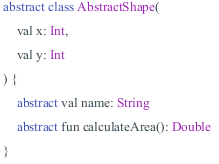
**Наследование в Kotlin:**

* Отсутствует множественное наследование
* На вершине иерархии наследования находится класс Any
* Необходимы модификаторы класса - open или abstract
* Дочерний класс наследует не-private члены родительского
* Класса

**Абстрактный класс:**

Абстрактный класс — это класс, представляющий из себя “заготовку” для целого семейства классов, который описывает для них общий шаблон поведения. абстрактный класс — это средство, для повторного использования кода. Объявляется абстрактный класс при помощи ключевого слова abstract. Наследование от такого класса осуществляется с помощью оператора: при этом абстрактному классу не нужен модификатор open, потому что он “открыт” для наследования по умолчанию.

* Предназначен для наследования
* Невозможно создать экземпляр абстрактного класса
* Может иметь абстрактные методы и свойства
* Может содержать обычные методы и свойства и хранить
* Состояние



**Интерфейс:**

Интерфейс — это совокупность методов и правил, которые определяют поведение класса или общее поведение для группы независимых друг от друга классов. Интерфейсы похожи на абстрактные классы тем, что нельзя создать их экземпляры и они могут определять абстрактные или конкретные функции и свойства. Отличие в том, что интерфейсу не важна связь “родитель-наследник”, он задаёт лишь правила поведения. Объявляется интерфейс ключевым словом interface.

interface Cultivable {

...

}

Реализация интерфейса осуществляется аналогично наследованию: после имени класса ставится оператор: и название интерфейса. В теле интерфейса можно определять абстрактные свойства и функции. Для этого не требуется использовать ключевое слово abstract Единственный способ определить свойство — это определить его в теле интерфейса, так как у интерфейса не бывает конструкторов.

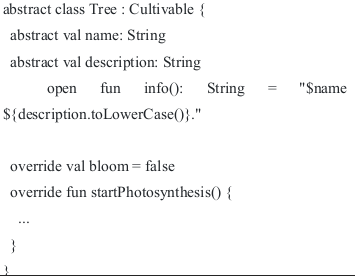
interface Cultivable {

val bloom: Boolean

fun startPhotosynthesis ()

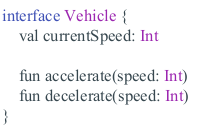
}

Класс должен реализовывать все абстрактные свойства и функции, определённые в интерфейсе.



Интерфейс

* Может иметь поля и методы
* Все поля и методы будут доступны пользователю интерфейса
* Нет внутренней реализации методов и полей
* Нельзя создать объект интерфейса
* Нет внутреннего состояния
* Реализуются классами, возможна множественная реализация



**Анонимные классы:**

* Не имеют имени
* Используются для создания объекта один раз
* С помощью них можно создать объект интерфейса,
* абстрактного класса либо наследовать существующий класс

