Лекция #9.4 Абстрактные классы и методы

Абстрактные классы — это классы, определенные с модификатором **abstract**. Отличительной особенностью абстрактных классов является то, что мы не можем создать объект подобного класса. Например, определим абстрактный класс Human:

```
abstract class Human(val name: String)
```

Абстрактный класс, как и обычный, может иметь свойства, функции, конструкторы, но создать его объект напрямую вызвав его конструктор мы не можем:

```
val kate: Human     // норм, просто определение переменной
val alice: Human = Human("Alice")   // ! ошибка, создать объект нельзя
```

Такой класс мы можем только унаследовать:

```
abstract class Human(val name: String){

    fun hello(){
        println("My name is $name")
    }
}
class Person(name: String): Human(name)
```

Стоит отметить, что в данном случае перед абстрактным классом не надо указывать аннотацию open, как при наследовании неабстрактных классов.

```
fun main(args: Array<String>) {

    val kate: Person = Person("Kate")
    val slim: Human = Person("Slim Shady")
    kate.hello()    // My name is Kate
    slim.hello()    // My name is Slim Shady
}
```

Абстрактные классы могут иметь абстрактные методы и свойства. Это такие функции и свойства, которые определяются с ключевым словом **abstract**. Абстрактные методы не содержат реализацию, то есть у них нет тела. А для абстрактных свойств не указывается значение. При этом абстрактные методы и свойства можно определить только в абстрактных классах:

```
abstract class Human(val name: String){

    abstract var age: Int
    abstract fun hello()
}
class Person(name: String): Human(name){

    override var age : Int = 1
    override fun hello(){
        println("My name is $name")
    }
}
```

Если класс наследуется от абстрактного класса, то он должен либо реализовать все его абстрактные методы и свойства, либо также быть абстрактным.

Так, в данном случае класс Person должен обязательно определить реализацию для функции `hello()` и свойства age. При этом, как и при переопределении обычных методов и свойств, применяется аннотация **override**.

Абстрактные свойства также можно реализовать в первичном конструкторе:

```
abstract class Human(val name: String){

    abstract var age: Int
    abstract fun hello()
}
class Person(name: String, override var age : Int): Human(name){
    override fun hello(){
        println("My name is $name")
    }
}
```

Зачем нужны абстрактные классы? Классы обычно отражают какие-то сущности реального мира. Но некоторые из этих сущностей представляют абстракцию, которая непосредственного воплощения не имеет. Например, возьмем систему геометрических фигур. В реальности не существует геометрической фигуры как таковой. Есть круг, прямоугольник, квадрат, но просто фигуры нет. Однако же и круг, и прямоугольник имеют что-то общее и являются фигурами.

В этом случае мы можем определить абстрактный класс фигуры и затем от него унаследовать все остальные классы фигур:

```
// абстрактный класс фигуры
abstract class Figure {
    // абстрактный метод для получения периметра
    abstract fun perimeter(): Float

    // абстрактный метод для получения площади
    abstract fun area(): Float
}
// производный класс прямоугольника
class Rectangle(val width: Float, val height: Float) : Figure()
{
    // переопределение получения периметра
    override fun perimeter(): Float{
        return width * 2 + height * 2;
    }
    // переопрелеление получения площади
    override fun area(): Float{
        return width * height;
    }
}
```