Curso : Análise e desenvolvimento de Sistemas Disciplina : Desenvolvimento de Software I – Valor 3,0 Carga Horária : 60 Horas Docentes : André Portugal	Faculdade Visconde de Cairu

Exercícios para laboratório – Orientação a Objetos

1. Exercício 1

1.1. Desenvolva uma classe denominada **Lampada** que irá representar uma lâmpada qualquer. A classe deverá possuir como atributos: A **potência** (em watts) da lâmpada e o **estado** da lâmpada (podendo ser "ligado" ou "desligado"). A classe deverá possui como métodos: O método **ligar** (que, ao ser executado, irá ligar a lâmpada), o método **desligar** (que, ao ser executado, irá desligar a lâmpada) e um construtor sem parâmetros (o construtor deverá deixar a lâmpada no estado "desligado").

1.2. Crie uma classe denominada **LampadaDemo** que, ao ser executada, crie um objeto da classe Lâmpada. Armazene algum valor ao atributo **potencia** do objeto e depois ligue e desligue a lâmpada. No final, imprima o valor da potencia da lâmpada.

1.3. Altere a classe **Lampada** para deixar o construtor parametrizado. O construtor deverá receber como parâmetro o estado da lâmpada e o valor da potência. Altere a classe **LampadaDemo** para passar os parâmetros necessários ao construtor da classe.

2. Exercício 2

2.1. Faça uma classe **Conta** que contenha o nome do cliente, o número da conta e o saldo. Estes valores deverão ser informados no construtor para que o mesmo possa inicializar o estado do objeto. Faça um método **depositarValor** e um método **retirarValor**. O método **retirarValor** irá devolver true ou false, dependendo se o cliente pode retirar o valor ou não (o cliente só poderá retirar um valor que for menor ou igual ao saldo atual da sua conta). Faça um método **consultarValorDisponivel** que retorne o valor disponível para saque (nesse caso, o valor do saldo).

2.2. Implemente uma classe denominada **ContaDemo** que deverá possuir um método main. Dentro do método main, deverá ser instanciado um novo objeto da classe **Conta** (item 2.1). Ao criar o objeto, deverão ser passados para o seu construtor o nome do cliente, o número da conta e o seu saldo. Depois de instanciar o objeto, execute o método **depositarValor** , depositando 150 reais, e em seguida o método **retirarValor**, retirando 40 reais da conta. No final, execute o método **consultarValorDisponivel** e imprima o saldo final da conta.

2.3. Agora desenvolva uma classe denominada **ContaEspecial** que herdará da classe **Conta** (criada no item 2.1). A classe **ContaEspecial** deverá ter um atributo adicional denominado **limite**, que representará o limite da conta. A classe deverá ter um construtor que receba o saldo, o nome do cliente, o número da conta e o limite. O construtor deverá chamar o construtor da superclasse (classe Conta), passando os três parâmetros necessários. O construtor deverá também inicializar o valor do atributo **limite**. A classe deverá sobrepor o método **retirarValor**, fazendo com que ele considere o limite da conta ao verificar se pode retirar o valor. Ou seja, poderá ser retirado um valor que for menor ou igual ao saldo da conta mais o seu limite. A classe também deverá sobrepor o método **consultarValorDisponivel**, que retorne o valor disponível para saque (nesse caso, o valor do saldo acrescido do limite).

2.4. Implemente uma classe denominada **ContaDemo2** que deverá possuir um método main. Dentro do método main, deverá ser instanciado um novo objeto da classe **ContaEspecial** (item 8.3). Ao criar o objeto, deverão ser passados para o seu construtor o nome do cliente, o número da conta, o seu saldo e o limite (com os valores respectivos: João, 1234, 100, 500). Depois de instanciar o objeto, execute as seguintes operações:

2.4.1. Execute o método **depositarValor**, depositando 400 reais.
2.4.2. Execute o método **retirarValor**, retirando 50 reais.
2.4.3. Execute o método consultarValorDisponivel e imprima o resultado na tela.
2.4.4. Execute o método **retirarValor**, retirando 900 reais.
2.4.5. Execute o método consultarValorDisponivel e imprima o resultado na tela.
2.4.6. Execute o método **retirarValor**, retirando 100 reais.
2.4.7. Execute o método **depositarValor**, depositando 50 reais.
2.4.8. Execute o método consultarValorDisponivel e imprima o resultado na tela.
2.4.9. Execute o método **retirarValor**, retirando 100 reais.
2.4.10. Execute o método consultarValorDisponivel e imprima o resultado na tela.

3. Exercício 3

3.1. Desenvolva uma classe denominada **Calculadora**. A classe deverá possuir os métodos somar, subtrair, multiplicar e dividir. Cada método deverá possuir dois parâmetros do tipo double, que representarão os números participantes do respectivo cálculo. Cada método deverá fazer o seu respectivo cálculo e retornar o resultado.

3.2. Crie uma classe denominada **CalculadoraDemo** que, ao ser executada, crie um objeto da classe Calculadora. Em seguida, execute os métodos dos objetos, passando os valores necessários para cada método. Imprima os resultados retornados pelos métodos.

3.3. Altere a classe Calculadora para que o método dividir verifique se o divisor é igual a zero (adote como padrão que o segundo parâmetro representa o divisor). Se for, o método deverá imprimir a mensagem "Operação não realizada. Favor entrar com um divisor válido".

4. Exercício 4

4.1. Crie uma classe com as seguintes características:
- Nome da classe: **Produto**;
- Atributos da classe:
  - descricao (texto)
  - preco (real);
  - saldo (inteiro);
- Deve existir um método construtor que inicia os valores dos atributos **descricao**, **preco** e **saldo** (nesta ordem);

4.2. Para testar a classe acima, desenvolva uma classe denominada **ProdutoDemo** com as seguintes características:
- Crie um vetor com 5 objetos da classe Produto;

- Leia o nome de um produto (usando a classe LeitorTeclado), pesquise no vetor de produtos e informe o preço e o saldo. Se o produto não existir, emita a mensagem "Produto inexistente no catálogo". Faça um loop para que possa ser informado vários produtos. O loop deverá ser encerrado quando for digitada a palavra "fim".

5. Exercício 5

5.1. Crie uma classe chamada **Motor** com as seguintes características:

- Atributos da classe:
  - nome (tipo texto);
  - potencia (tipo real);
  - ligado (tipo booleano);
- Um método "ligar()" e um método "desligar()" que alteram respectivamente o atributo **ligado**;
- Um método construtor que recebe e armazena os atributos **nome** e **potencia** e inicia o atributo **ligado** com o valor ***false***.

5.2. Desenvolva uma classe adicional denominada **MotorDemo**, para testar a classe Motor, que faça as seguintes operações:

- Crie um vetor com 5 objetos da classe Motor;
- A seu critério, ligue 3 motores;
- Percorra o vetor de motores e informe as potências dos motores que estiverem desligados.

6. Exercício 6

6.1. Desenvolva uma classe chamada **Refrigerante** com as seguintes características:

- Atributos da classe:
  - nome (tipo texto);
  - preco (tipo real);
  - qtdEstoque (tipo inteiro);
- Um método construtor que recebe e armazena os atributos **nome**, **preco** e **qtdEstoque** (nesta ordem) quando um objeto da classe **Refrigerante** é instanciado;
- Um método **efetuarVenda** que atualiza o estoque do produto após uma venda bem sucedida.

6.2. Desenvolva uma outra classe denominada **RefrigeranteDemo** que, no seu método **main**, faça as seguintes operações:

- Crie 5 objetos da classe Refrigerante com os seguintes dados (que são constantes):

Nome	Preço	Saldo
Cocacola	1,50	40
FantaUva	0,90	10
FantaLaranja	0,90	8
Sprite	0,80	20
Kuat	1,00	100

- Utilizando os métodos da classe Refrigerante, efetue duas vendas dos refrigerantes da sua escolha, sendo que cada cliente pagou com uma nota de R$5,00. Informe o saldo de cada refrigerante e o troco do cliente.

7. Exercício 7

7.1. Desenvolva uma classe chamada **Empresa** com as seguintes características:

- Atributos da classe:
  - nome (tipo texto);
  - capital (tipo real);
  - nFuncionarios (tipo inteiro);
- Um método construtor que recebe e armazena os 3 atributos da classe.

7.2. Desenvolva outra classe denominada **EmpresaDemo** que faça as seguintes operações:

- Crie um vetor com 5 objetos da classe Empresa;
- Percorra o vetor de objetos e informe qual o nome da empresa que tem o menor número de funcionários.
- Percorra o vetor de objetos e informe qual o nome da empresa que tem o maior capital.

8. Exercício 8

8.1. Declare uma classe Java denominada **Livro**, que contém um construtor e um método main. Um livro possui um **título** e **quantidade** de páginas. Use Strings para representar o título. Use inteiros para representar a quantidade de páginas. O construtor deve receber através de parâmetros os dois dados suficientes para criar um livro. O método main deverá criar um array que armazenará quatro livros.

9. Exercício 9

9.1. Defina uma classe Java para representar uma pessoa. Uma pessoa, no nosso caso, possui os seguintes atributos:

- Nome
- Idade
- Altura
- Peso
- Sexo

A classe deve possuir métodos para modificar e acessar cada um dos atributos, de possuir construtores, um método para calcular o IMC (Índice de Massa Corpórea) e um método toString() que retorna o nome da pessoa e uma descrição de sua categoria de acordo com o IMC. O ICM é obtido através da divisão do peso pela altura ao quadrado e é interpretado da seguinte forma:

- IMC <= 18,5: Abaixo do peso normal
- 18,5 < IMC <= 25: Peso Normal
- 25 < IMC <= 30: Acima do peso normal

- IMC > 30: Obesidade

9.2. Desenvolva uma classe denominada **PessoaDemo** que, ao ser executada, criará 4 objetos da classe Pessoa (uma abaixo do peso, uma com peso normal, outra acima do peso e outra com obesidade). Execute o método **calcularIMC** para os quatro objetos e depois execute o método **toString** para os quatro objetos.

10. Exercício 10

10.1. Construa uma classe denominada **Turma** que representará uma turma de alunos. Essa classe terá como atributos um array de strings contendo os nomes dos alunos e um array de floats contendo as notas de cada aluno. A classe deverá possuir um construtor que receberá os dois arrays como parâmetros, para que eles possam ser inicializados. A classe também deverá ter um método denominado **calcularMediaNota** que deverá **calcular** e **retornar** a média das notas dos alunos e um método **imprimirAlunos**, que imprimirá na tela a relação dos nomes dos alunos com as suas respectivas notas.

10.2. Faça uma classe denominada **TurmaDemo** que deverá possuir um método main. Dentro do método main, deverá ser instanciado um novo objeto da classe **Turma** (item 10.1). O construtor da classe Turma deverá receber um array de strings contendo 10 nomes de alunos e um array de floats contendo 10 notas, uma para cada aluno. Depois de instanciar o objeto, execute os métodos **calcularMediaNota** e **imprimirAlunos** a partir do objeto instanciado.

11. Exercício 11

11.1. Desenvolva uma classe denominada **Mercadoria**, que representará um produto comercializado por uma loja qualquer. A classe possuirá os seguintes elementos:

ATRIBUTOS	
Nome do atributo	**Descrição do atributo**
Nome	Nome da mercadoria.
Valor	Valor (em reais) da mercadoria.
qtdEstoque	Quantidade em estoque da mercadoria na loja.

MÉTODOS	
Nome do método	**Descrição do método**
Mercadoria	Método construtor da classe, que deverá receber como parâmetros os valores necessários para inicializar cada atributo do objeto.
Comprar	Método sobrecarregado, que não receberá nenhum parâmetro e que efetuará a compra de um item da mercadoria, abatendo do estoque e retornando o valor da compra.
comprar	Método sobrecarregado, que receberá a quantidade de itens da mercadoria como parâmetro, abatendo do estoque e retornando o valor total da compra.
Comprar	Método sobrecarregado, que receberá como parâmetros a quantidade de itens da mercadoria e um valor de desconto (em percentual), abatendo do

	estoque e retornando o valor total da compra (com o desconto abatido).

12. Exercício 12

12.1. Desenvolva uma classe denominada **Funcionario**, que representará um funcionário de uma empresa. A classe deverá possuir os seguintes elementos:

ATRIBUTOS	
Nome do atributo	**Descrição do atributo**
Nome	Nome do funcionário
Sexo	Sexo do funcionário (‘M’ para masculino e ‘F’ para feminino)
Nível	Nível hierárquico do funcionário na empresa (‘T’ para trainee, ‘P’ para pleno e ‘S’ para sênior)
dependentes	Array de strings que armazenará todos os dependentes do funcionário.

MÉTODOS	
Nome do método	**Descrição do método**
Funcionario	Método construtor da classe, que deverá receber como parâmetros os valores necessários para inicializar cada atributo do objeto.
calcularSalario	Método que deverá calcular e imprimir na tela o salário do funcionário, baseando-se no atributo “nível”. Se o nível possuir valor ‘T’, o salário será R$2.000,00. Se o nível possuir valor ‘P’, o salário será R$3.500,00. Finalmente, se o nível possuir valor ‘S’, o salário será R$5.000,00.
mostrarDependentes	Método que irá percorrer o array de dependentes e imprimir cada um na tela.
validarSexo	Método que deverá ser chamado pelo construtor e que possuirá a função de verificar se o valor a ser armazenado no atributo “sexo” é um valor válido (‘M’ ou ‘F’). Por exemplo, se for passado o valor ‘X’ para o construtor, esse método deverá imprimir “Sexo inválido” e retornar o valor false para o construtor. Se o valor for válido, ele não irá imprimir nada e retornar o valor true para o construtor.

12.2. Desenvolva uma classe denominada **ConsultorDeVendas**, que será uma subclasse da classe **Funcionario**, criada na questão anterior. Todo consultor de vendas é um funcionário que recebe um valor de salário fixo (com base no seu nível hierárquico) mais um valor de bônus em função das suas vendas mensais. Além dos métodos e atributos herdados da classe Funcionario, a classe ConsultorDeVendas deverá possuir os seguintes elementos :

ATRIBUTOS	
Nome do atributo	**Descrição do atributo**
ValorBonusMensal	Valor do bônus mensal do consultor

MÉTODOS

Nome do método	Descrição do método
calcularSalario	Método que irá sobrepor o método herdado pela superclasse. Esse método deverá calcular o salário de forma diferente do método original da classe funcionário. Ele deverá verificar o nível, achar o salário e depois somar ao salário o valor do bônus (armazenado no atributo valorBonusMensal).

13. Exercício 13

13.1. Desenvolva uma classe denominada **Dicionario**, que representará um dicionário de significados, como o dicionário Aurélio. A classe deverá ser composta pelos seguintes elementos :

ATRIBUTOS	
Nome do atributo	**Descrição do atributo**
palavras	Um array de strings contendo todas as palavras do respectivo dicionário.
significados	Um array de strings contendo o significado de cada palavra existente no atributo "palavras". Os índices entre os arrays "palavras" e "significados" deverão coincidir, ou seja, por exemplo, caso o primeiro elemento do array "palavras" seja a palavra "banana", o primeiro elemento do array "significados" deverá conter o significado da palavra "bananas" ("fruta tropical rica em potássio"). De forma geral, podemos definir como regra que : O significado da palavra "palavras[n]" deverá estar no elemento "significados[n]".

MÉTODOS	
Nome do método	**Descrição do método**
Dicionario	Método construtor da classe, que deverá receber como parâmetros os valores necessários para inicializar cada atributo do objeto.
pesquisar	Método sobrecarregado, que não receberá nenhum parâmetro, e que irá retornar o significado da primeira palavra do dicionário.
pesquisar	Método sobrecarregado, que receberá uma palavra (String) como parâmetro, e retornará o seu significado.
pesquisar	Método sobrecarregado, que receberá o índice (int) de uma palavra como parâmetro, e retornará o significado da palavra do índice correspondente do array.
validarPalavra	Método que verifica se uma determinada palavra já não existe no dicionário. Ele receberá como parâmetro uma palavra e irá percorrer o array de palavras para ver se a palavra passada como parâmetro já não existe no array. Caso exista, o método deverá imprimir a mensagem "Palavra já existente no dicionário" e retornar o valor false . Caso a palavra não exista, ele não imprimirá nada e retornará o valor true.

13.2. Desenvolva uma classe denominada **DicionarioDemo**, que deverá criar um objeto da classe Dicionário (classe criada na questão anterior) dentro do seu método main. Observe que, ao criar o

objeto deverão ser passados como parâmetros para o construtor dois arrays : Um contendo as palavras e outro contendo os signficados. Os valores contidos nesses arrays estão descritos abaixo. Após criar o objeto, execute o método pesquisar três vezes: Uma não passando nenhum parâmetro; outra passando uma palavra como parâmetro; outra passando um índice como parâmetro. Finalmente, execute o método **validarPalavra** duas vezes: Uma passando uma palavra existente e outra passando uma palavra que não existe no dicionário.

Palavras	**Significados**
Java	Linguagem de programação
Funcionário	Pessoa que trabalha
Aluno	Pessoa que estuda
Futebol	Modalidade de esporte