



**Abstrações são conceitos** genéricos de **propriedades e comportamentos** que servem apenas para serem **herdados** por **implementações específicas**. Um bom exemplo é um funcionário, funcionários tem cargos, seja *dev*, *diretor*, *PO* etc... todos são **funcionários**, mas ainda sim tem uma **implementação** de **cargo específico**, o que torna a classe **Funcionário** perfeita para ser abstrata.

Uma **classe abstrata** não pode ser instanciada, ela só **serve para ser herdada pelas implementações que podem ser instanciadas**.

Em **Kotlin** abstrações podem ser feitas adicionando a palavra chave **“abstract”**, classes abstratas em kotlin já sabem que são herdáveis, por isso não precisa usar **“open”** nelas:

```
abstract class Funcionario(  
    val nome: String,  
    val cpf: String,  
    var salario: BigDecimal  
) {  
    open val bonificacao: BigDecimal get() = this.salario.multiply(BigDecimal.valueOf(0.1))  
        .setScale(newScale(2), RoundingMode.HALF_UP)  
}
```

```
val luiz = Funcionario()  
// Cannot create an instance of an abstract class
```

```
public constructor Funcionario(  
    val nome: String,  
    val cpf: String,  
    val salario: BigDecimal  
)
```

Membros (*função, propriedade etc*) de **classe** também podem ser **abstratos** e seguem a mesma **ideia de abstração no geral** que é um **comportamento comum e genérico** que deve ser herdado, quem deve **IMPLEMENTAR DE FORMA CONCRETA** esse **membro abstrato** é a classe herdeira.

O role do **membro abstrato** é que não tem mais reutilização de código, cada herdeiro dessa **classe** com membro abstrato é **OBRIGADO** a fazer sua **própria implementação**:

```
abstract class Funcionario(  
    val nome: String,  
    val cpf: String,  
    var salario: BigDecimal  
) {  
    abstract val bonificacao: BigDecimal  
}
```

```
class Gerente(  
    nome: String,  
    cpf: String,  
    salario: BigDecimal,  
    private val senha: String  
) : Funcionario(nome = nome, cpf = cpf, salario = salario) {  
    override val bonificacao: BigDecimal  
    get() {  
        return salario.setScale(newScale(2), RoundingMode.HALF_UP)  
    }  
}
```