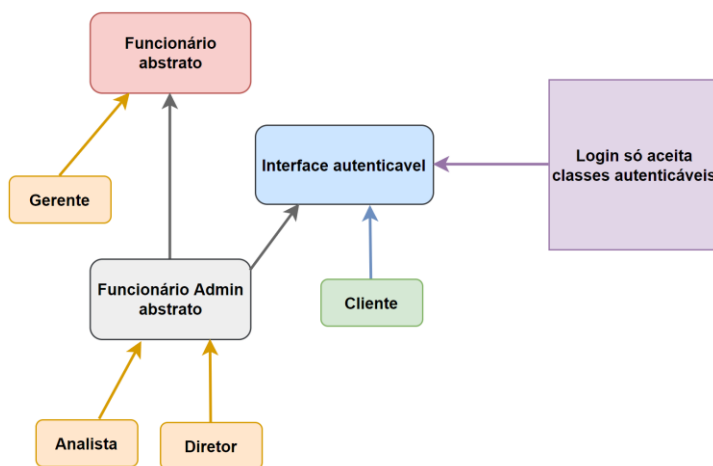




Interfaces são contratos que outras **classes** podem "assinar", quando uma **classe** assina a uma **interface** ela se torna **OBRIGADA** a implementar os **métodos** que a **interface** contém. **Interfaces** em sua maioria só tem **métodos** (*majoritariamente abstratos/sem corpo*), mas também podem conter **propriedades** **OBRIGATORIAMENTE abstratas**.

Interfaces não podem ser instanciadas e não guardam **estado** (*valor nas propriedades e métodos dela, isso porque são abstratas*). O grande ponto da **Interface** é que ela **pode ser implementada por qualquer classe**, e as **classes** podem **implementar mais de uma Interface**, então ela tem uma **flexibilidade muito grande sem um vínculo tão grande quanto herança**.

Seguindo a ideia de que eu tenho **funcionários** e **clientes** e **alguns deles podem se conectar na aplicação de maneira autenticada**, bastaria eu escrever uma **Interface autenticável** e fazer minhas **classes** assinarem seu **contrato**, se tornando **classes autenticáveis**:



Conceito normal de interface, suave. Implementar isso em **Kotlin** é bem simples, **interfaces** tem a palavra reservada de **interface** (*igual java*), para as **classes** assinarem o contrato de uma **interface** só precisa passar **ela** depois de ":" caso seja a primeira, se tiver mais vai dando ",",":

```
interface Autenticavel {
    fun autentica(senha : String) : Boolean
}

class Cliente (
    val nome: String,
    val cpf : String,
    private val senha: String
) : Autenticavel {
    override fun autentica(senha: String): Boolean {
        return when {
            this.senha != senha -> false
            else -> true
        }
    }
}

abstract class FuncionarioAdmin(
    nome: String,
    cpf: String,
    salario: BigDecimal,
    val senha : String
) : Funcionario(nome = nome, cpf = cpf, salario = salario), Autenticavel {
```

Agora no **método de login da aplicação** basta se referenciar a quem quer entrar como sendo um **Autenticavel** (qualquer classe que for Autenticavel é aceita, independente de herança ou não):

```
class SistemaInterno {
    fun entra(autenticavel: Autenticavel, senha : String) : Boolean {
        return autenticavel.autentica(senha)
    }
}

fun main(args: Array<String>) {
    val jarlos = Gerente( nome: "Jarlos", cpf: "12232323", BigDecimal( val: 1000.00), senha: "olaMario")
    val marlas = Diretor( nome: "Marlas", cpf: "12232324", BigDecimal( val: 2000.00), senha: "olamaikon", BigDecimal( val: 0.1))
    val mialdo = Cliente( nome: "Mialdo", cpf: "777666", senha: "Dale")

    val sistemaInterno = SistemaInterno()
    println(sistemaInterno.entra(jarlos, senha: "olaMario"))
    println(sistemaInterno.entra(marlas, senha: "olamaikon" ))
    println(sistemaInterno.entra(mialdo, senha: "Daled" ))
}

true
true
false
```

Interfaces no **Kotlin** também podem ter **implementações padrões** (esquisito né), então eu poderia colocar o **código da autenticação diretamente na interface**, dessa maneira eu não ia precisar sobrescrever em nenhum nível abaixo dela.

Como eu disse antes, dá pra colocar **propriedades abstratas** em **interfaces**, então **para garantir que a senha exista para a implementação padrão**, eu posso obrigar que a **classe concreta** que a **implemente** forneça pra minha interface essa propriedade:

```

interface Autenticavel {
    val senha : String

    fun autentica(senha : String) : Boolean {
        return when {
            this.senha != senha -> false
            else -> true
        }
    }
}

abstract class FuncionarioAdmin(
    nome: String,
    cpf: String,
    salario: BigDecimal,
    override val senha : String
) : Funcionario(nome = nome, cpf = cpf, salario = salario),
    Autenticavel

class Cliente (
    val nome: String,
    val cpf : String,
    override val senha: String
) : Autenticavel

```

O grande problema das implementações padrões é que se você precisar de **UMA PROPRIEDADE** ela vai ser **pública**, **não** tem como restringir ela com modificadores de visibilidade. Então essa prática com uma senha **NÃO DEVERIA SER FEITA**.