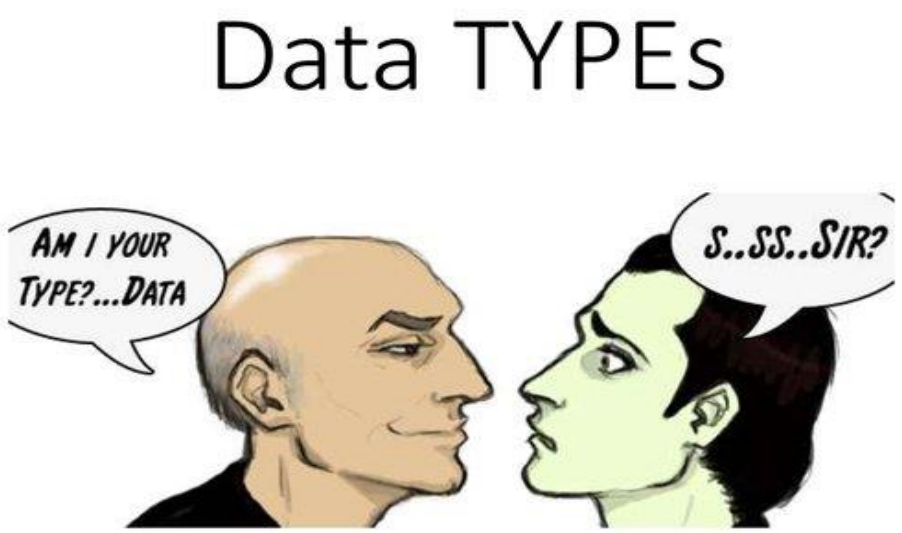
Predefined types and the cases of it using

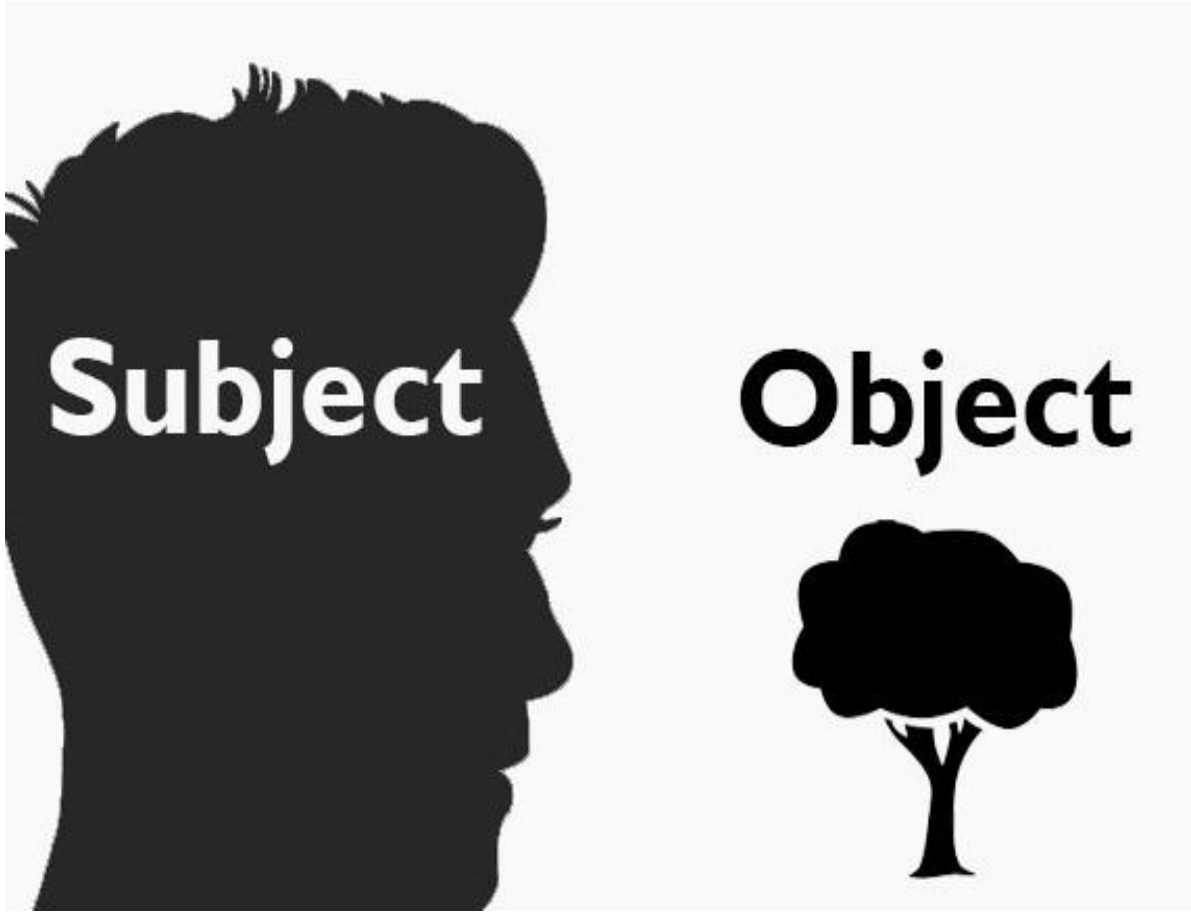
Предопределённые типы и случаи их использования

1.1 Что такое типы?



2.1 Объекты

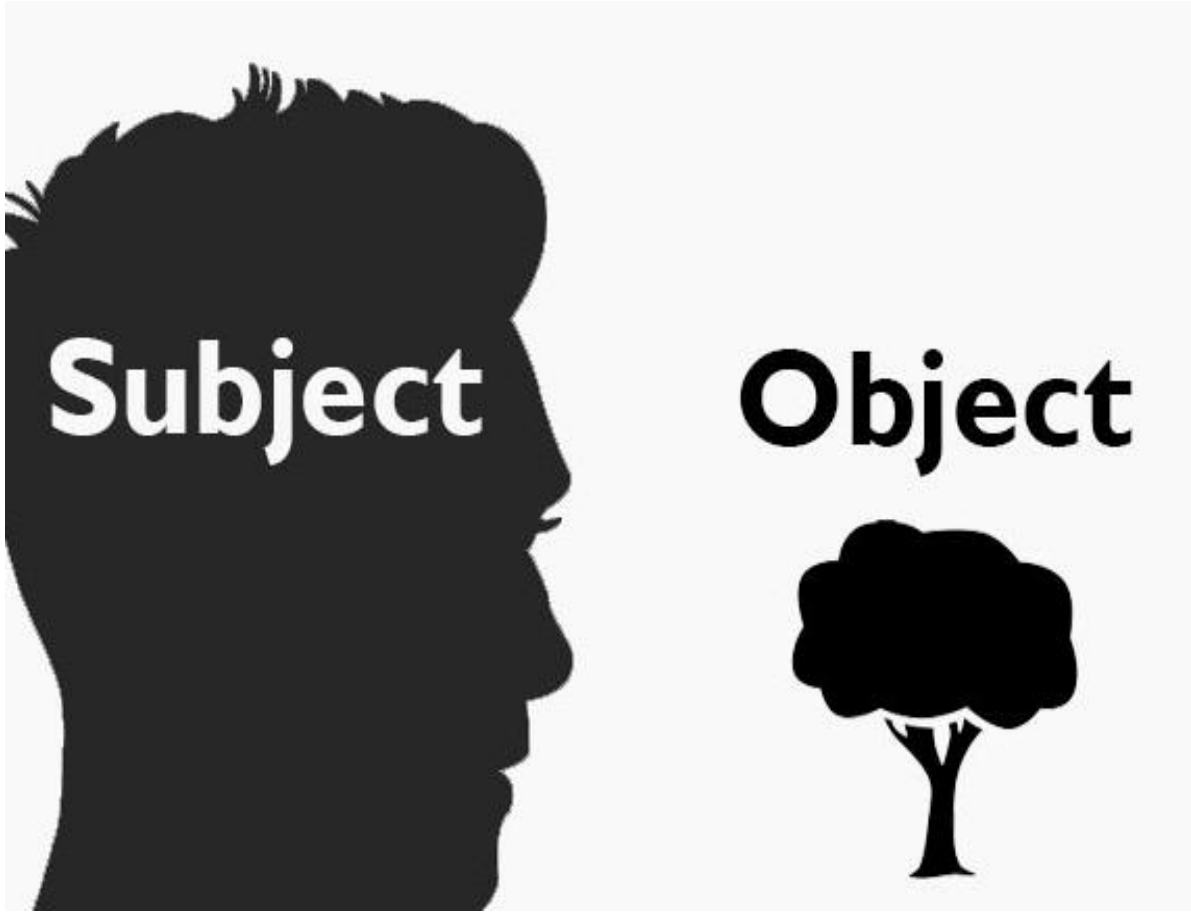
Объект – это всё что угодно, что вы (являясь в свою очередь **субъектом**) можете выдел как отдельную сущность.



2.1 Объекты

Объект – это всё что угодно, что вы (являясь в свою очередь **субъектом**) можете выдел как отдельную сущность.

Субъект – сущность которая наблюдает объекты.

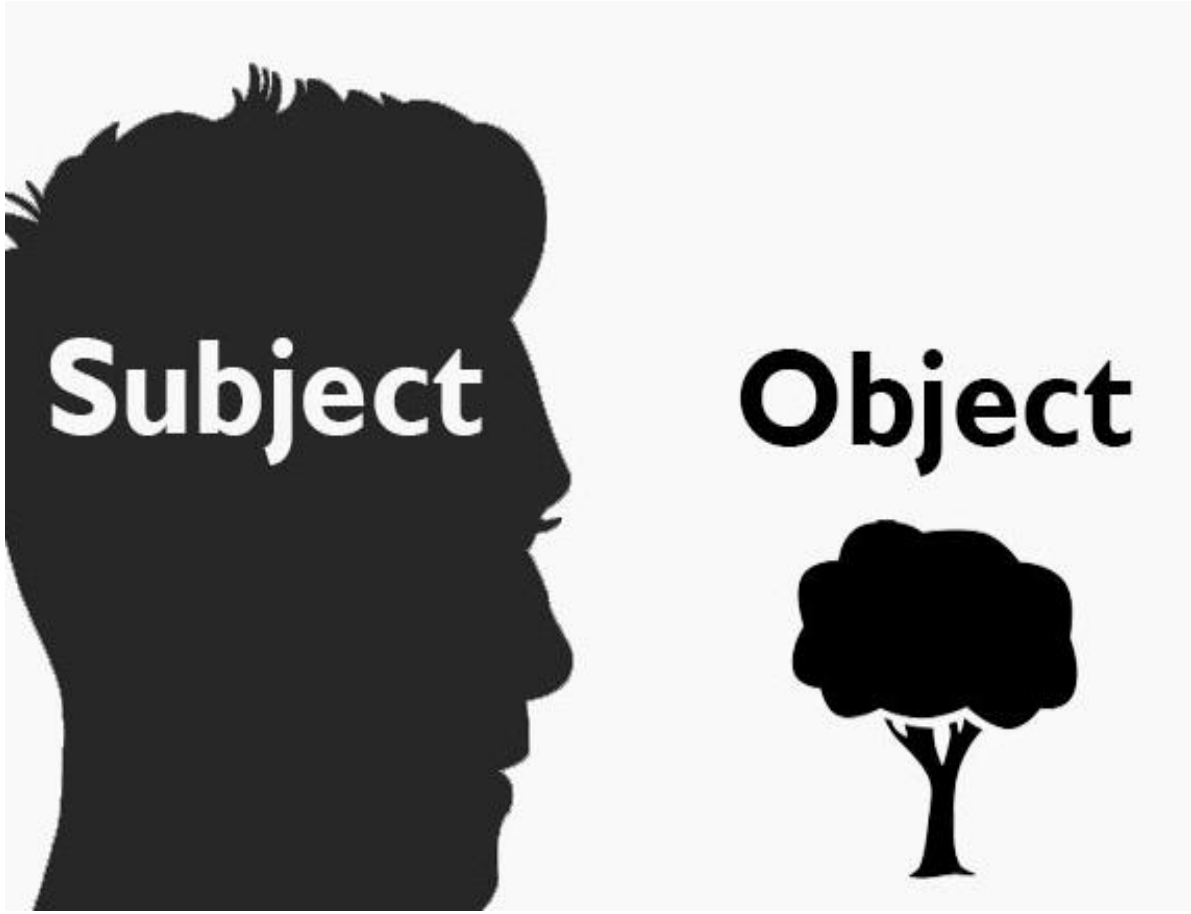


2.1 Объекты

Объект – это всё что угодно, что вы (являясь в свою очередь **субъектом**) можете выдел как отдельную сущность.

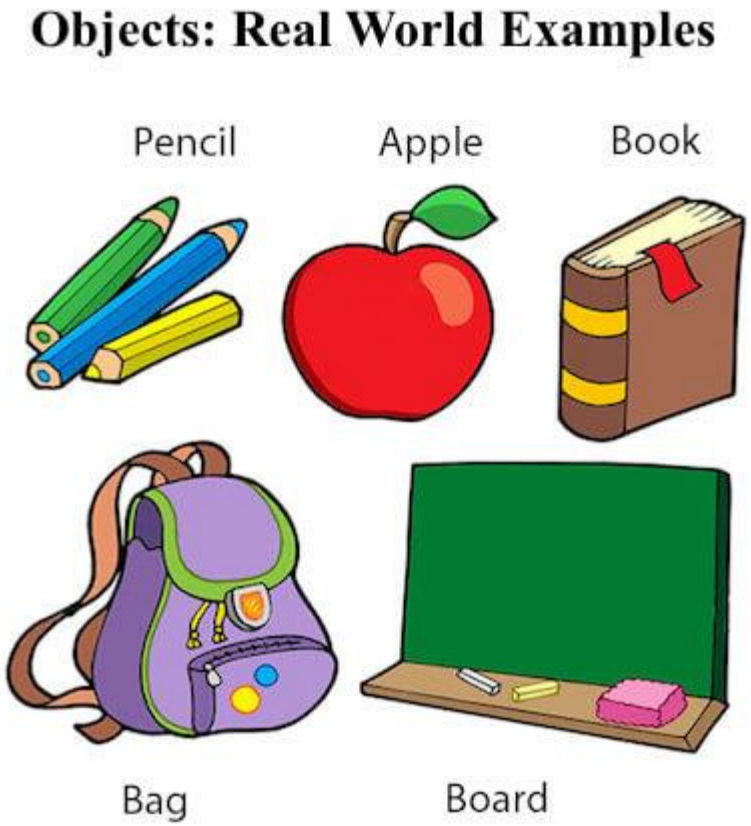
Субъект – сущность которая наблюдает объекты.

Это простая, но очень важная концепция которая лежит в основе современной математики и программирования.



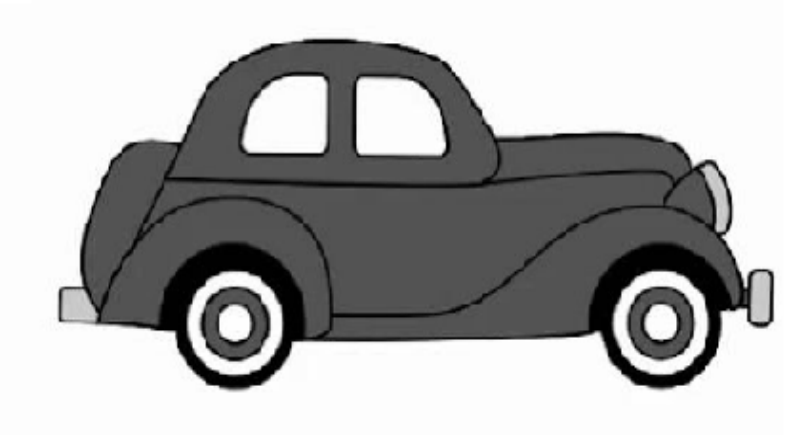
2.2 Всё есть объект

- **Карандаши, яблоко, книга, рюкзак** и **доска** – это объекты
- **Слова, буквы, знаки препинания** – это тоже объекты
- **Линии, фигуры** и **цвета** – это также объекты
- И даже сам это **рисунок** – это объект!



2.3 ООП Объекты

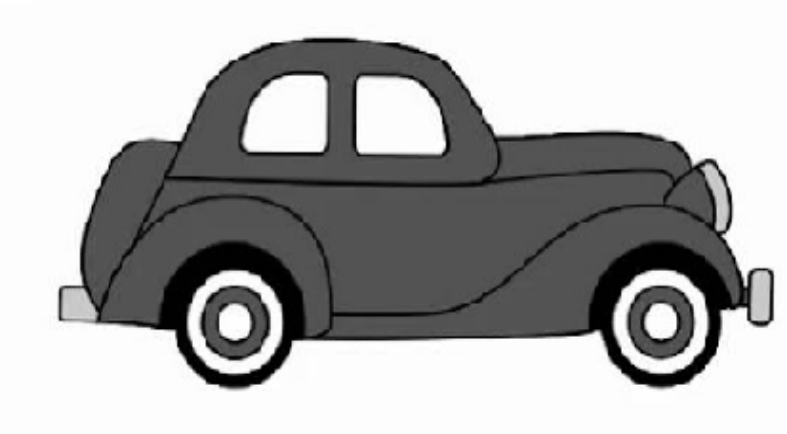
- **ООП Объект** – это численная модель некоторого другого объекта.



```scala
object Car {

  val color: String = "red"
  val size: Int = 4000
  val capacity: Int = 5
  val weight: Double = 1.2

  def start(key: String): Unit = ???
  def stop(): Unit = ???

}
```

2.3 ООП Объекты

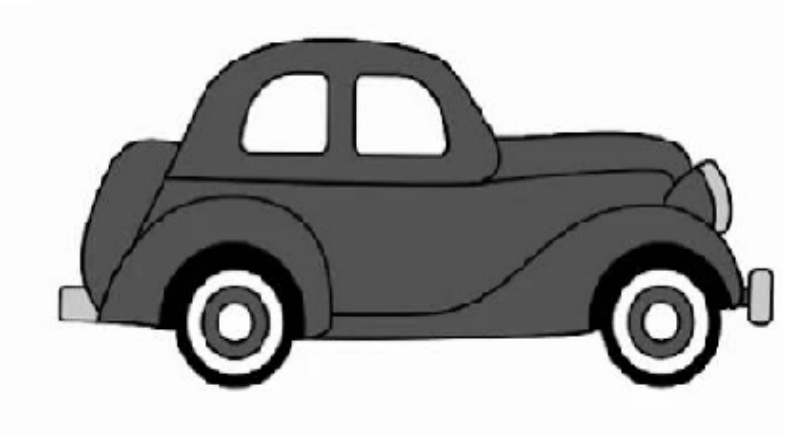
- **ООП Объект** – это численная модель некоторого другого объекта.
- **Объектно-ориентированное программирование (ООП)** – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. (Wikipedia)



```
object Car {

  val color: String = "red"
  val size: Int = 4000
  val capacity: Int = 5
  val weight: Double = 1.2

  def start(key: String): Unit = ???
  def stop(): Unit = ???

}
```

2.3 ООП Объекты

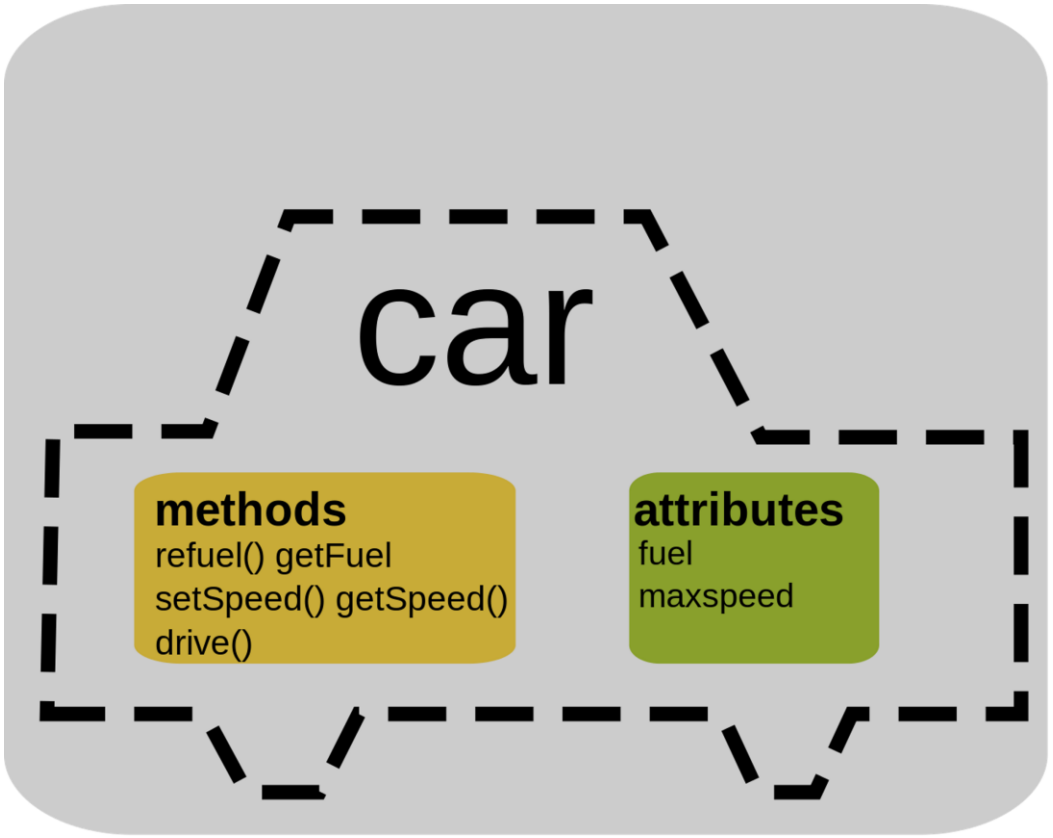
- **ООП Объект** – это численная модель некоторого другого объекта.
- **Объектно-ориентированное программирование (ООП)** – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. (Wikipedia)
- **Объектно-ориентированный дизайн (ООД)** – это процесс использования объектно-ориентированной методологии для проектирования вычислительной системы или программы. Эта методика позволяет реализовать программное решение на основе концепций объектов. (Откуда то из интернетов)



```
object Car {

  val color: String = "red"
  val size: Int = 4000
  val capacity: Int = 5
  val weight: Double = 1.2

  def start(key: String): Unit = ???
  def stop(): Unit = ???

}
```

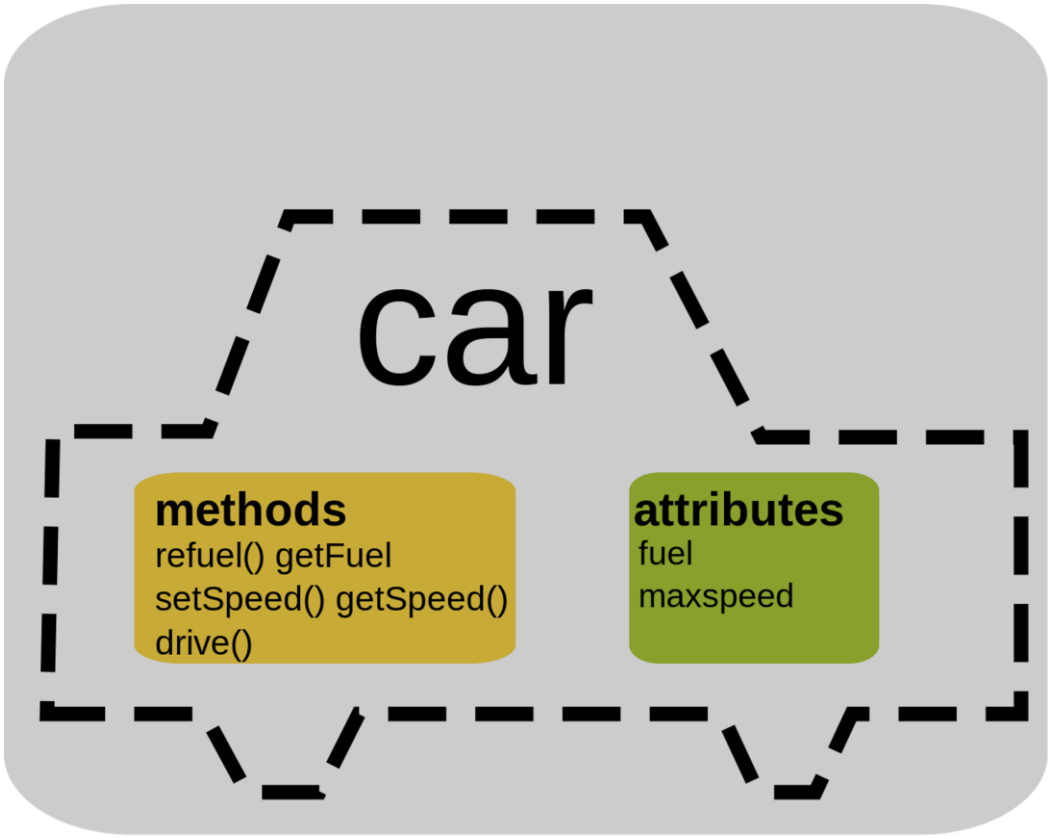
2.4 Атрибуты и методы

- ООП Объект с точки зрения компьютера всего лишь набор атрибутов и методов.



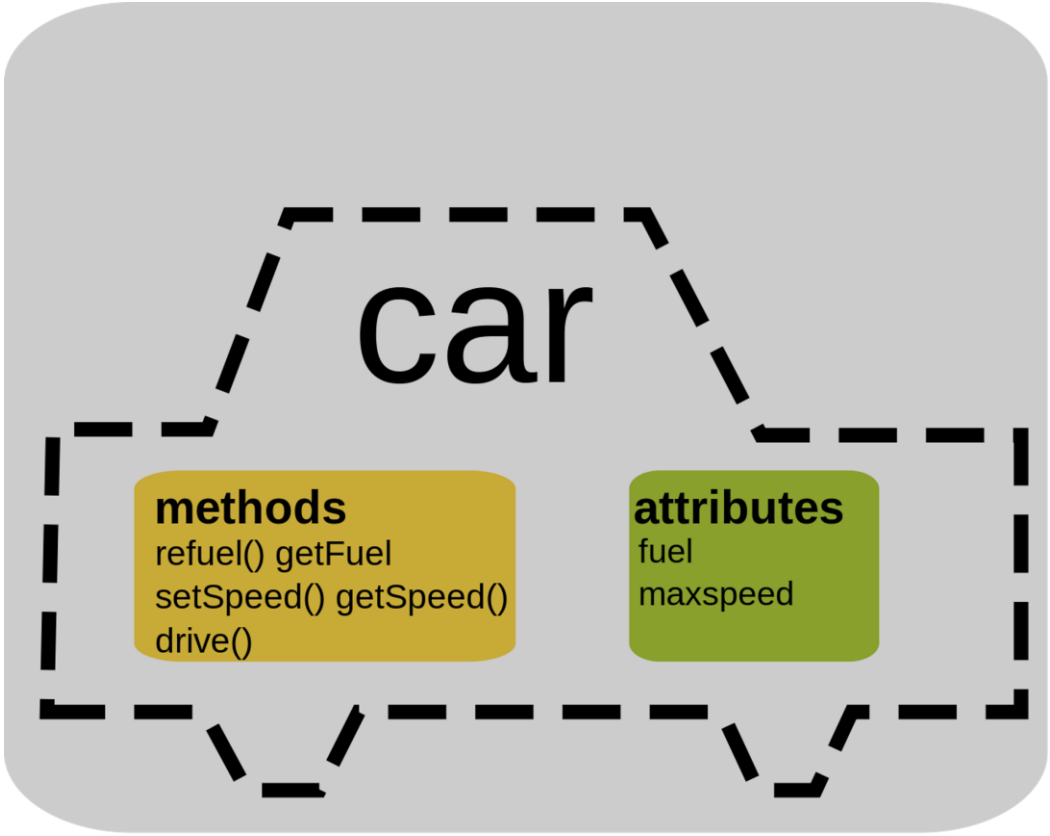
2.4 Атрибуты и методы

- ООП Объект с точки зрения компьютера всего лишь набор атрибутов и методов.
- **Атрибуты** (они же переменные, значения, константы, состояние) – это отражение свойств или признаков моделируемого объекта.



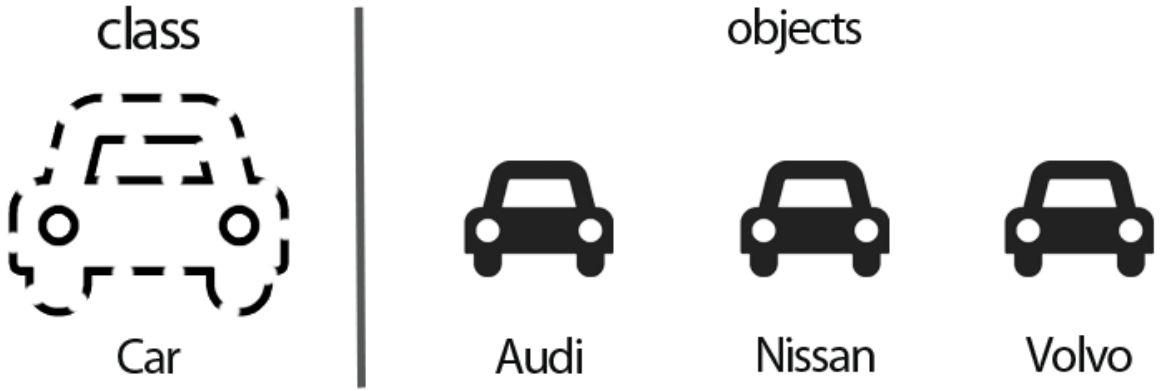
2.4 Атрибуты и методы

- ООП Объект с точки зрения компьютера всего лишь набор атрибутов и методов.
- **Атрибуты** (они же переменные, значения, константы, состояние) – это отражение свойств или признаков моделируемого объекта.
- **Методы** (они же функции, процедуры, подпрограммы, сообщения) – это отражение действий, которые можно выполнить с моделируемым объектом.



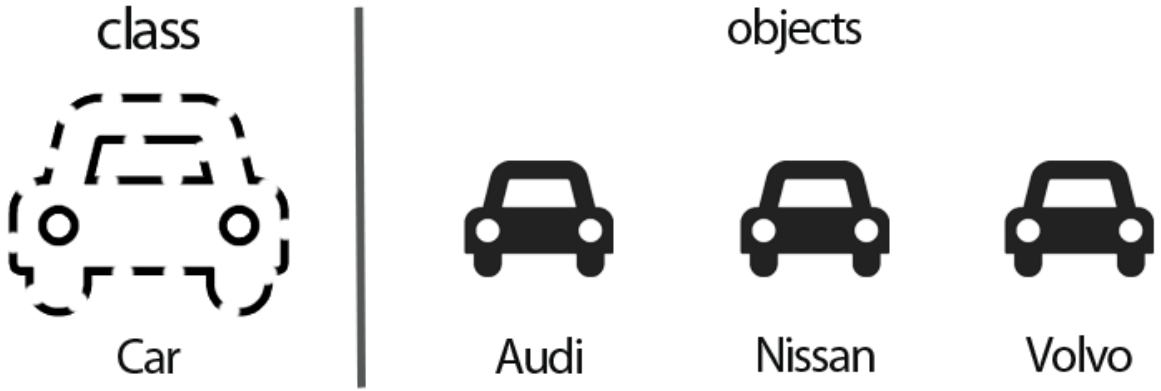
2.5 Типы объектов

- **Классификация** – система группировки объектов исследования или наблюдения в соответствии с их общими признаками. (Wikipedia)



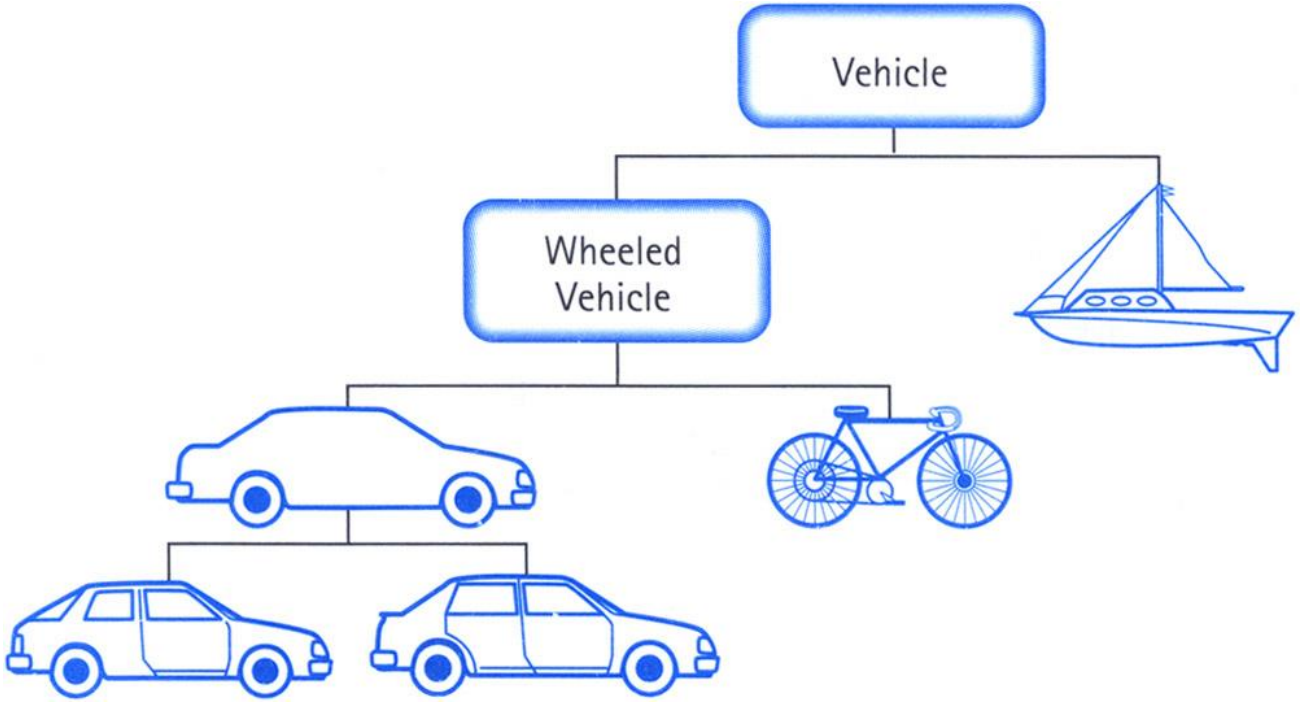
2.5 Типы объектов

- **Классификация** – система группировки объектов исследования или наблюдения в соответствии с их общими признаками. (Wikipedia)
- **Тип** (он же **класс**) – это группа объектов имеющая некоторые общие свойства или признаки.



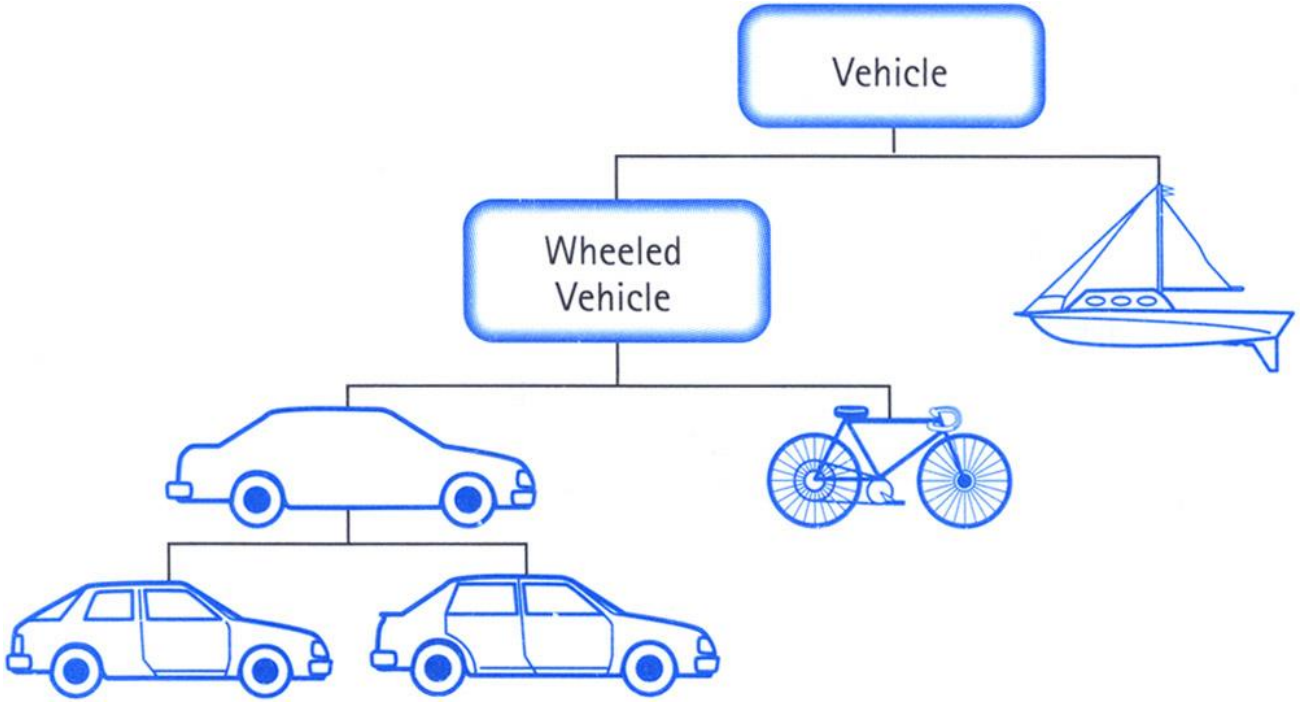
2.6 Иерархия типов

- Типы также можно группировать в другие типы, тем получая **иерархию типов**.



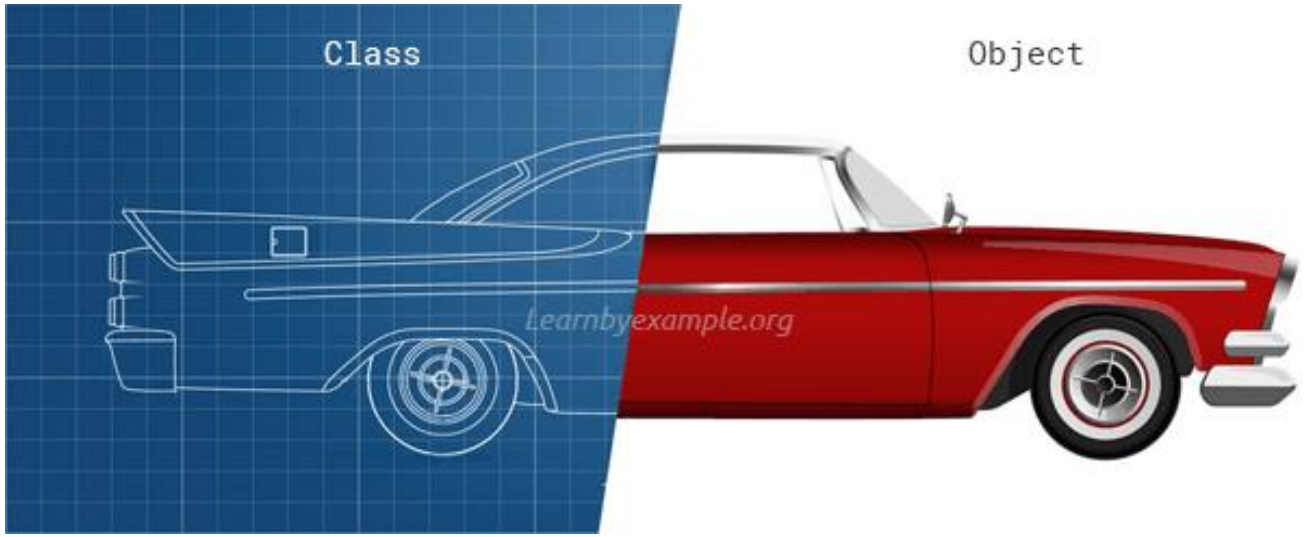
2.6 Иерархия типов

- Типы также можно группировать в другие типы, тем получая **иерархию типов**.
- В верху иерархии наиболее абстрактный тип. Внизу иерархии наиболее конкретный тип.



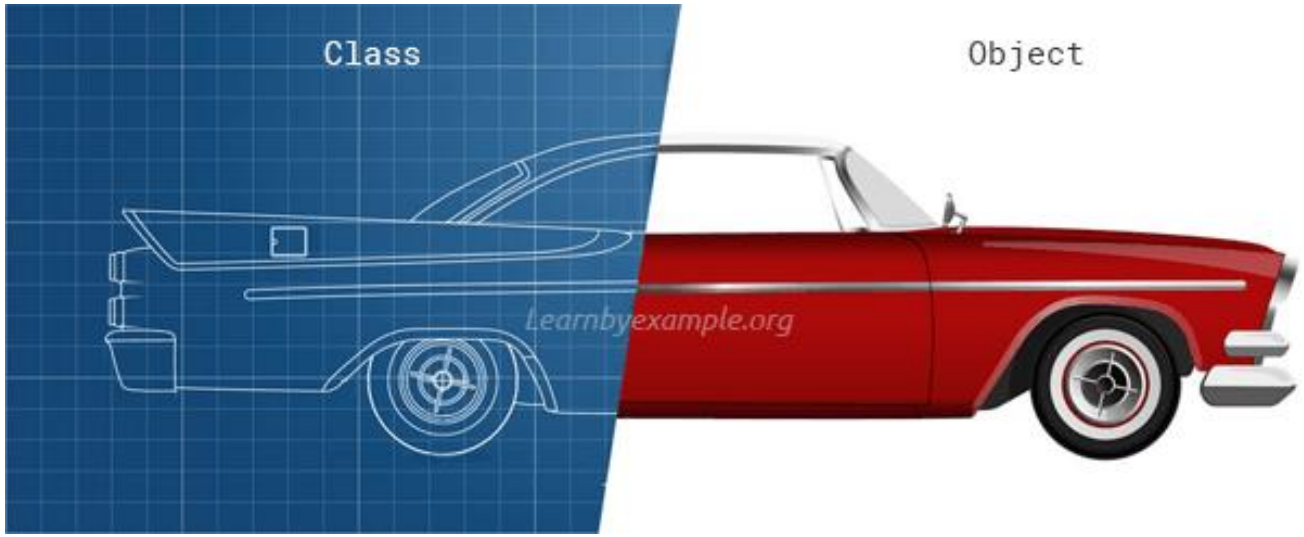
2.7 ООП Типы

- В ООП тоже есть **типы** и их **иерархия**, они так же являются классификацией объектов. Однако…



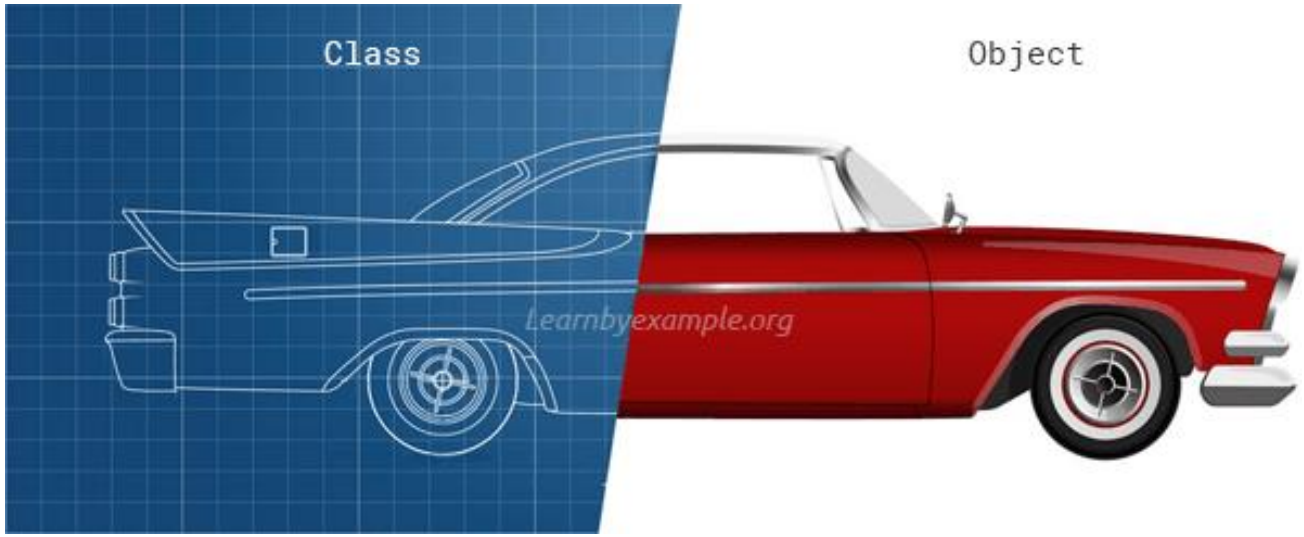
2.7 ООП Типы

- В ООП тоже есть **типы** и их **иерархия**, они так же являются классификацией объектов. Однако…
- **ООП типы** это не только и не столько классификация. Это прежде всего контейнеры для кода! Место для атрибутов и методов.



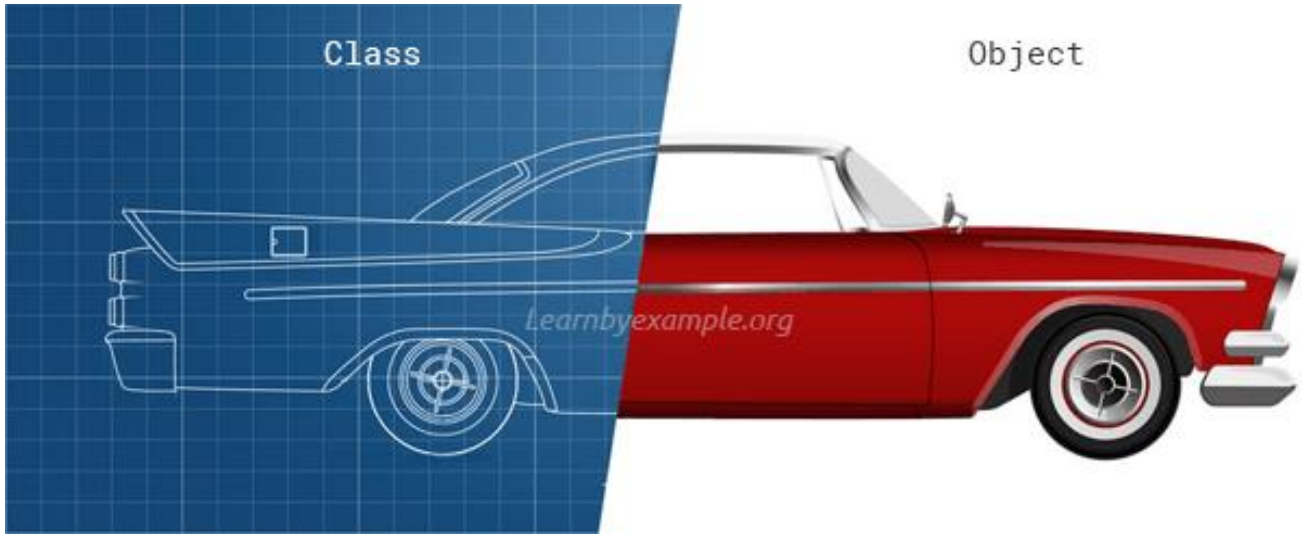
2.7 ООП Типы

- В ООП тоже есть **типы** и их **иерархия**, они так же являются классификацией объектов. Однако…
- **ООП типы** это не только и не столько классификация. Это прежде всего контейнеры для кода! Место для атрибутов и методов.
- ООП тип это как **чертёж** ООП объекта.



2.7 ООП Типы

- В ООП тоже есть **типы** и их **иерархия**, они так же являются классификацией объектов. Однако…
- **ООП типы** это не только и не столько классификация. Это прежде всего контейнеры для кода! Место для атрибутов и методов.
- ООП тип это как **чертёж** ООП объекта.
- Система типов — совокупность правил в языках программирования, назначающих свойства, именуемые типами, различным конструкциям, составляющим программу — таким как переменные, выражения, функции или модули. (Wikipedia)



3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.

3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.
- Не надо так

3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.
- Не надо так
- ООП и ФП не просто совместимы но и прекрасно дополняют друг друга.

3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.
- Не надо так
- ООП и ФП не просто совместимы но и прекрасно дополняют друг друга.
- ООП – это инструменты дизайна и структурирования кода.

3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.
- Не надо так
- ООП и ФП не просто совместимы но и прекрасно дополняют друг друга.
- ООП – это инструменты дизайна и структурирования кода.
- ФП – это инструменты для манипуляции с объектами.

3.1 ООП + ФП = Scala

- Многие противопоставляют Объектно-Ориентированное и Функциональное программирование.
- Не надо так
- ООП и ФП не просто совместимы но и прекрасно дополняют друг друга.
- ООП – это инструменты дизайна и структурирования кода.
- ФП – это инструменты для манипуляции с объектами.

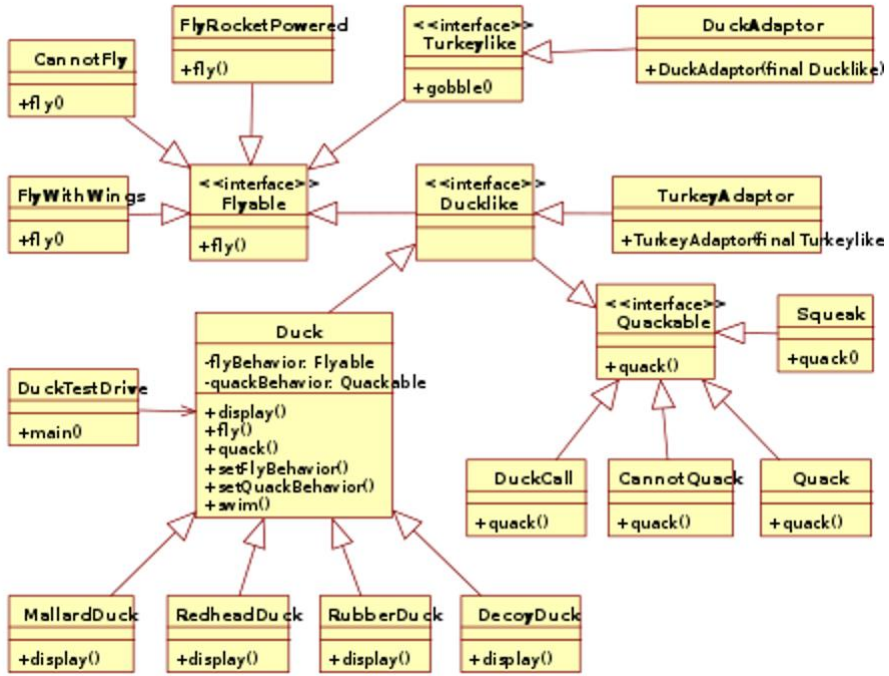
Сегодня будем больше говорить об ООП, про ФП в следующих лекциях.

3.2 Демо-кодинг

- Определение типа **Car**
- Создание объекта **new Car**
- Запуск приложения
- Аргументы конструктора

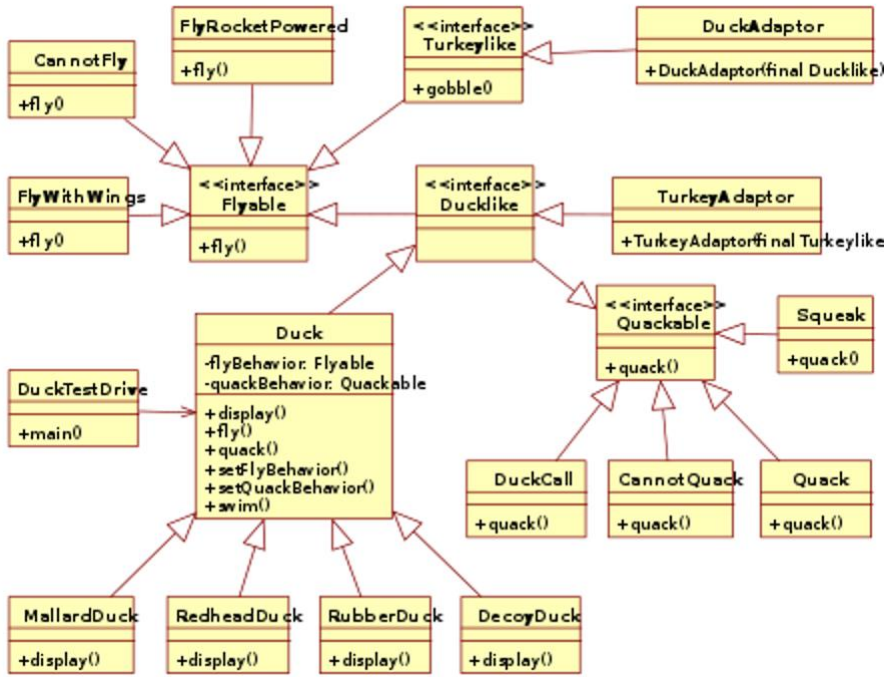
4.1 Все типы из типов

- Определяя тип Car мы собрали его из типов String, Int, Boolean.



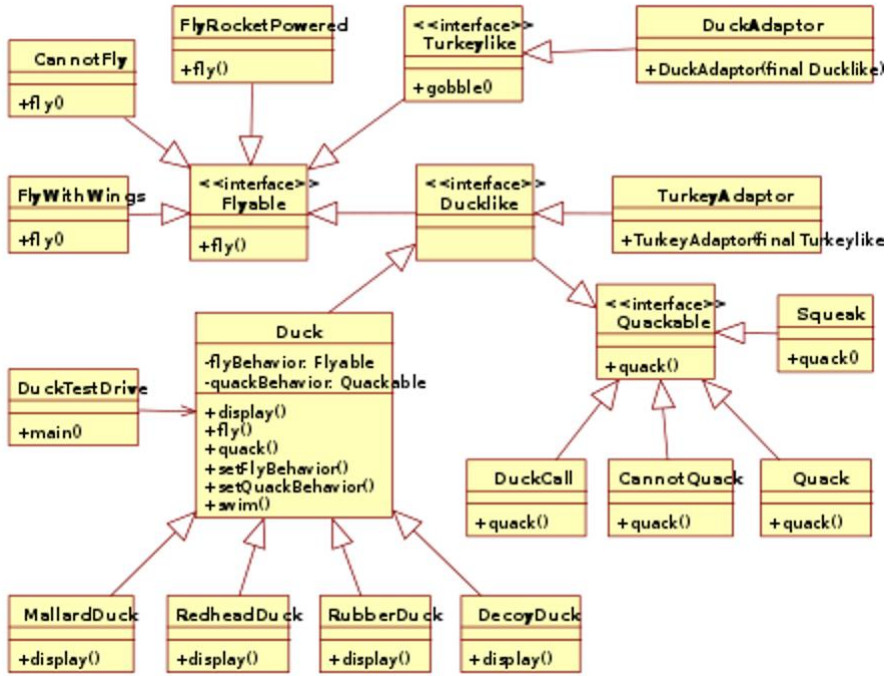
4.1 Все типы из типов

- Определяя тип Car мы собрали его из типов String, Int, Boolean.
- Сборка типов из типов это фундаментальный принцип в ООП, называемый **композиция**.



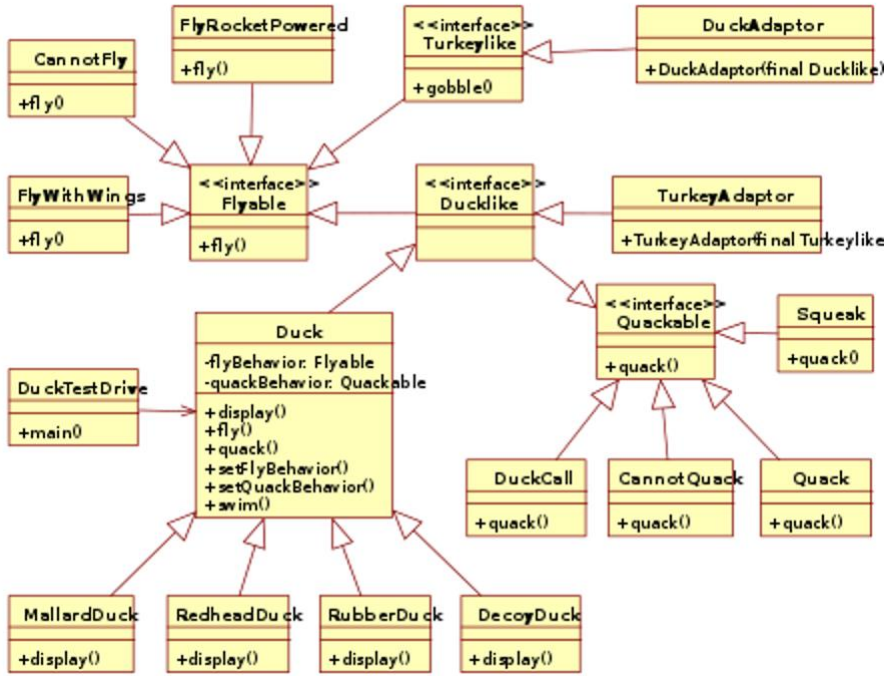
4.1 Все типы из типов

- Определяя тип Car мы собрали его из типов String, Int, Boolean.
- Сборка типов из типов это фундаментальный принцип в ООП, называемый **композиция**.
- Композиция это то чему вы будете посвящать существенную часть времени будучи профессиональным программистом.



4.1 Все типы из типов

- Определяя тип Car мы собрали его из типов String, Int, Boolean.
- Сборка типов из типов это фундаментальный принцип в ООП, называемый **композиция**.
- Композиция это то чему вы будете посвящать существенную часть времени будучи профессиональным программистом.
- Значит ли это что **все типы являются композицией из типов**!?



4.2 Предопределённые типы

- **Нет**, не все типы являются композицией из типов!

4.2 Предопределённые типы

- **Нет**, не все типы являются композицией из типов!
- Должны существовать самые первые типы, до которых не существовало типов совсем. И следовательно они не могут быть собраны из других типов.

4.2 Предопределённые типы

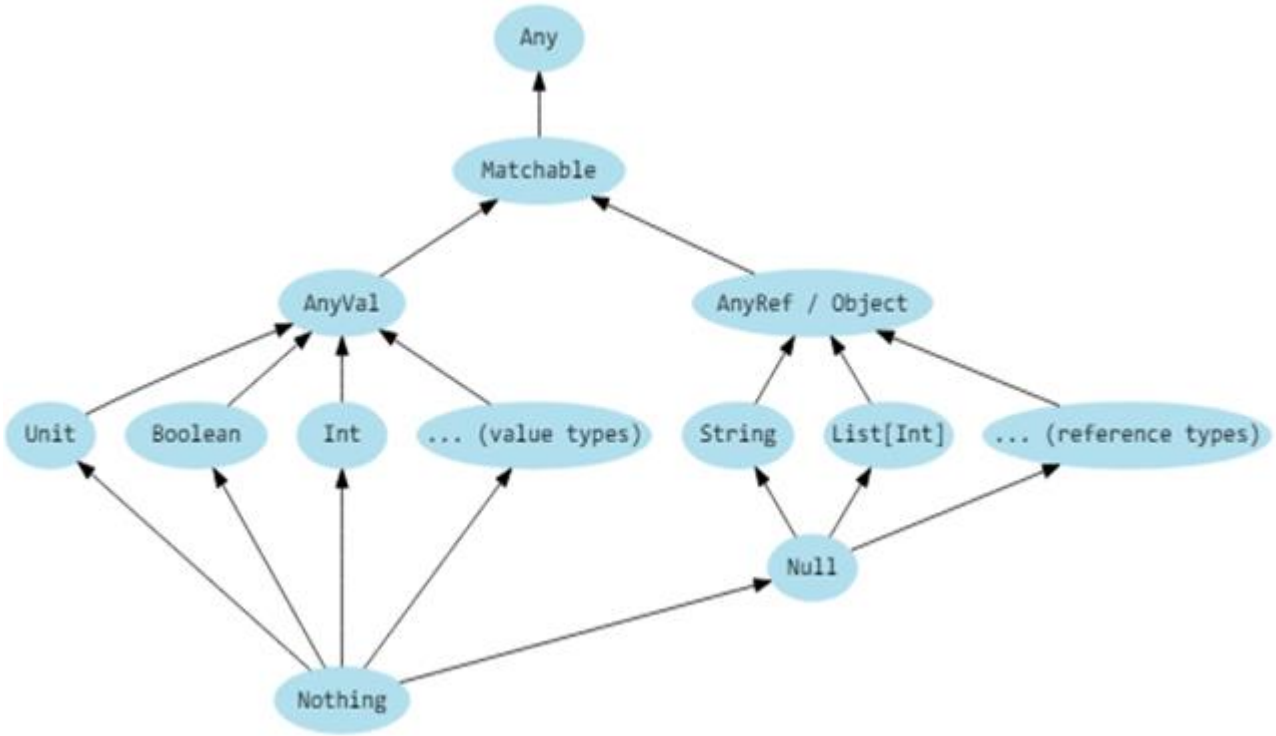
- **Нет**, не все типы являются композицией из типов!
- Должны существовать самые первые типы, до которых не существовало типов совсем. И следовательно они не могут быть собраны из других типов.
- Такие типы есть в скале и они называются **предопределённые типы**.

4.2 Предопределённые типы

- **Нет**, не все типы являются композицией из типов!
- Должны существовать самые первые типы, до которых не существовало типов совсем. И следовательно они не могут быть собраны из других типов.
- Такие типы есть в скале и они называются **предопределённые типы**.
- Можно думать о них как о самых простых и базовых кирпичиках, в основании всех остальных типов.

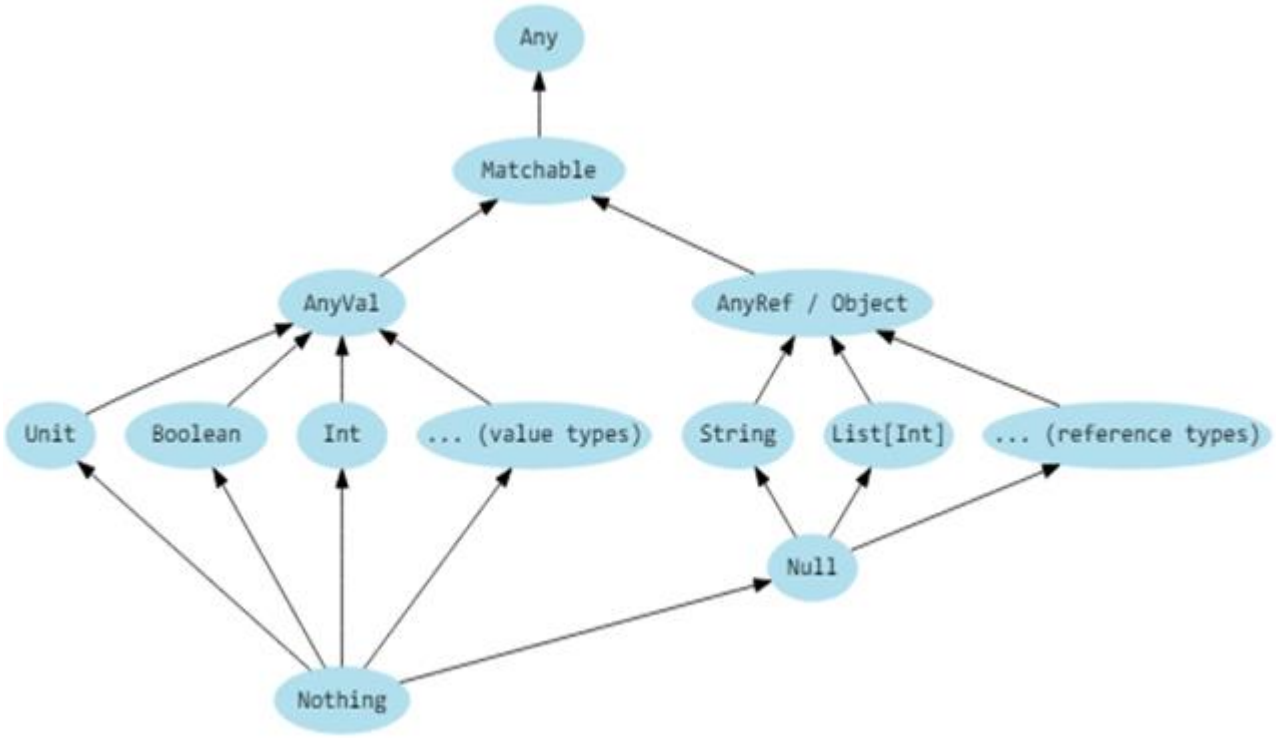
5.1 Предопределённые типы в Scala

- Предопределённых типов в Scala всего 9: **Byte**, **Double**, **Float**, **Boolean**, **Char**, **Long**, **Int**, **Short** и **Unit.**



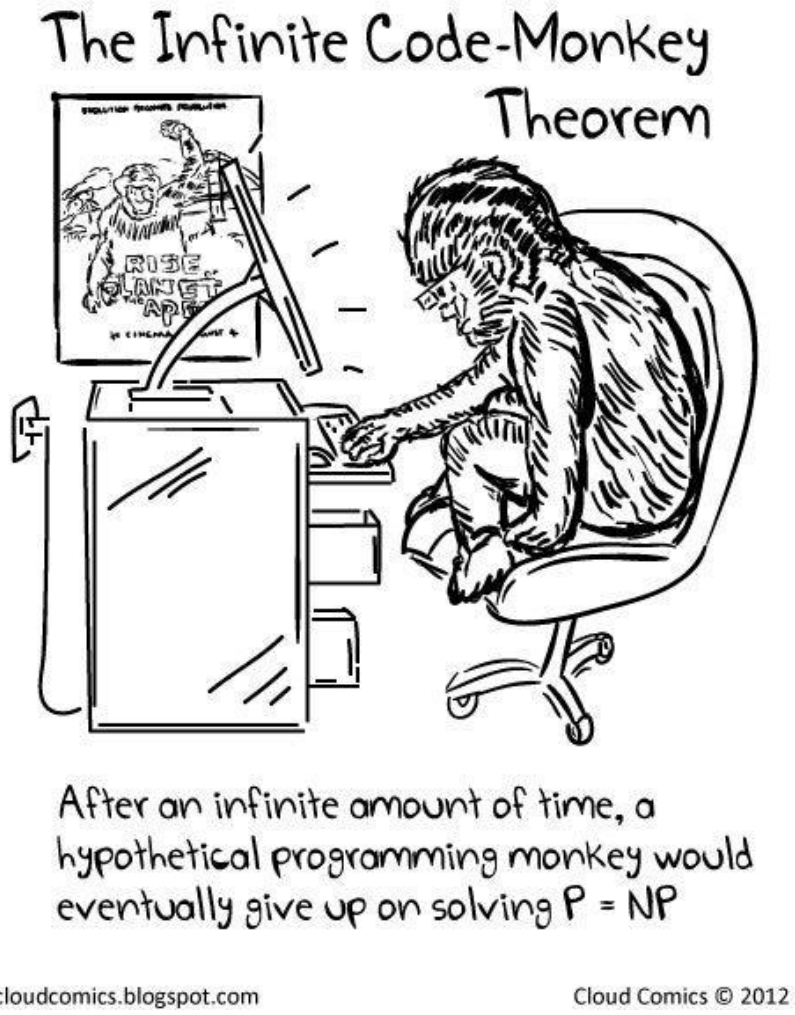
5.1 Предопределённые типы в Scala

- Предопределённых типов в Scala всего 9: **Byte**, **Double**, **Float**, **Boolean**, **Char**, **Long**, **Int**, **Short** и **Unit.**
- В иерархии типов Scala, предопределённые типы занимают свое место между **AnyVal** и **Nothing**



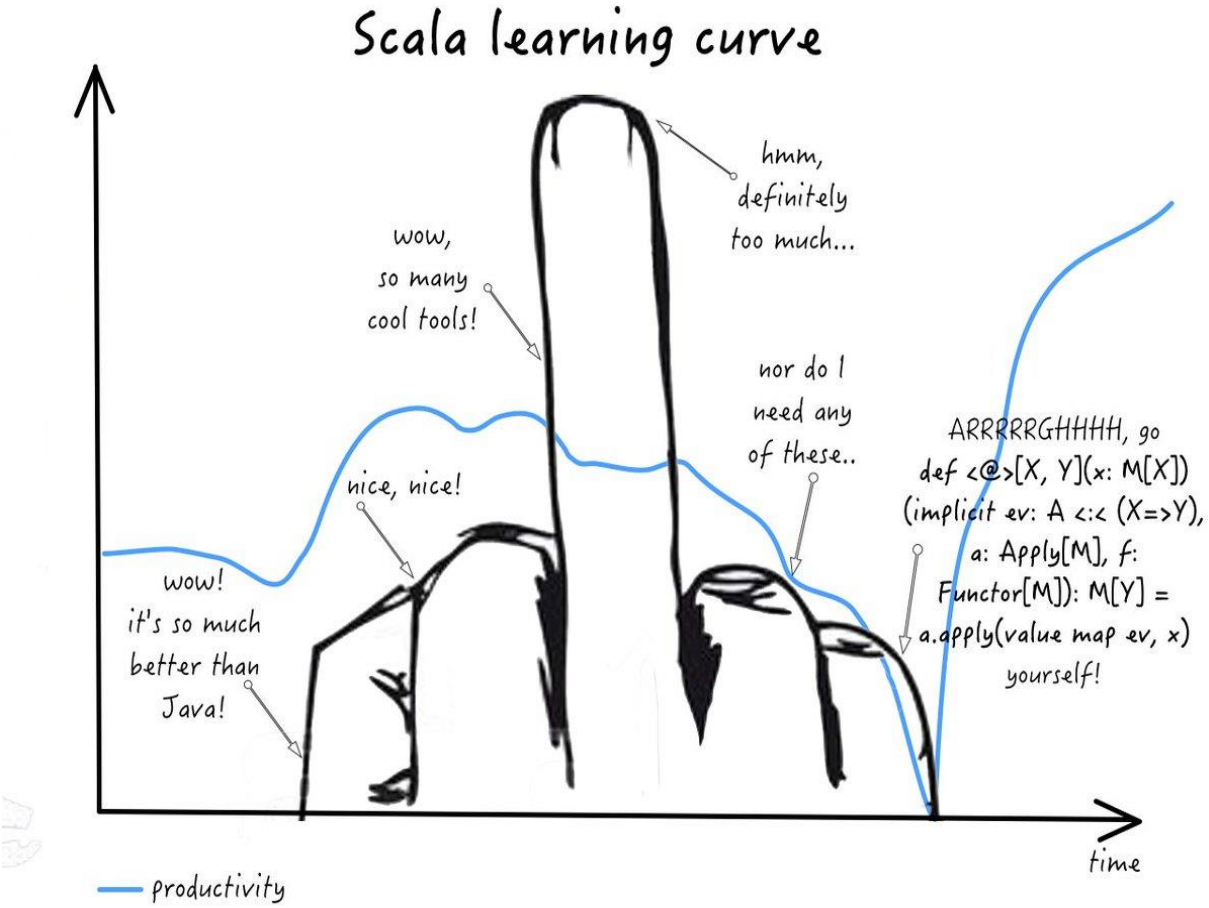
5.2 Демо-кодинг

- На поза-прошлой лекции мы узнали о Byte, Double, Float, Boolean и Char. Сегодня об оставшихся 4-х: **Long**, **Int**, **Short** и **Unit**.
- Арифметические операции это методы.
- В любой непонятной ситуации читай исходный код!



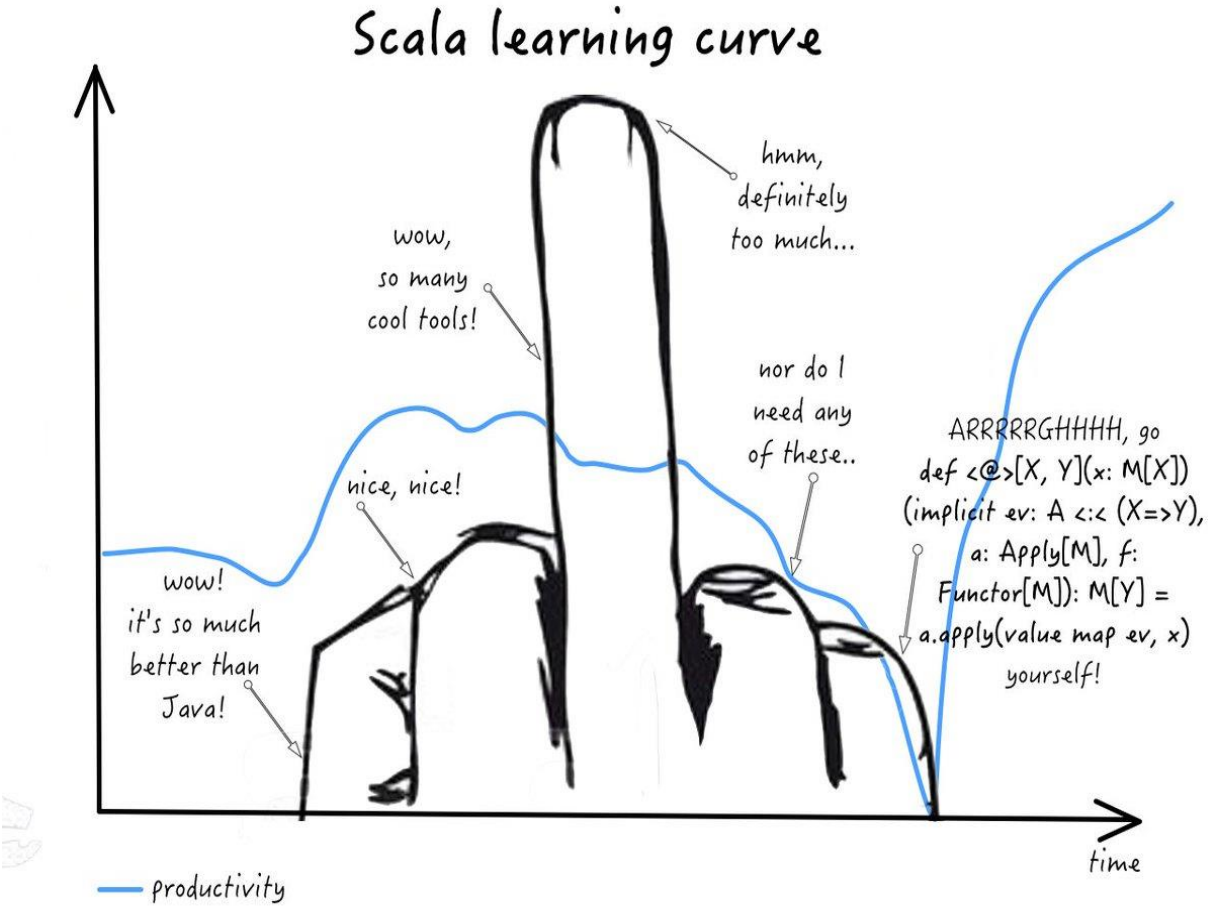
6.1 На сегодня всё

- На следующей лекции вы узнаете как трансформировать объекты с помощью функций и объединять их в коллекции.



6.1 На сегодня всё

- На следующей лекции вы узнаете как трансформировать объекты с помощью функций и объединять их в коллекции.
- Домашки не будет!



6.1 На сегодня всё

- На следующей лекции вы узнаете как трансформировать объекты с помощью функций и объединять их в коллекции.
- Домашки не будет!

- Вопросы?