EXERCÍCIOS DE FIXAÇÃO DE MÉTODOS E ATRIBUTOS DE CLASSE

Exercício 1. Escreva uma classe *ConversaoDeUnidadesDeTemperatura* que contenha métodos estáticos para calcular a conversão entre diferentes escalas de temperatura. Considere as fórmulas de conversão abaixo:

- De graus Celsius (C) para graus Fahrenheit (F): $F = (9 \times C/5) + 32$
- De graus Fahrenheit (F) para graus Celsius (C): $C = (F - 32) \times +5/9$
- De graus Celsius (C) para graus Kelvin (K): $K = C + 273.15$
- De graus Kelvin (K) para graus Celsius (C): $C = K - 273.15$
- De graus Celsius (C) para graus Réaumur (Re): $Re = C * 4/5$
- De graus Réaumur (Re) para graus Celsius (C): $C = Re * 5/4$
- De graus Kelvin (K) para graus Rankine (R): $R = K * 1.8$
- De graus Rankine (R) para graus Kelvin (K): $K = R/1.8$

Veja que já que existem cinco sistemas de medidas de temperatura, deve haver 20 diferentes métodos de conversão de temperatura. Alguns podem ser escritos indiretamente, por exemplo, para converter de Celsius para Rankine, podemos converter de Celsius para Kelvin e converter esse resultado para Rankine. Opere sobre cada um destes métodos a partir de uma classe principal, sendo que cada método de conversão deve ser chamado diretamente a partir da classe *ConversaoDeUnidadesDeTemperatura.*

Exercício 2. Crie uma classe *senha* que contenha um atributo de classe chamado *senhaAtual* e um atributo de instância chamado *minhaSenha*, ambos do tipo inteiro e ambos privados. Crie ainda um método *get* para cada um destes atributos. No construtor da classe senha, faça com que *senhaAtual* seja atribuído a *minhaSenha* e que em seguida incrementa o atributo de classe. Finalmente, crie uma classe *Principal*, onde deverá ser definido um vetor de objetos do tipo *senha*, de tamanho informado pelo usuário. De forma aleatória, instancie cada um dos objetos do vetor. Ao final, imprima o valor contido no atributo *minhaSenha* de cada objeto.

Exercício 3. Elabore um sistema de controle de supermercado.

O sistema deve ser composto pela classe *Caixa*, que deve possuir como atributos os seguintes atributos de instância: (a) identificação: int; (b) tamFilaAtual: int; (c) faturamento: double (d) status: Operacional. Adicionalmente, a classe *Caixa* deve possuir os seguintes atributos de classe: (a) nroMaximoNaFila: int e (b) classID: int. Crie métodos get para cada um dos atributos. Além dos getters básicos, crie também o seguinte conjunto de métodos:

- setStatus(Operacional status): void → modifica o status atual do caixa. Como resultado, elimina a fila.
- incFila():bool → permite a inclusão de mais um cliente na fila do caixa, caso o tamanho da fila não exceda o *nroMaximoNaFila*.

- Cheio():bool → retorna *TRUE* se a fila chegou a sua capacidade máxima, ou seja, é igual a *nroMaximoNaFila*.
- realizaAtendimento():void → Caso a fila seja maior do que zero, o caixa consome um cliente da fila. Durante este passo, um valor aleatório de compra é realizado a ser contabilizado como faturamento do caixa. Caso não exista ninguém na fila, nada é feito.

Operacional, associado a um dos atributos da classe *Caixa*, é um tipo enumerado. Indica que um caixa está funcionando ou que está parado.

Adicionalmente, elabore a classe *Supermercado*, que deve ter com atributos de instância: (a) um conjunto de caixas; (b) nroClientesEntraram: int; (c) clientesNaoAtendidos: int. A classe Supermercado deve fornecer as seguintes operações a partir de sua elaboração:

- entraCliente(): void → Sinaliza a entrada de um cliente no supermercado que realiza comprar, o que deve ser anotado no atributo *nroClientesEntraram*. Neste método, deve-se procurar por um caixa disponível, ou seja, um caixa que esteja operando e que o tamanho da fila não tenha atingido o máximo. Caso um caixa com esta característica seja encontrado, deve-se incluir mais um a fila do caixa. Caso contrário, deve-se gerar um valor aleatório de compra (R$), anotar como cliente não atendido e incrementar o valor de compra como perdido.
- avanca(): void → faz com que os caixas atendam um cliente em sua fila. Durante o atendimento, deve-se gerar um valor aleatório de compra, o qual deve ser considerado como faturamento.
- listaCaixas():void → lista cada caixa com sua identificação, informando seu status, o faturamento de cada um e o tamanho da fila.
- mudaStatusCaixa(int identificacao): void → permite que um caixa seja parado ou que volte a funcionar. Caso seja posto em funcionamento, esta caixa fica apto a receber novos clientes a partir da nova entrada de clientes. Caso este seja parado, os clientes da fila devem ser realocados conforme disponibilidade dos demais caixas. Caso haja caixa disponível, deve-se incluir na fila. Caso não haja, deve-se contabilizar a perda de faturamento
- valorFaturado(): double → apresenta o valor faturado até o momento de forma global no supermercado
- valorPerdido():double → apresenta o valor perdido até o momento de forma global no supermercado
- caixasDisponíveis: int → apresenta a quantidade de caixas que existem no supermercado
- caixasOperando():int → apresenta a quantidade de caixas que estão operando no momento
- caixasParados():int → apresenta a quantidade de caixas que estão parados neste momento.

Para manipular este sistema, crie uma classe *Principal*, na qual deve-se elaborar um menu que permita realizar cada uma das operações listadas na classe supermercado. Elabore um diagrama de classes para ilustrar tais relações.