UNIVERSIDADE REGIONAL DE BLUMENAU - FURB
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
DISCIPLINA: Programação I

Prof. Aurélio Hoppe

LISTA DE EXERCÍCIOS – CLASSES, MÉTODOS E ATRIBUTOS

1. Os métodos são responsáveis pela _____ dos objetos com o resto do sistema ao passo que os atributos contém as _____ que os definem.

2. Um método pode receber valores? Como são passados? E existem métodos que não retornam valores? Justifique.

3. Escreva um método que recebe como argumento um ano e retorne 1 se este for bissexto e 0 se não for um ano bissexto. Um ano é bissexto se for divisível por 4, mas não por 100. Um ano também é bissexto se for divisível por 400.

4. Escreva um método que some dois valores do tipo int passados como parâmetros. Sobrecarregue o método para que ele consiga também somar valores do tipo double. Cada método retorna o resultado da soma.

5. Escreva um método que retorne uma String com o dia da semana, dado como argumentos o ano, o dia e o mês. Utilize o algoritmo abaixo:

valor = ano + dia + 3(mês -1) -1;
Se mês < 3 então ano = ano -1;
Senão valor = valor – (int)(0.4 x mês + 2.3);
valor = valor + (int) $\frac{ano}{4}$ – (int)(0.75 x (1 + $\frac{ano}{100}$)
valor = valor MOD 7
Se valor = 0 então "domingo"
Se valor = 1 então "segunda-feira"
...

Exemplo de execução:

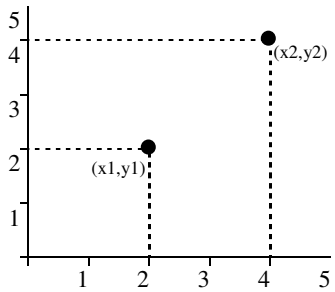
Exemplo de entrada	**Saída para o exemplo de entrada**
calculaDiaDaSemana(2010, 2, 2)	2 – "terça-feira"
calculaDiaDaSemana(2004, 1, 9)	3 – "quarta-feira"

6. Escreva uma classe ModeloDeComputador que encapsule valores que definam a configuração de um microcomputador (tipo de processador, memória RAM, tamanho do disco rígido, tamanho do monitor, por exemplo). Essa classe deve ter um método calculaPreço que calcule o preço do computador como sendo a soma do custo de seus componentes:

- Placa-mãe: R$800
- Opções de processadores: 1.67Mhz a R$700, 2.23Mhz a R$830, 2.56Mhz a R$910
- Opções de memória: 2, 4, 6 ou 8Gb, cada 2Gb custa R$350.
- Opções de disco rígido: 320Gb a R$300, 5000Gb a R$420, 750Gb a R$500.
- Opções de monitor: 19 polegadas a R$320, 21 polegadas a R$520

7. É necessário desenvolver um método para calcular a distância de dois pontos em um sistema de coordenadas cartesiano. Este valor é dado pela equação:

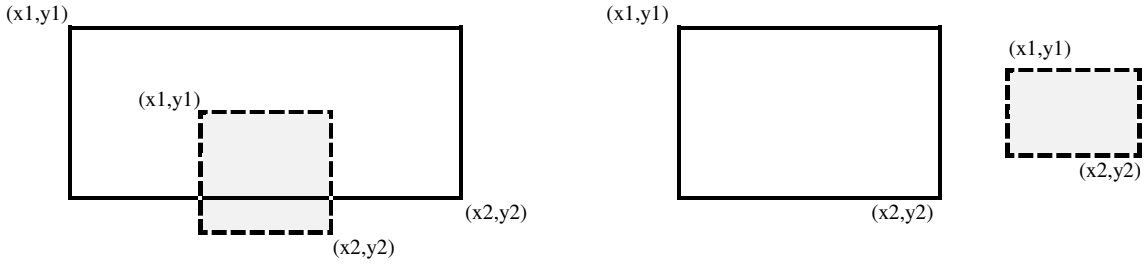
$$d = raiz[(x_1 - x_2)^2 + (y_1 - y_2)^2]$$



onde: x_1 e y_1 são as coordenadas do ponto 1 e x_2 e y_2 as do ponto 2, que devem ser passadas como argumentos para calcular a distância entre os dois pontos. Retorne o valor em double e responda: a distância pode ser negativa? Sim/Não e por quê?

8. Crie uma classe Retangulo para representar um retângulo cujos pontos opostos sejam duas instâncias da classe Ponto. Crie um método para calcular a área do retângulo.

9. Modifique a classe Retangulo (exercício anterior) para que esta contenha um método calculaIntersecção, que recebe como argumento uma outra instância da própria classe Retangulo e calcule um retângulo que é a intersecção do retângulo encapsulado como passado como argumento, retornando uma nova instância da classe Retangulo correspondente à intersecção. **Dicas**: Os pontos do retângulo-intersecção podem ser calculados com regras simples, implementadas através de ifs encadeados. Nem sempre existe intersecção entre dois retângulos. Considere a figura:



no lado esquerdo existem dois retângulos (mostrado sem cores diferentes) que têm intersecção, e, no lado direito, dois que não têm. No caso de não existir intersecção, o método deve retornar null.

10. Desenvolver uma classe chamada Eleitor com os atributos: nome e idade. Implementar os métodos:

Método	**Descrição**
verificar()	Este método retoma uma String e não recebe parâmetro. Deve exibir na tela mensagens de acordo com as seguintes condições: ▪ caso a idade seja inferior a 16 anos, exibir na tela "Eleitor não pode votar" ▪ para idade superior ou igual a 16 anos e inferior ou igual a 65, exibir na tela "Eleitor deve votar" ▪ para idade superior a 65 anos, exibir a tela "Voto facultativo".
imprimir()	Este método retorna uma String e deverá executar o método verificar() como último comando

Desenvolver uma classe Principal. Nesta classe, deve-se criar um objeto da classe Eleitor usando o comando: Eleitor umEleitor = new Eleitor() e, para cada atributo da classe atribuir um valor. Executar o método imprimir() e analisar os valores exibidos na tela.

Exemplo de execução:

Exemplo de entrada	**Saída para o exemplo de entrada**
Nome: Maria Antonieta Idade: 23	Eleitor deve votar

11. Crie uma classe Lutador que contenha os seguintes atributos: Nome e Peso. A classe deve também possuir métodos para modificar e acessar cada um dos atributos, construtores, um método para definir a categoria do lutador e um método imprimir() que fornece o nome do lutador e a descrição de sua categoria, conforme tabela abaixo:

Lutador	
Faixa de Peso	**Categoria**
até 54kg	Pluma
acima de 54, até 60	Leve
acima de 60, até 75	Meio-leve
acima de 75	Pesado

Crie um programa principal para testar a classe Lutador.

Exemplo de execução:

Exemplo de entrada		**Saída para o exemplo de entrada**
Caso teste 1:	Nome: João da Silva Peso: 70kg	O lutador João da Silva está na categoria [Meio-leve]
Caso teste 2:	Nome: Pedro dos Santos Peso: 57kg	O lutador Pedro dos Santos está na categoria [Leve]

12. Desenvolver uma classe chamada Apólice, com os seguintes atributos: nome, idade e valorPremio. A classe Apolice deverá conter os seguintes métodos:

Método	**Descrição**
imprimir()	Este método retorna uma String que mostrará na tela todos os atributos da classe Apólice
calcularPremio()	Este método não retoma valor, mas, deverá calcular o valor do prêmio seguindo as seguintes regras, para a idade: ▪ maior ou igual a 18 e menor ou igual a 25 anos, use a fórmula: valorPremio += (valorPremio * 20) / 100 ▪ superior a 25 e menor ou igual a 36 anos, use a fórmula: valorPremio += (valorPremio * 15) / 100 ▪ superior a 36, use a fórmula: valorPremio += (valorPremio * 10)/100.
oferecerDesconto()	Este método não retoma valor, mas recebe o parâmetro cidade, que irá conter o nome da cidade para o cálculo do desconto. Caso a cidade seja Ilhota, dê um desconto no valor do prêmio de 5%. Caso seja Blumenau, dê um desconto no valor do prêmio de 3%. Caso seja Itajaí, dê um desconto no valor do prêmio de 1%. Para realizar a comparação de strings neste exercício utilizar o método equals().

Desenvolver uma classe chamada PrincipalApolice, com método main(). Nesta classe, deve-se criar um objeto da classe Apolice usando o comando: Apolice umaApolice = new Apolice() e,

a) Para cada atributo da classe atribuir um valor;
b) Executar o método imprimir() e analisar o que será impresso na tela;
c) Em seguida, executar o método calcularPremio();
d) Executar o método imprimir() novamente e analisar o que será exibido na tela ;
e) Executar o método oferecerDesconto() passando como parâmetro a cidade de Blumenau;
f) Executar novamente o método imprimir() e analisar o que será mostrado na tela

Exemplo de execução:

Item	Exemplo de entrada	Saída para o exemplo de entrada
A	Nome: João da Silva Idade: 27 Valor Premio: 900.0	
B		Imprimindo os dados inicializados Nome: João da Silva Idade: 27 Valor Premio: 900.0
C	calcularPremio()	
D		Imprimindo os dados após a execução do método calcularPremio() Nome: João da Silva Idade: 27 Valor Premio: 1035.0
E	oferecerDesconto("Blumenau")	
F		Imprimindo dados após a execução do método oferecerDesconto() Nome: João da Silva Idade: 27 Valor Premio: 1003.95

13. Desenvolver uma classe chamada Estoque com os atributos código, valor e quantidade. Implemente os métodos:

Método	Descrição
adicionarProduto()	Este método não retoma valor e deverá receber como parâmetro a quantidade de produtos a serem somados ao total do atributo quantidade. Este método não fará nada quando o valor do parâmetro for menor ou igual a zero.
verificarDisponibilidade()	Este método deverá retomar verdadeiro/false e receber um parâmetro inteiro. Será retornado TRUE caso existam produtos disponíveis ou FALSE em caso contrário. A existência de produtos disponíveis significa que o atributo tem quantidade é maior que 0 e maior ou igual ao parâmetro recebido.
retirarProduto()	Este método retorna verdadeiro/false e deverá receber como parâmetro a quantidade de produtos a serem retirados. Antes de fazer a retirada do estoque, deve-se verificar se há disponibilidade do produto solicitado. Caso tenha, poderá diminuir do atributo quantidade o valor recebido como parametro. Este método deverá retornar TRUE, caso tenha sucesso na retirada dos produtos. Caso contrário, retomar FALSE.

Desenvolver uma classe chamada PrincipalEstoque, com método main(). Nesta classe, deve-se criar um objeto da classe Estoque e atribuir valores aos atributos código, descrição e valor, deixando a quantidade em estoque zerada. Executar a chamada dos métodos para que seja possível analisar todas as possibilidades que os métodos criados retomam.

Exemplo de execução:

Exemplo de entrada	Saída para o exemplo de entrada
adicionarProduto(1000)	
verificarDisponibilidade(5000)	FALSE – Quantidade insuficiente
retirarProduto(300)	TRUE – Retirada realizada com sucesso

14. Desenvolver uma classe chamada EntradaCinema com os atributos dataFilme, horário, sala e valor. Implemente os métodos:

Método	Descrição
EntradaCinema()	Construtor: com a finalidade de inicializar todos os atributos.
CalculaDesconto()	Este método receberá como parâmetro a data de nascimento do cliente (do tipo Data) e caso seja menor de 12 anos, deve ser dado um desconto de 50% no valor normal.
CalculaDesconto()	Este método receberá como parâmetro a data de nascimento do cliente (do tipo Data) e o número de sua carteira de estudante (do tipo int). Se o estudante tiver idade entre 12 e 15 anos, deve ser dado um desconto de 40%, de 16 a 20 um desconto de 30% e mais que 20 anos um desconto de 20% no valor normal.
CalculaDescontoHorário()	Este método deve dar um desconto de 10% sobre o valor aferido após todas as outras opções de desconto, caso o horário do filme seja antes das 16 horas.
toString()	Este método deve imprimir todos os dados do ingresso.

Desenvolver uma classe chamada PrincipalCinema, com método main(), que leia os dados necessários para instanciar e imprimir o ingresso para clientes normais, menores de 12 anos e estudantes.

15. Deseja-se criar uma classe para controlar a velocidade do carro, cujo diagrama UML é dado abaixo.

Carro
- velocidade: double
+ Carro() + Carro(double) + setCarro(double): void + getCarro(): String + setVelocidade(double): void + getVelocidade(): double + acelerar(double): void + reduzir(double): void

Implemente os métodos:

Método	**Descrição**
Carro()	Construtor sem parâmetros que inicializa o atributo velocidade com o valor 0.0, usando o método setCarro()
Carro (double)	Construtor que inicializa o atributo velocidade com 0.0, usando setCarro()
setCarro(double): void	Método que chama setVelocidade para inicializar o atributo velocidade com o valor recebido como parâmetro
getCarro(): String	Método que usa getVelocidade e depois retorna uma String contendo o conteúdo do atributo velocidade no seguinte formato: A velocidade atual do carro é de <velocidade>
setVelocidade(double): void	Método que modifica o atributo velocidade, conforme parâmetro recebido, desde que não seja negativo
getVelocidade(): double	Método que retorna a informação contida no atributo velocidade
acelerar(double): void	Método que modifica o conteúdo do atributo velocidade para velocidade + parâmetro recebido (usar setVelocidade), mas desde que o valor do parâmetro seja maior ou igual a zero e menor que 20. Senão, imprime a mensagem "Não foi possível acelerar" e mantém o valor atual do atributo velocidade
reduzir(double): void	Método que modifica o conteúdo do atributo velocidade para velocidade - parâmetro recebido (usar setVelocidade), mas desde que o valor do parâmetro seja maior ou igual a zero e menor que 30. Senão, imprime a mensagem "Não foi possível reduzir" e mantém o valor atual do atributo velocidade.

Desenvolver uma classe chamada PrincipalCarro, com método main(). Nesta classe, deve-se criar um objeto a partir da classe Carro e atribuir valores aos atributos. Executar a chamada dos métodos para que seja possível analisar todas as possibilidades que os métodos criados retomam.

Exemplo de execução:

Exemplo de entrada	**Saída para o exemplo de entrada**
carro.acelerar(25.0)	Não foi possível acelerar
carro.acelerar(15.0)	
carro. getCarro()	A velocidade atual do carro é 15.0
carro.reduzir(35.0)	Não foi possível reduzir
carro.reduzir(10.0)	
carro. getCarro()	A velocidade atual do carro é 5.0

16. Deseja-se criar uma classe que simule operações financeiras, cujo diagrama UML é dado abaixo.

ContaCorrente
- nome: String - saldo: double
+ ContaCorrente() + ContaCorrente(String) + ContaCorrente(String, double) + setContaCorrente(String, double): void + setNome(String): void + setSaldo(double): void + getNome(): String + getSaldo(): double + getStatusContaCorrente(): String + fazerDeposito(double): boolean + fazerRetirada(double): boolean + fazerTransferencia(ContaCorrente, double): boolean

Implemente os métodos:

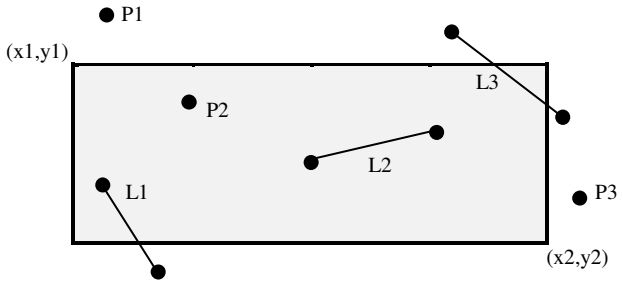
Método	**Descrição**
ContaCorrente()	Construtor sem parâmetros que inicializa os atributos nome e saldo com os valores "SemNome" e 0.0, respectivamente, usando o método setContaCorrente
ContaCorrente(String)	Construtor que inicializa o atributo nome com o parâmetro recebido e o atributo saldo com 0.0, usando setContaCorrente()
ContaCorrente(String, double)	Construtor que inicializa o atributo nome com o primeiro parâmetro recebido e o atributo saldo com o segundo.
setContaCorrente(String, double)	Método que chama setNome e setSaldo para inicializar os atributos nome e saldo com os valores recebidos como parâmetro.
setNome(String): void	Método que modifica o atributo nome, conforme parâmetro recebido
setSaldo(double): void	Método que modifica o atributo saldo, conforme parâmetro recebido
getNome(): String	Método que retorna a informação contida no atributo nome
getSaldo(): double	Método que retorna a informação contida no atributo saldo
getStatusContaCorrente(): String	Método que usa os métodos getNome e getSaldo para retornar uma String contendo o status da conta a partir das informações contidas nos atributos nome e saldo. Formato saída: <nome> seu saldo atual é de <saldo>
fazerDeposito(double): boolean	Método que modifica o conteúdo do atributo saldo para saldo + parâmetro recebido (usar setSaldo). Se a operação for realizada com sucesso, retornar True, caso contrário, retornar False
fazerRetirada(double): boolean	Método que modifica o conteúdo do atributo saldo para saldo - parâmetro recebido (usar setSaldo). Antes de fazer a retirada, deve-se verificar se há disponibilidade (saldo maior ou igual ao parâmetro recebido). Se a operação for realizada com sucesso, retornar True, caso contrário, retornar False
fazerTransferencia(ContaCorrente, double): boolean	Método que modifica o conteúdo do atributo saldo para saldo – parâmetro recibo (usar fazerRetirada). Se o método fazerRetirada retornar False, isso quer dizer que o saldo suficiente (retorne False indicando que a operação não pode ser realizada). Do contrário, efetue a transferência a partir da conta corrente recebida como parâmetro (referência da conta corrente destino), chamando o método fazerDeposito passando o segundo parâmetro recebido como valor a ser transferido.

Desenvolver uma classe chamada PrincipalContaCorrente, com método main(). Nesta classe, deve-se criar dois objetos a partir da classe ContaCorrente e atribuir valores aos atributos. Executar a chamada dos métodos para que seja possível analisar todas as possibilidades que os métodos criados retomam.

Exemplo de execução:

Exemplo de entrada	**Saída para o exemplo de entrada**
ContaCorrente cc = new ContaCorrente("João")	
cc. fazerDeposito(950.0)	
cc.getStatusContaCorrente()	João seu saldo atual é de 950.0
cc.fazerRetirada(1000.0)	False
ContaCorrente contaDest = new ContaCorrente("Maria", 50.0)	
cc.fazerTransferencia(contaDest, 500.0)	True

17. Modifique a classe Retangulo (exercício 8) para que esta contenha dois métodos adicionais: um para verificar se uma instância da classe Ponto passada como argumento está localizada dentro da instância da classe Retangulo, que deverá retornar true se o ponto estiver dentro do retângulo, e outro para fazer o mesmo com uma instância da classe Linha. **Dica**: Para verificar se um ponto está dentro do retângulo, verifique se as coordenadas do ponto estão dentro das coordenadas do retângulo. Considerando a figura:



onde (x1,y1) e (x2,y2) são as coordenadas que definem o retângulo, o ponto P1 estaria fora do retângulo, uma vez que a sua coordenada y é menor do que a menor coordenada y do retângulo. O ponto P2 estaria dentro do retângulo, e o ponto P3 também estaria fora do retângulo. Para verificar se uma linha está dentro ou fora do retângulo, basta verificar os dois pontos que formam suas extremidades: somente se os dois pontos estiverem dentro do retângulo, a linha também estará: na figura acima, a linha L2 está dentro do retângulo, as linhas L1e L3, não.