LISTA DE EXERCÍCIOS: MANIPULAÇÃO DE REFERÊNCIAS, ASSOCIAÇÃO 1:1 ENTRE OBJETOS, E REGRAS PARA IDENTIFICADORES EM JAVA

Última atualização: 26/04/2023

Exercício 1 – Investigando atribuição de referências

Objetivo do exercício – Investigaremos neste exercício variáveis de referência Java, criação de objetos e atribuição de variáveis de referência.

Tarefas:

a. Implemente a classe `Ponto` (em um pacote de sua escolha), conforme diagrama de classes :

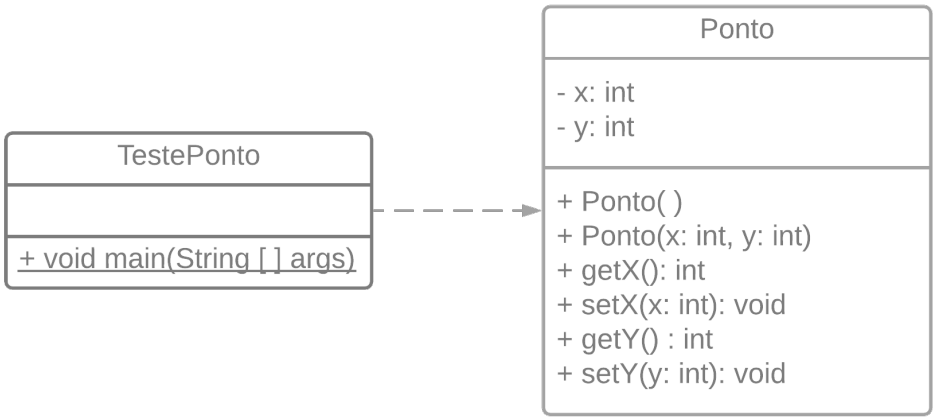

Figura 1. Diagrama de classes UML usado no exercício.

b. Crie um programa com um método `main` chamado `TestePonto`, conforme diagrama de classes. O programa deverá fazer o seguinte:

1. Declare duas variáveis do tipo `Ponto` chamadas `primeiroPonto` e `segundoPonto`
2. Atribua `primeiroPonto` a um objeto do tipo `Ponto criado` a partir do construtor *default* (padrão) da classe `Ponto`.

```
Ponto()
```

3. Atribua `segundoPonto` a um objeto do tipo `Ponto criado` a partir do construtor abaixo. O valor `x` de `segundoPonto` é 400 e o valor `y` de `segundoPonto` é 230.

```
Ponto(int x, int y)
```

4. Inicie os valores `x` e `y` de `primeiroPonto` com o valor 200.
5. Exiba no *console* ambas variáveis `Ponto`. A saída deverá ser similar à abaixo:

```
Coordenadas do primeiro ponto (x,y): (200,200)
Coordenadas do segundo ponto (x,y): (400,230)
```

Figura 2. Saída parcial do exercício.

6. Declare uma nova variável do tipo `Ponto` chamada `outraRefSegundoPonto`.
7. Atribue à `outraRefSegundoPonto` a referência representada pela variável `segundoPonto`.
8. Atribua valores novos para os membros `x` e `y` do objeto referenciado por `outraRefSegundoPonto`.
9. Imprima com instruções System.out.println o conteúdo de `outraRefSegundoPonto`, `segundoPonto` e `primeiroPonto`.
10. Execute `TestePonto`. A saída deverá ser similar à abaixo:

```
Coordenadas do primeiro ponto (x,y): (200,200)
Coordenadas do segundo ponto (x,y): (400,230)

Criação de uma segunda referência para o segundo ponto, chamada outraRefSegundoPonto

Exibindo o conteúdo de todas referências

Coordenadas do primeiro ponto (x,y): (200,200)
Coordenadas do segundo ponto (x,y): (400,230)
Coordenadas do objeto apontado pela referência outraRefSegundoPonto (x,y): (400,230)

Alterando as coordenadas do segundo ponto para (840,350)

Coordenadas do primeiro ponto (x,y): (200,200)
Coordenadas do segundo ponto (x,y): (840,350)
Coordenadas do objeto apontado pela referência outraRefSegundoPonto (x,y): (840,350)
```

Figura 3. Saída completa do exercício.

Importante: Os valores de `segundoPonto` refletem a mudança feita em `outraRefSegundoPonto`, indicando que ambas variáveis se referem ao mesmo objeto `Ponto`. Todavia, `primeiroPonto` não sofreu alteração, o que indica que ela é independente das outras duas variáveis.

Exercício 2 – Criando Contas de Clientes

Objetivo do exercício – Expandiremos nosso projeto `Banco` adicionando uma classe `Cliente`. Um cliente se relacionará com um objeto `Conta`.

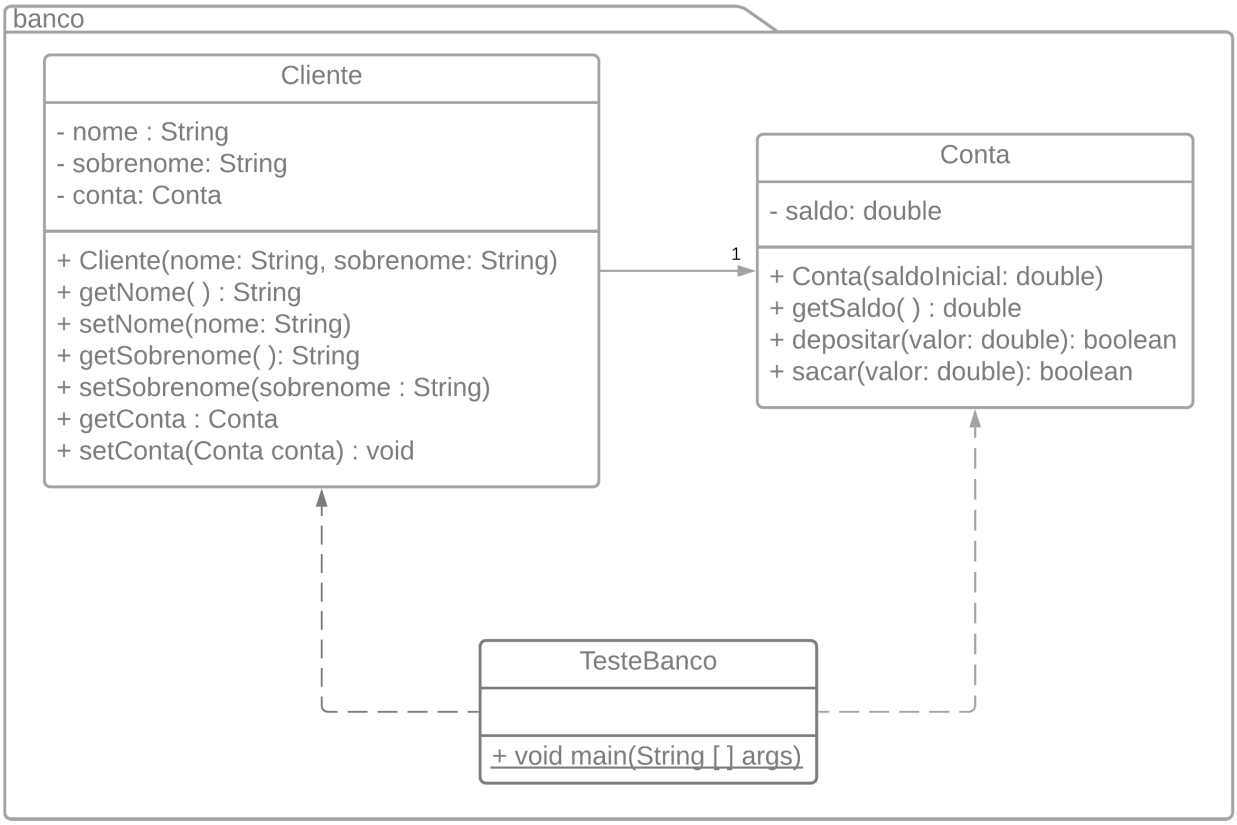

Figura 4. Diagrama de classes do projeto Banco (segunda parte).

Tarefas:

1. Copie os arquivos do projeto `banco` criados na última lista de exercícios.

2. Implemente a classe Cliente conforme diagrama de classes.

3. Modifique a classe `Conta` criada no projeto anterior para retornar `true` ou `false` para os métodos `depositar` e `sacar` conforme diagrama de classes apresentado na Figura 4. Além disso, o méodo sacar deverá checar se o total sendo sacado não é maior que o saldo atual. Se o total for menor que saldo, então subtrair o total de saldo e retorne `true`; caso contrário, retorne `false` e não altere o atributo saldo.

4. Crie uma classe chamada TesteBanco para testar a associação entre Cliente e Conta. Você pode criar um nova classe ou apenas editar o método `main` criado na última lista de exercícios (os arquivos copiados do projeto anterior já possuem uma classe `TesteBanco`). O seguinte caso de teste deverá ser implementado no método `main`:

Caso de testes:

- Criar conta para o cliente Bruno Henrique com saldo inicial de R$ 50.000,00
- Sacar R$ 1.200.00 da conta.
- Depositar R$ 8.525,00
- Sacar R$ 12.800,00
- Sacar R$ 50.000,00

Resultado esperado:

- Ao final das transações, o saldo da conta deverá ser de R$ **44.525,00**

5. Após implementação, execute o programa. A saída deverá ser similar à abaixo:

```
Criando o cliente Bruno Henrique
Criando uma conta com saldo de R$ 50.000,00 para o cliente Bruno Henrique.
Sacando R$ 1.200,00: true
Depositando R$ 8.525,00: true
Sacando R$ 12.800,00: true
Sacando R$ 50.000,00: false
O saldo da conta é R$ 44525.0
```

Figura 5. Saída do programa com transações na conta do cliente Bruno Henrique.

Obs: a parte mais importante do exercício é a associação entre o cliente e a conta. Assim, certifique-se que você tenha explicitamente representado no código com método *main* esta associação:

```
cliente.setConta(conta);
```

Uma forma simples de testar a associação é criar uma instrução para obter a conta associada ao cliente com a chamada abaixo e exibir os seus detalhes no console.

```
Conta conta = cliente.getConta();
```

6. Gere um diagrama de Sequência UML para sua solução.

Exercício 3 – Leitura de teclado em linha de comando – Projeto Banco

No Exercício 2 deste laboratório, simulamos a criação de contas para clientes de um banco. Além disso, no Exercício 2, simulamos várias transações bancárias. No exercício, tanto os dados dos clientes e valores das transações são descritos. (*hard-coded*) dentro do método `main`. Neste exercício você vai aperfeiçoar este código (Exercício 2), permitindo a leitura dos 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 e deste laboratório para testar a sua versão com leitura via teclado.