



PRATICANDO

PRATICANDO

1. Leia do usuário a posição (x, y) e os valores de largura e altura. Em seguida, crie um novo Retângulo usando essas informações.

PRATICANDO

1. Altere o exercício 1 e acrescente as chamadas para os métodos definidos.
2. Acrescente ao programa o método dividir: dê um significado ao seu método dividir para o Retângulo. Faça a chamada para esse novo método.

PRATICANDO

4. Altere o exercício 3 e crie métodos de acesso para a classe Retangulo. Use esses métodos de acesso para alterar os atributos do retângulo.

PRATICANDO

5. Implemente a classe Circulo:

- a) Crie um nova classe com o nome Circulo no mesmo projeto onde está a classe Retangulo.
- b) Defina os atributos.
- c) Defina dois construtores.
- d) Defina os métodos de acesso.
- e) Defina os demais métodos (os mesmos métodos da classe Retângulo)
- f) Leia do usuário os valores de x, y e raio e crie um objeto da classe Circulo. Em seguida, chame alguns métodos da classe Circulo para o objeto criado.

PRATICANDO

6. Em um novo projeto (Utilidades), crie uma classe Sequencia para representar uma sequência de números inteiros. A classe deverá ter atributos para representar o início e o fim da sequência. Crie um método para imprimir a sequência de números do início até o fim, inclusive. A impressão da sequência pode ser de 1 em 1 (valor default) ou de p em p. Exemplos:
- Sequência de 2 a 6: 2 3 4 5 6
 - Sequência de 2 a 10 com p=2: 2 4 6 8 10
 - Sequência de 0 a 15 com p=3: 0 3 6 9 12 15
 - Sequência de 0 a 10 com p=4: 0 4 8

PRATICANDO

7. No projeto Utilidades, crie uma classe Data para representar uma data. Para criar uma data é obrigatório informar dia, mês e ano. Crie, também, três métodos:
- ehValida() que deverá retornar true se os valores de dia, mês e ano formarem uma data válida ou false caso contrário.
 - ehBissexto() que deverá retornar true se o ano for bissexto ou false caso contrário.
 - imprime() que deverá imprimir a data com o separador default "/" ou com um separador definido pelo usuário. Caso a data seja inválida, o método deverá imprimir "INVÁLIDA".

Ano bissexto é aquele que é múltiplo de 4 e não é múltiplo de 100 OU aquele que é múltiplo de 400.

The image features a solid blue background. Two thin, dark grey lines intersect diagonally. One line runs from the top-left towards the bottom-right, and the other runs from the top-right towards the bottom-left. They cross each other in the upper-left quadrant of the image.

DOJO

Você foi contratado para desenvolver um sistema simples de gerenciamento de contas bancárias. Para isso, deverá criar uma classe chamada **ContaBancaria**, que represente uma conta de cliente em um banco.

Implemente essa classe obedecendo aos seguintes requisitos:

1.Abstração:

- A classe deve conter os seguintes atributos: titular (nome do cliente), numeroConta (número da conta) e saldo (valor disponível).
- Todos os atributos devem ser definidos de forma apropriada para que representem corretamente as informações de uma conta bancária.

2.Encapsulamento:

- Os atributos devem ser privados, e seu acesso deve ser controlado por meio de métodos get e set, quando necessário.
- O atributo saldo **não pode ser alterado diretamente** por métodos set, devendo ser manipulado apenas pelos métodos de operação da conta.

3.Métodos obrigatórios:

- depositar(valor): adiciona um valor ao saldo. Não deve permitir valores negativos.
- sacar(valor): subtrai um valor do saldo. Para cada saque será debitada também uma taxa de operação no valor de R\$ 1,50. O saque só poderá ser efetuado se houver saldo suficiente para a quantia solicitada mais a taxa da operação.
- exibirDados(): imprime na tela o nome do titular, número da conta e saldo atual.

Você foi contratado para desenvolver um sistema simples de gerenciamento de planos de assinatura de streaming de músicas. Para isso, deverá criar uma classe chamada **PlanoAssinatura**, que represente um plano disponível para clientes desse streaming.

Implemente essa classe obedecendo aos seguintes requisitos:

1.Abstração:

- A classe deve conter os seguintes atributos: nome do plano, limite de quantidade de assinantes possíveis para o plano, quantidade total de assinantes do plano e valor da assinatura do plano.
- Todos os atributos devem ser definidos de forma apropriada para que representem corretamente as informações de um plano.

2.Encapsulamento:

- Os atributos devem ser privados, e seu acesso deve ser controlado por meio de métodos get, quando necessário.
- Os atributos da classe **não poderão ser alterados diretamente** por métodos set, devendo ser manipulados apenas pelos métodos obrigatórios do plano.

3.Métodos obrigatórios:

- adicionarAssinante(): adiciona um assinante na quantidade total de assinantes do plano. Não deve permitir adicionar assinantes acima do limite de quantidade de assinantes possíveis para o plano.
- ofertarDesconto(desconto): subtrai o desconto do valor do plano. Não pode ser possível definir descontos acima de 50% do valor do plano. O desconto não é dado em porcentagem.
- exibirDados(): imprime na tela o nome do plano, quantidade de assinantes e valor do plano.

Você foi contratado para desenvolver um sistema simples de gerenciamento de vagas em um estacionamento privado. Para isso, deverá criar uma classe chamada **VagaEstacionamento**, que represente um vaga disponível para veículos no local.

Implemente essa classe obedecendo aos seguintes requisitos:

1.Abstração:

- A classe deve conter os seguintes atributos: identificador da vaga (ex.: 'A12'), tipo de vaga (moto, carro ou caminhão), valor por hora da vaga, status da vaga (livre ou ocupada).
- Todos os atributos devem ser definidos de forma apropriada para que representem corretamente as informações de uma vaga.

2.Encapsulamento:

- Os atributos devem ser privados , e seu acesso deve ser controlado por meio de métodos get, quando necessário.
- Os atributos da classe **não poderão ser alterados diretamente** por métodos set, devendo ser manipulados apenas pelos métodos obrigatórios do plano.

3.Métodos obrigatórios:

- ocuparVaga(tipoVeiculo): altera o status da vaga para “ocupada”. Caso a vaga já esteja ocupada ou caso o tipo do veículo seja incompatível com o tipo da vaga a ação não deve ser realizada.
- liberarVaga(tempoHoras): altera o status da vaga para livre e retorna o valor total a ser pago, de acordo com o tempo de permanência.
- exibirInformacoes(): imprime na tela o identificador da vaga, tipo da vaga, status atual e valor por hora.

Você está desenvolvendo um sistema de carrinho de compras para um e-commerce e foi encarregado de implementar a classe que representa um produto adicionado ao carrinho. Para isso, deverá criar uma classe chamada **ItemProduto**, que conterá todas as informações e operações relacionadas a um item do carrinho.

Implemente essa classe obedecendo aos seguintes requisitos:

1.Abstração:

- A classe deve conter os seguintes atributos: nome do produto, quantidade comprada, preço unitário, subtotal (calculado automaticamente com base na quantidade \times preço unitário).
- Todos os atributos devem ser definidos de forma apropriada para que representem corretamente as informações de uma vaga.

2.Encapsulamento:

- Os atributos devem ser privados, e seu acesso deve ser controlado por meio de métodos get, quando necessário.
- O valor do **subtotal não deve ser alterado diretamente**, apenas calculado internamente com base nas alterações de quantidade e preço.

3.Métodos obrigatórios:

- atualizarQuantidade(novaQuantidade): atualiza a quantidade do produto. Não permite valores menores que 1. Sempre que a quantidade for alterada, o subtotal deve ser atualizado automaticamente.
- atualizarPreco(novoPreco): atualiza o preço unitário do produto. Não permite valores negativos ou zero. O subtotal também deve ser recalculado automaticamente.
- exibirInformacoes(): imprime o nome do produto, quantidade, preço unitário e subtotal.