Formação Analista Desenvolvedor Java

Exercícios Propostos

Enums, Classes Wrappers e Autoboxing

1 Exercício

Crie uma classe `Matematica` com métodos estáticos que executam algumas operações matemáticas. Todos os parâmetros e tipos de retorno destes métodos devem ser declarados utilizando classes *wrappers* correspondentes aos tipos primitivos associados (e não os tipos primitivos diretamente). Além disso, o código destes métodos deve tirar proveito do *autoboxing* sempre que possível, de forma que o Java fique responsável por "embrulhar" e "desembrulhar" os valores nos tipos correspondentes. Os métodos que devem ser criados são os seguintes:

```
Double converterAngulo(Double angulo, TipoAngulo tipo)
```

Este método converte um ângulo de radianos para graus ou vice-versa. Ele recebe como parâmetro o valor do ângulo e o seu tipo, que pode ser `GRAUS` ou `RADIANOS`. Se o tipo do ângulo for `GRAUS`, o método deve retornar o ângulo em radianos. Já se o tipo do ângulo for `RADIANOS`, o método deve retornar o ângulo em graus.

Dica: Consulte na documentação do Java os métodos `Math.toDegrees()` e `Math.toRadians()`, que já realizam estes processos de conversão.

```
public static Integer somar(Integer n1, Integer n2, Integer n3)
```

Este método recebe uma sequência de 3 números, soma todos eles e retorna o resultado.

```
public static Integer converterBinarioParaDecimal(String numBinario)
```

Este método recebe um número no formato binário e retorna este mesmo número no formato decimal.

Crie também uma aplicação que chame estes métodos de diversas formas para verificar se estão funcionando adequadamente. Para exercitar o *autoboxing*, você deve chamar os métodos acima passando tipos primitivos como parâmetro e atribuir o retorno dos métodos também a tipos primitivos.

2 Exercício

Crie um `enum` chamado `Operacao` e declarado num arquivo próprio. Este `enum` deve conter os seguintes elementos: `SOMA`, `SUBTRACAO`, `MULTIPLICACAO` e `DIVISAO`.

Cada um dos elementos do `enum` deve estar associado a um símbolo do tipo `char`, como por exemplo: `+`, `-`, `X` e `/`. Quando o método `toString()` (sobrescrito da classe `Object`) for invocado para o elemento, este símbolo deve ser retornado.

Além de tudo isto, o `enum` também deve declarar um método `calcular()`, que recebe como parâmetro dois números do tipo `double` e retorna outro número, também do tipo `double`. A lógica deste método depende do elemento no qual ele foi invocado. Por exemplo, se o método `calcular()` for chamado para o elemento `SOMA`, ele deve somar os dois valores e retornar o resultado. A mesma regra vale para as outras operações.