения (open) только в случае необходимости.
* Выполнены все пункты задания.

**Как отправить задание на проверку**

* Используйте репозиторий **[learning\_materials / android\_basic](https://gitlab.skillbox.ru/learning_materials/android_basic" \t "_blank).**
* Скачайте изменения в репозитории на локальную машину.
* Выполните домашнее задание в папке **ClassesAndInheritance**.
* Отправьте коммиты в удалённый репозиторий.

Если написанная вами программа по какой-то причине не работает и вы не можете разобраться, в чём дело, — воспользуйтесь отладчиком.

Основную информацию о нём можно найти тут:

<https://medium.com/@artem_shevchenko/android-studio-debugging-%D0%B1%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B5-%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D1%8F-%D0%B8-%D0%B2%D0%BE%D0%B7%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8-658ee6dcc641>

С помощью точек останова и просмотра информации о состоянии объектов вы сможете выявить ошибку в коде.

Мы будем рассматривать работу с отладчиком в одном из следующих модулей.

### Проверка домашнего задания

Ваша работа принята. Общение с преподавателем продолжится в следующем уроке.5.Основы Kotlin: ООП, классы и объекты. Часть 116сообщений



Виктор Кузьмин

04 ноября 2020, 15:11

Здравствуйте,

Все отлично работает!

Задание принято.

С уважением, Виктор.



Алексей Марков

04 ноября 2020, 12:24

Добрый день, Виктор!

Коллекцию поправил. Прошу проверить домашнюю работу <https://gitlab.skillbox.ru/markov_aleksei_2/android_basic/-/tree/master/KotlinOop/app/src/main/java/com/markov/classesandinheritance>

С уважением, Марков Алексей.



Виктор Кузьмин

04 ноября 2020, 11:42

Здравствуйте,

Правильно все у вас работает!

Маленькую правку можно внести!

*//С этими переменными вы потом по коду вообще не будете работать.*var dog: Animal = Dog(100, 5)  
var fish = Fish(100, 1)  
var bird = Bird(100, 3)  
var bird2 = Bird(130, 6)  
var bird3 = Bird(120, 7)  
var bird4 = Bird(135, 6)  
var bird5 = Bird(125, 5)  
var fish2 = Fish(120, 5)  
var fish3 = Fish(130, 8)  
var dog2 = Dog(135, 7)  
var panda = object : Animal(130, 100, "Miha") {}  
var tiger = object : Animal(135, 80, "Kisa") {}  
  
*//****Поэтому их анонимно можно в коллекцию положить, просто вызывая конструкторы! А от этих можно избавиться...***

var zoo = Zoo()  
*// zoo.animalList =  
// mutableListOf(dog, fish, bird, bird2, bird3, bird4, bird5, fish2, fish3, dog2, panda, tiger)  
  
 //****так:***zoo.animalList =  
 *mutableListOf*(  
 Dog(100, 5),  
 Bird(100, 3),  
 Bird(130, 6),  
 object : Animal(135, 80, "Kisa"){} *//и так далее* )  
 *//потом уже идет работа с этой коллекцией... там правильно у вас!*

С уважением, Виктор.



Алексей Марков

03 ноября 2020, 21:45

Добрый день, Виктор!

Прошу проверить домашнюю работу <https://gitlab.skillbox.ru/markov_aleksei_2/android_basic/-/tree/master/KotlinOop/app/src/main/java/com/markov/classesandinheritance>

С уважением, Марков Алексей.



Виктор Кузьмин

03 ноября 2020, 12:08

Здравствуйте,

В цикле main с собакой работает, но почему то не работало с птицей - все время циклилось на "летит".

Потому что у собаки вы всегда пробрасываете в Animal

class Dog constructor(  
 e: Int,  
 w: Int  
) : Animal(100, 5, "Muhtar"), Soundable { *//параметры 100 и 5*

Даже когда делаете ребенка:

val child = Dog(childEnergy, childWeight)

Независимо от того, что вы пробрасываете, у вас в Animal уходит 100 и 5.

а Bird вы бросаете уже параметры из конструктора Bird:

class Bird constructor(  
 e: Int,  
 w: Int  
): Animal(e, w, "Ptaha"), Soundable {

Поэтому первый раз ваша птаха всем занимается, когда ей достаточно большие параметры кинуть.

А после рождения - только летит. Смотрите параметры, при которых ребенок будет все выполнять, что требуется!

val childEnergy = Random.nextInt(10) +1  
val childWeight = Random.nextInt(5) +1  
val child = Bird(childEnergy, childWeight)

Но летит - это не потому что она летит, а потому что

*println*("$name летит")

всегда вызывается в конце метода. А super методы, как и положено у вас не проходят! Ищите баланс!

И всех остальных животных нужно сделать как Bird, а не как у вас сейчас Dog, куда 100 и 5 только пробрасывается!

Что касаемо самого цикла...

*//цикл рабочий*while (true){  
 if(dog.isTooOld){  
 dog = dog.makeChild()  
 } else {  
 dog.eat()  
 dog.move()  
 dog.sleep()  
 }  
}

С уважением, Виктор.



Алексей Марков

02 ноября 2020, 20:47

Добрый день, Виктор!

Прошу проверить домашнюю работу <https://gitlab.skillbox.ru/markov_aleksei_2/android_basic/-/tree/master/KotlinOop/app/src/main/java/com/markov/classesandinheritance>.

Выполнил по 16 пункт включительно.

В цикле main с собакой работает, но почему то не работало с птицей - все время циклилось на "летит".

С уважением, Марков Алексей.



Виктор Кузьмин

01 ноября 2020, 15:51

Здравствуйте,

Вижу, что до 14 задания дошли...

В целом идете по заданию, но  у вас программа не запускается. Причина:

fun main() {  
  
 *//Вы не можете создать инстанс абстрактного класса. Чтобы создать конкретное животное,  
 //нужно вызвать конструктор этого конкретного животного.  
//* ***val animal: Animal = Animal(100, 5, 15, "Keks")*** *//... допустим, птицы:* val animal: Animal = Bird(100, 5)

Хорошо, допустим вы вызвали конструктор птицы и теперь программа ваша запустится, но...

while (!animal.isTooOld){  
 animal.eat()  
 animal.move()  
 animal.sleep()  
}  
*//У вас программа на этом заканчивается. А нужно по заданию 9:* ***Животное должно есть,  
// двигаться и спать до того момента, пока не станет старым.  
// После этого рождается новое животное, и оно продолжает этот цикл.  
// Убедитесь по логам, что цикл продолжается бесконечно и постоянно происходит рождение  
// новых животных.***animal.makeChild()

Сами классы правильно по заданию реализованы, нужно доработать Main!

С уважением, Виктор.



Алексей Марков

30 октября 2020, 13:51

Добрый день, Виктор!

Прошу проверить домашнюю работу <https://gitlab.skillbox.ru/markov_aleksei_2/android_basic/-/tree/master/KotlinOop/app/src/main/java/com/markov/classesandinheritance>

С уважением, Марков Алексей.



Виктор Кузьмин

18 октября 2020, 14:56

Здравствуйте,

Все по заданию делаете правильно.

Единственное, что для проверки я бы energy проверял не равно нулю, а Меньше или равно нулю:

Не

if (isTooOld || energy == 0 || weight <= 1) {

А так

if (isTooOld || energy <= 0 || weight <= 1) {

Иначе, если у вас энергия будет в минусе (-5, допустим), и условие, что энергии не хватает не пройдет, потому что ваше условие рассчитано исключительно под 0.

С уважением, Виктор.



Алексей Марков

17 октября 2020, 22:45

Добрый вечер, Виктор!

Прошу проверить часть домашней работы

<https://gitlab.skillbox.ru/markov_aleksei_2/android_basic/-/blob/master/KotlinOop/app/src/main/java/com/markov/classesandinheritance/Animal.kt>

С уважением, Марков Алексей.



Виктор Кузьмин

17 октября 2020, 13:53

Здравствуйте,

У меня такой папки нет. В какой папке мне выполнять эту домашнюю работу?

Создайте свою папку, только, пожалуйста, ссылку на эту папку дайте!

С уважением, Виктор.



Алексей Марков

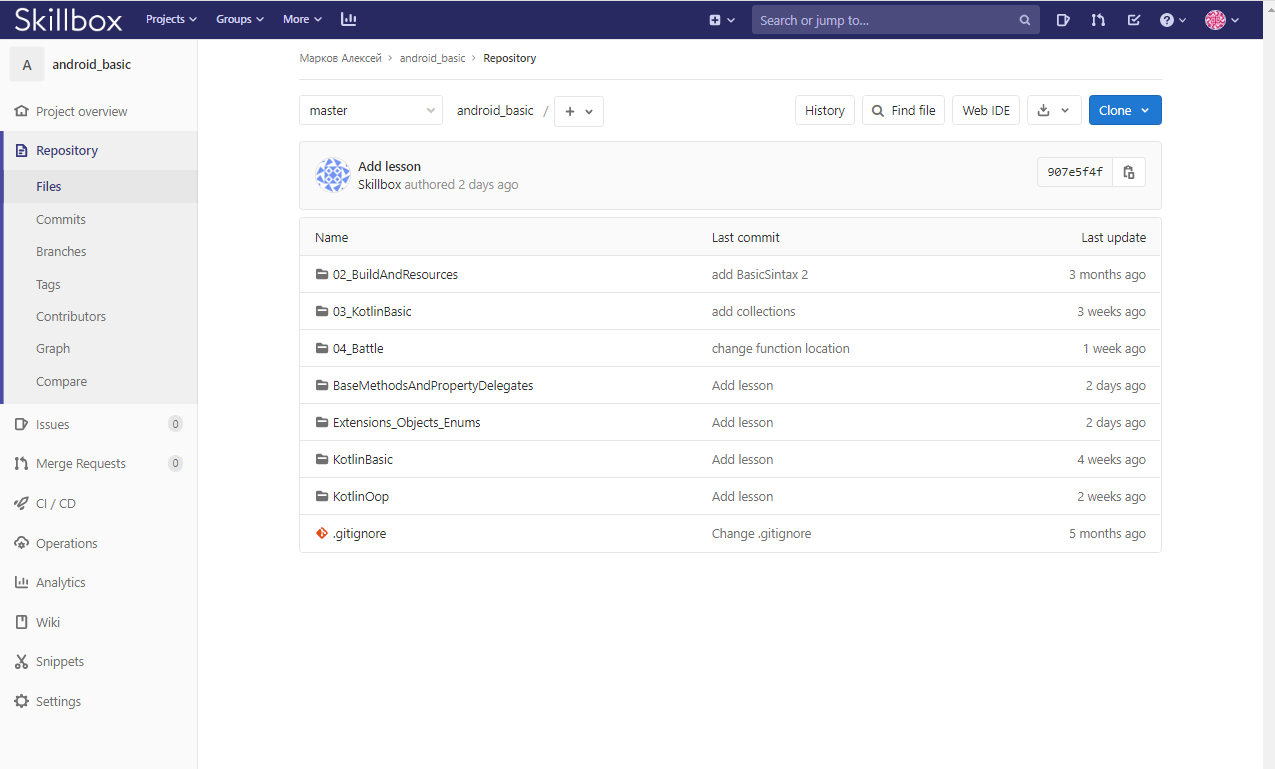
17 октября 2020, 13:22

Добрый день, Виктор!

Снова произошли изменения в программе...

Вопрос по заданию:

* Выполните домашнее задание в папке **ClassesAndInheritance**.  
  У меня такой папки нет. В какой папке мне выполнять эту домашнюю работу?



С уважением, Марков Алексей.