




Swift POO



Encapsulamento



Encapsulamento vem de **encapsular**, que em programação orientada a objetos significa separar o programa em partes, o mais isolado possível. A idéia é tornar o software mais flexível, fácil de modificar e de criar novas implementações. O **Encapsulamento** serve para controlar o acesso aos atributos e métodos de uma classe.

The background features a large, light gray diamond-shaped grid pattern. In the top-left and bottom-left corners, there are clusters of small, overlapping triangles in red, white, and dark blue. In the top-right corner, there is a larger, more complex geometric pattern composed of red, white, and dark blue triangles.

public e private

Métodos Get e Set

```
public class Carro {  
    private var modelo: String  
  
    init(modelo: String) {  
        self.modelo = modelo  
    }  
  
    func getModelo() -> String {  
        return modelo  
    }  
  
    func setModelo(modelo: String) {  
        self.modelo = modelo  
    }  
}
```

Métodos Get e Set

```
public class Carro {  
    private var modelo: String  
  
    init(modelo: String) {  
        self.modelo = modelo  
    }  
  
    func getModelo() -> String {  
        return modelo  
    }  
  
    func setModelo(modelo: String) {  
        self.modelo = modelo  
    }  
}
```

Motorista de Uber

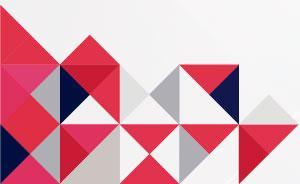


Atributos

- Nome
- Cpf
- Placa do carro
- Saldo em conta

Funções

- Viajar(valor: Float)
- CalculaViagemX
- CalculaViagemPool
- CalculaViagemBlack



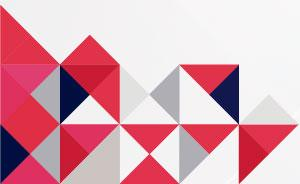
Exercício vendedor



Um vendedor de loja e possui **nome**, **idade** e **cpf**, **saldo em conta** vende em sua loja ternos, vestidos e bonés, definir um método **vender(quantidadeDePecas: Int, tipoDePeca: String)**.


- Cada terno custa 400 reais, caso o cliente compre 3 ou mais ternos, ele recebe 50 de desconto para cada terno.
- Cada vestido custa 900 reais, caso o cliente compre 5 vestidos ele ganha um véu de noiva de brinde.
- Cada boné custa 50 reais, e para cada 2 bonés vendidos, o terceiro é grátis, logo não haverá lucro.

encapsule todos os métodos necessários.






Protocol / Interface



Protocol é um contrato que
você faz com sua classe



```
protocol Dirigivel {  
    func acelerar()  
    func freiar()  
}
```



```
class Carro: Dirigivel {  
    var modelo: String  
    var marca: String  
  
    init(modelo: String, marca: String) {  
        self.modelo = modelo  
        self.marca = marca  
    }  
  
    func acelerar() {  
    }  
  
    func freiar() {  
    }  
}
```

