LABORATÓRIO: CLASSES, OBJETOS, MODIFICADORES DE ACESSO E ENCAPSULAMENTO

Última atualização: 25/04/2023

Exercício 1 – Explorando o Encapsulamento

Objetivo do exercício – Explorar o **encapsulamento de objetos**. Implementaremos um exemplo em três versões no intuito de demonstrar o conceito de **ocultamento de informação**.

Este exercício apresenta o exemplo de um sistema para uma **Transportadora** constituída de uma frota de veículos. O exemplo será evoluído ao longo desta e outras listas de exercícios ao longo do curso.

Versão 1 – Sem ocultamento de informação

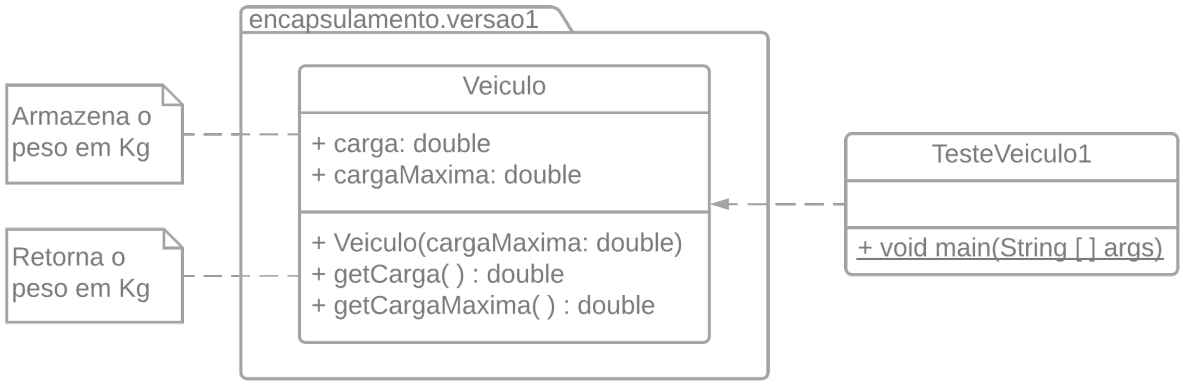

Figura 1. Diagrama classes para a versão 1 do sistema de transportadora de veículos.

Tarefas

- Crie um projeto chamado `Transportadora` em seu IDE predileto. Este projeto será usado para as 3 versões deste exercício.
- Implemente o **diagrama de classes** acima. Use a linguagem de programação **Java**.
- Alasse `TesteVeiculo1` está no pacote *default*. A classe deve ter um método `main` que cria um objeto da classe `Veiculo` e adicione ao veículo algumas caixas com **peso** fornecido. A saída deverá ser idêntica à abaixo:

```
Criando um veículo com carga máxima de 10.000kg.
Adicionando Caixa número 1 (500kg)
Adicionando Caixa número 2 (250kg)
Adicionando Caixa número 3 (5000kg)
Adicionando Caixa número 4 (4000kg)
Adicionando Caixa número 5 (300kg)
A carga do veículo é :10050.0 kg
```

Figura 2. Saída no console do programa criado para a versão 1 do sistema de transportadora de veículos.

- Observe que temos um "*bug*" no programa, tendo em vista que a **carga máxima** do veículo na saída é de 10.000kg. Quando a última caixa é adicionada à **carga** do veículo, o código não checa se a adição de caixa excede a **carga máxima**.

Versão 2 – Ocultamento básico de informação

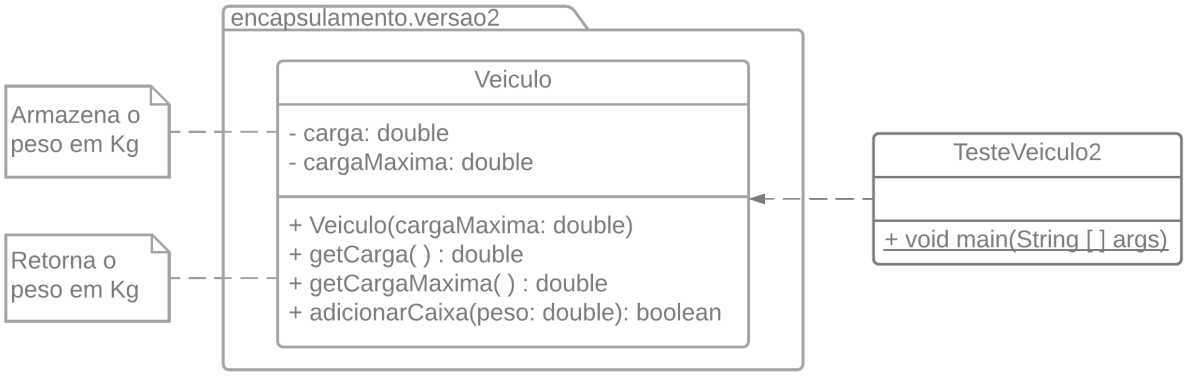

Figura 3. Diagrama classes para a versão 2 do sistema de transportadora de veículos.

Para resolver o problema da primeira versão, esconderemos os dados internos (`carga` e `cargaMaxima`) e forneceremos um método, `adicionarCaixa`, para executar a checagem de sobrecarga do veículo. O método deve verificar que a adição da caixa não violará a **carga máxima**. Se ocorrer uma violação, a caixa é rejeitada retornando o valor `false`; caso contrário o peso da caixa é adicionada à **carga** do veículo e o método retorna `true`.

Tarefas

- Copie o código desenvolvido na versão anterior para um novo pacote confome o diagrama de classes da Figura 3.
- Modifique o código desenvolvido no exercício anterior para refletir o novo **diagrama de classes**.

- A classe `TesteVeiculo2` deve ter um método `main` que cria um objeto da classe `Veiculo` e adicione algumas caixas com **peso** fornecido. **O código não deverá permitir modificar diretamente o atributo carga** como na primeira versão, mas agora deve usar o método `adicionarCaixa`. Este método retorna `true` ou `false`, que é impresso na tela. A saída deverá ser idêntica à saída abaixo

```
Criando um veículo com carga máxima de 10.000kg
Adicionando caixa número 1 (500kg) : true
Adicionando caixa número 2 (250kg) : true
Adicionando caixa número 3 (5000kg) : true
Adicionando caixa número 4 (4000kg) : true
Adicionando caixa número 5 (300kg) : false
A carga do veiculo é: 9750.0 kg
```

Figura 4. Saída no console do programa criado para a versão 2 do sistema de transportadora de veículos.

- Observe que não foi possível adicionar a última caixa.

Versão 3 – Demonstrando o encapsulamento de objetos a partir da alteração da Representação interna de peso de Kg para Newtons

Agora suponha que você precise de alguns cálculos e que os mesmos seriam mais fáceis se a **carga** estivesse medida em **Newtons**.

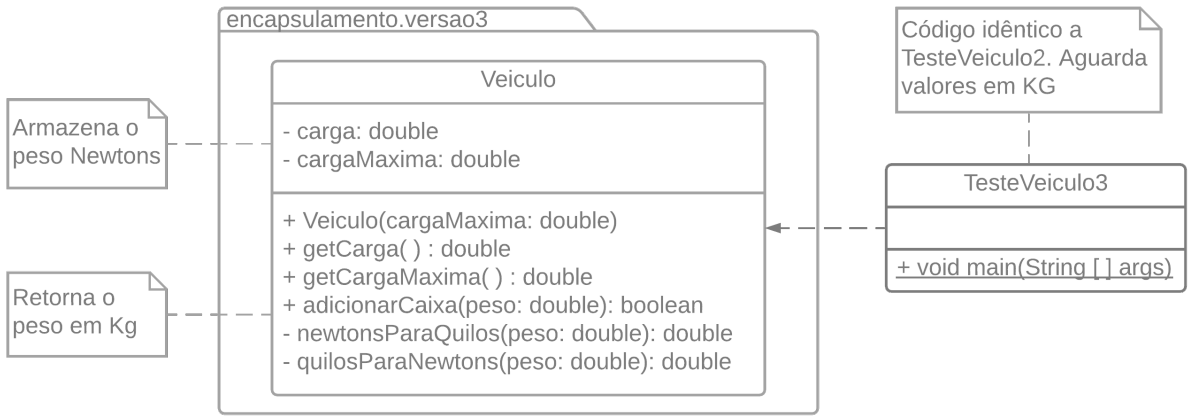

Figura 5. Diagrama classes para a versão 2 do sistema de transportadora de veículos.

Dica: 1 Newton = 9.8kg

Tarefas:

- Copie o código desenvolvido na versão anterior para um novo pacote confome o diagrama de classes da Figura 5.
- Modifique o código desenvolvido no exercício anterior para refletir o novo **diagrama de classes**.
- Modifique a implementação da classe `Veiculo` criada na versão anterior para utilizar os novos métodos de conversão de peso

Agora os dados internos dos objetos `Veiculo` estão em ***newtons*** e os dados externos (passado entre os métodos) ainda estão em **quilogramas**. Para demonstrar isso, crie uma classe chamada `TesteVeiculo3` com um conteúdo IDÊNTICO à classe `TesteVeiculo2`. Execute a classe `TesteVeiculo3`. A saída gerada deve ser:

```
Criando um veículo com carga máxima de 10.000kg
Adicionando caixa número 1 (500kg) : true
Adicionando caixa número 2 (250kg) : true
Adicionando caixa número 3 (5000kg) : true
Adicionando caixa número 4 (4000kg) : true
Adicionando caixa número 5 (300kg) : false
A carga do veiculo é: 9750.0 kg
```

Figura 6. Saída no console do programa criado para a versão 3 do sistema de transportadora de veículos.

A Figura 7 ilustra um **diagrama de sequência UML** para esta versão do exercício. Diferentemente do diagrama de classes que representa uma estrutura estática do seu código, o diagrama de Sequência ilustra a criação e interação entre objetos em tempo de execução, ou seja, representa uma estrutura dinâmica (comportamental) do código.

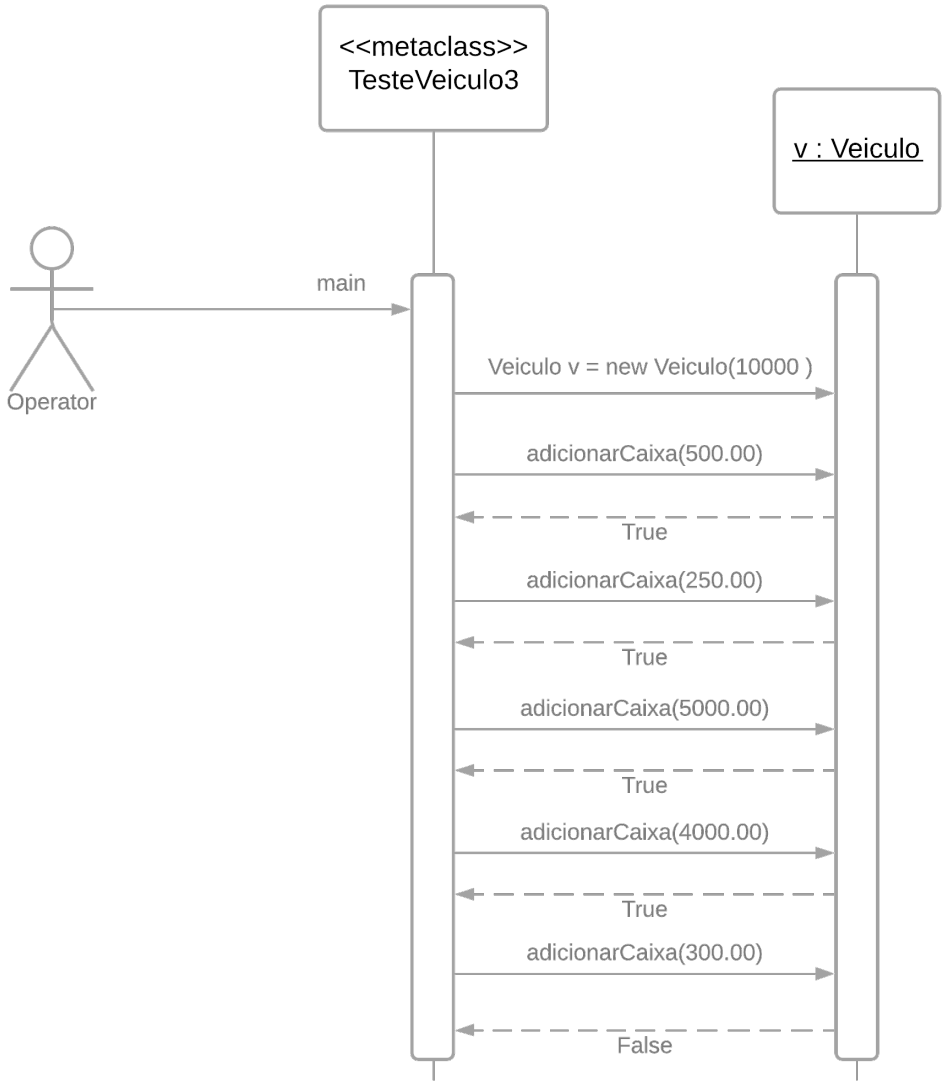

Figura 7 . Diagrama de Sequência UML para versão 3 do sistema de transportadora.

Exercício 2 – Projeto Banco

Objetivos do exercício – Criar a classe `Conta` para o projeto Banco a ser evoluído em exercícios posteriores.

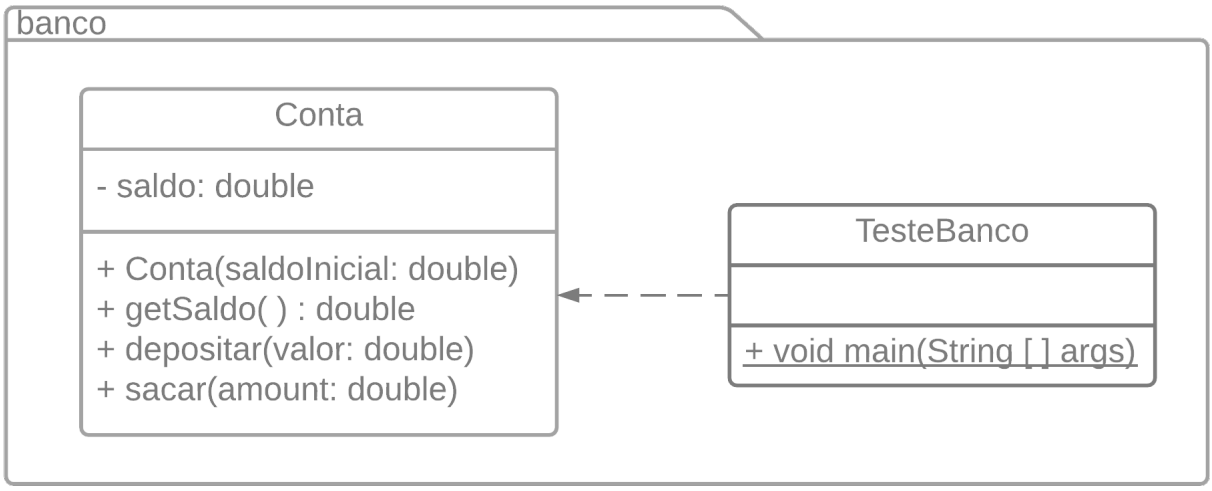

Figura 8. Diagrama de classes UML para um exemplo bancário.

Tarefas

- Implemente o diagrama de classes
- Implemente um programa de testes, chamado `TesteBanco` deve criar uma única conta. Ele inicia o saldo da conta e executa várias transações simples. Finalmente o programa exibe o saldo final da conta. Note que diferentemente dos exercícios anteriores, a classe com método `main` (TesteBanco), não está no pacote *default*.
- Execute a classe `TesteBanco`. Você deverá ver a seguinte saída:

```
Criando uma conta com saldo de R$ 1.000,00.
Sacando R$ 250,00
Depositando R$ 320,50
Sacando R$ 120
O saldo da conta é R$ 950.5
```

Figura 9. Saída no console do programa criado para o projeto Banco.

- Crie um diagrama de Sequência UML para a solução deste exercício.

Exercício 3 – Aprendendo a criar diversos objetos a partir do método `main`

Em todos os exercícios, apenas 1 objeto de cada classe (`Veiculo`, `Banco`) foi criado. Na prática, criaremos diversos objetos (instâncias) de uma mesma classe em um programa em Java. Neste exercício, cabe a você experimentar modificando as classes `TesteVeiculo3` e `TesteBanco` para a criação de diversos objetos, experimentar chamar alguns métodos e observar os resultados produzidos no *console*.

Exercício 4 – Leitura de teclado em linha de comando – Projeto Transportadora

No Exercício 1 deste laboratório, simulamos a adição de diversas caixas em um veículo que possui uma carga máxima suportada. No exercício, tanto a carga máxima suportada pelo veículo quando o peso das caixas individuais sendo adicionadas ao veículo são descritos. (*hard-coded*) dentro do método `main`. Neste exercício você vai aperfeiçoar este código (Exercício 1, versão 3), permitindo a leitura da carga máxima suportada para o veículo e inclusão de caixas com valores fornecidos via um programa em linha de comando em Java que leia do teclado.

Dica #1: O código-exemplo de leitura do teclado em Java (fornecido na aula #06 disponível no Moodle) tem todo o conteúdo necessário para a realização deste exercício:

- Aula #06 - Pacotes. Wrapper classes. Leitura via teclado para programas em linha de comando em Java

Dica #2: Neste e no próximo exercícios você pode assumir que os valores fornecidos pelo usuário via teclado estão corretos com relação a seus tipos e faixa de valores. Por exemplo, se for solicitada a entrada de um valor numérico, supõe-se que o usuário digita um número e não uma `String`, por exemplo. Em Laboratórios futuros iremos aperfeiçoar nosso código com tratamento de exceções e validação de dados.

Dica #3: Você pode usar os mesmos valores usados no exercício 1 deste laboratório para testar a sua versão com leitura via teclado.

Exercício 5 – Leitura de teclado em linha de comando – Projeto Banco

Este exercício é similar ao Exercício 4. A diferença é que neste exercício o discente deverá aperfeiçoar o código desenvolvido no Exercício 2 deste laboratório (Projeto Banco) de forma a permitir entrada de dados via teclado.

Dicas: Valem as mesmas dicas do exercício 4.

Bons códigos !