In Java, all classes (who do not extend some other class) are extending *Object* super class. Kotlin has its super class called ***Any***. From *Any* class we inherit the following methods:

- *equals()*
- *hashCode()*
- *toString()*

Let’s take a look at inheritance example (*Inheritance.kt*):

```
// Inherits Any
open class Human

// Indian extends Human
class Indian : Human()

// Class with non-empty constructor
open class Vehicle(type: String)

// Truck extends Vehicle and it's constructor
class Truck(type: String) : Vehicle(type)

// Train extends Vehicle but has empty constructor.
// Value is passed to parent constructor.
class Train : Vehicle("Civil")

// Another way to extend class
class Bus : Vehicle {
   constructor(type: String) : super(type)
}
```

It is important to note that *class* can’t be extended until it is defined as *Open*! *Open* is opposite to Java’s *final*.

Overriding

To override method in class we must define it as *open* in our *super* class (*Overriding.tk*):

```
open class Engine {
   open fun turnOn() {
      println("Turn on")
   }
}

class SuperEngine : Engine() {
   override fun turnOn() {
      super.turnOn()
      println("Do a special thing!")
```

```
    }
}

val engine = SuperEngine()
engine.turnOn()
```

Console output:

```
Turn on
Do a special thing!
```

A member marked override is itself open! It may be overridden in subclasses. If you want to prohibit re-overriding, use *final*: *final override fun* ...

Abstraction

In Kotlin, as in Java, class and some of its members can be declared abstract. An abstract member doesn't have an implementation in its class. Note that we do not need to annotate an abstract class or function with open – it goes without saying! The same applies for interfaces. Also, we can override a non-abstract open member with an abstract one.

Let's take a look at example *Abstraction.kt*:

```
abstract class Animal {
    abstract fun walk()

    fun run(){
        println("run")
    }
}

open class Cat : Animal() {
    override fun walk() {
        println("walk")
    }
}

abstract class BigCat : Cat() {
    override abstract fun walk()
}
```

In Kotlin we can make class fields abstract as well. Depending if these fields are val or var we must implement proper getter or getter with a setter in class that is extending.

Class companion object

In Kotlin there are no *static* methods! It is recommended to use *package* level functions instead.