15/08/2023, 18:15 Encapsulamento

Encapsulamento

Impresso por: Matheus Ferreira Santos

terça-feira, 15 ago. 2023, 18:15

Data:

Site: <u>HackaTruck MakerSpace</u>

Conceitos e Fundamentos: Algoritmos e Programação

Curso: Orientada a Objetos com Swift

Livro: Encapsulamento

15/08/2023, 18:15 Encapsulamento

Índice

1. Encapsulamento

1. Encapsulamento



ENCAPSULAMENTO

Na programação orientação a objetos dizemos que um objeto possui uma interface, ou seja, o que ele conhece e o que sabe fazer. Por meio da interface é possível saber quais serviços podem ser executados e também as mensagens que o objeto recebe. Através do conceito de encapsulamento podemos definir e limitar o acesso por meio de diferentes níveis para as classes, atributos e métodos.

Utilizamos encapsulamento quando queremos definir como nossas classes, propriedades e métodos serão acessados por outras classes ou objetos dentro da aplicação. Este impõe diferentes restrições de acesso direto às informações trazendo mais segurança, integridade aos dados e controle durante o desenvolvimento, pois quando chamamos um método, não é necessário saber o que ele faz, mas sim como chamá-lo. A fim de nos beneficiarmos desse cenário com diferentes possibilidades, utilizamos algumas palavras reservadas para indicar e delimitar quem acessa o que, fazendo com que as classes, atributos e métodos, sejam visíveis somente onde é estritamente necessário.

Existem três níveis de encapsulamento no Swift:

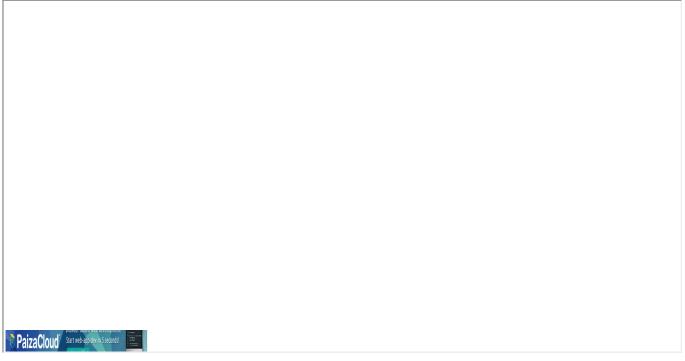
- public Permite acesso a qualquer outro elemento e por qualquer função.
- *internal* Permite acesso apenas dentro da própria classe e nas classes herdeiras. Aprofundaremos sobre este conceito no próximo capítulo (4 Herança).
 - private Permite acesso apenas pela própria classe.

Um exemplo de encapsulamento é a variável saldoBancario de um cliente de banco. Ela não pode ter um acesso público, senão qualquer parte do programa poderia mudar o seu valor. Neste caso, definimos a variável como privada para que seu valor seja alterado usando somente os mecanismos da classe que tem suas devidas travas e regras para cada operação (saque, transferência, extrato, etc.).

Nota: por padrão, o nível de encapsulamento é internal.

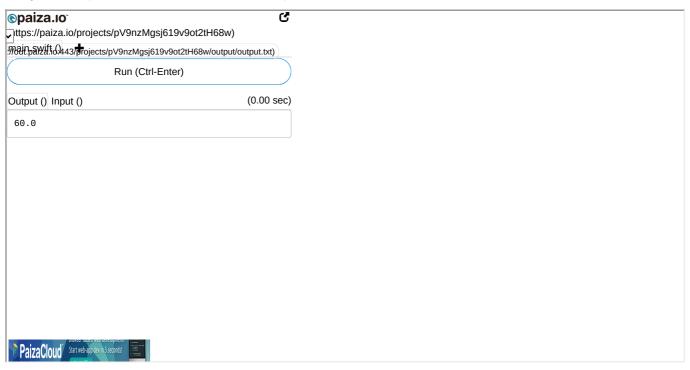
Vamos a um exemplo:

Criaremos uma classe Carro, com uma das propriedades como *private*, que pode ser acessada apenas pela própria classe, e tentaremos alterá-la com uma nova atribuição por meio de um objeto:



Obtivemos um erro, pois nossa propriedade qtdeCombustivel é *private*, e não pode ser alterada por nada que não seja da própria classe.

Dissemos anteriormente que nossos conceitos de encapsulamento podem ser aplicados tanto a atributos como a métodos, então vamos nos aproveitar disso para conseguir fazer essa alteração de valor. Conseguem pensar em uma solução viável para modificar o valor da variável qtdeCombustivel?



Sim, com o uso de um método! Utilizando um método que é acessível externamente conseguimos modificar os valores atribuídos na classe.

Encapsular não é algo mandatório para o funcionamento do programa, mas é uma boa prática para que nossa estrutura seja sólida e nossos objetos sejam seguros do ponto de vista de escrita e leitura, pois ambas as operações só serão feitas de dentro da própria classe se forem declarados como privados.

O encapsulamento do Swift funciona apenas se a classe e sua instância estiverem em arquivos separados. Apenas para efeito de ensino, mantivemos no mesmo simulador.

Olica

Em nosso EAD utilizamos compiladores online, caso algum deles não carregue, basta clicar em RUN .



Caso persista, recarregue a página!

Outras dúvidas ou sugestões entre em contato com contato@hackatruck.com.br.

Faça já os exercícios desde capítulo.

Exercícios – Encapsulamento 🔼