UFRPE UAST Unidade Acadêmica de Serra Talhada - PE Desde 2006

BSi Bacharelado em Sistemas de Informação

MPOO

Site: https://sigs.ufrpe.br/sigaa/ava/index.jsf

Disciplina: Modelagem e Programação Orientada a Objetos (MPOO)

Profº: Richarlyson D'Emery

LISTA DE EXERCÍCIOS I

Leia atentamente as instruções gerais:

- Ao responder as perguntas desta lista informe, em cada questão, se você baseou sua resposta em alguma pesquisa ou se você respondeu a partir se seus próprios conhecimentos. Sendo assim use: "REFERÊNCIA: Elaboração própria" ou "REFERÊNCIA: citar local da pesquisa".

  Você sabe como referenciar uma fonte? Saiba mais sobre as normas da ABNT em
  - https://www.normasabnt.org/referencias-bibliograficas/
  - ABNT NBR 6023 (https://www.ufpe.br/documents/40070/1837975/ABNT+NBR+6023+2018+%281%29.pdf/3021f721-5be8-4e6d-951b-fa354dc490ed)

- No Eclipse crie um novo **projeto chamado br.edu.mpoo.listaI.SeuNomeSobrenome**, o qual deverá ter **pastas de pacotes** (Menu File -> Submenu New -> Opção Source Folder) para cada questão: questao1, questao2, e assim sucessivamente, contendo todas as respostas da lista.
- A lista envolve questões práticas e conceituais.
- Quando a questão envolver uma discussão teórica utilize um arquivo .txt (Menu File -> Submenu New -> Opção File), por exemplo, questao1.txt
- Quando se tratar de questão prática, na pasta de pacote utilize um arquivo .java (Menu File -> Submenu New -> Opção Class), por exemplo, Classe.java

Responda:

1) Explique:

   1.1) Os porquês de o paradigma da orientação a objetos (POO) ser uma alternativa para o antigo paradigma clássico (imperativo ou estruturado).

   1.2) Como deu o surgimento do POO?

   1.3) Quais as principais diferenças entre o paradigma clássico do POO?

2) Analise a frase:

   "Quando um produto grande é formado por um único bloco monolítico de código, sua manutenção se torna um pesadelo. Mesmo para o autor de tal monstruosidade, tentar depurar o código é extremamente difícil; para outro programador, compreendê-lo é praticamente impossível" (Schach, 2010 p. 177[1]).
   Como o paradigma OO dirime essa problemática?

3) No intuito de relacionarmos o paradigma clássico a disciplina MPOO, responda como fazemos na linguagem de programação C para criar uma solução de um problema de forma que possamos reutilizar o código em outras situações?

4) Qual a relação entre `struct` da linguagem de programação C e o conceito de classe?

5) Em uma demanda de desenvolvimento de software, uma das principais dúvidas surge na modelagem para diferenciar o que se será código ou que apenas trate de uma interação do usuário com o sistema. Sendo assim:

   5.1) Apresente uma estratégia para diferenciar o que será código e o que não será código, ou seja, apenas uma interação com o sistema.

   5.2) Como uma demanda no desenvolvimento de software que será transformada em código pode ser apresentada para uma equipe de desenvolvimento de software?

   5.3) Apresente uma situação em que há pelo menos duas demandas: uma relacionada a interação com o sistema e outra que se trata de uma funcionalidade codificada. Utilize trechos de um documento de requisitos, diagrama de classes e *use case*.

[1] SCHACH, S. R. Engenharia de Software: os paradigmas clássico e orientado a objetos. Porto Alegre: AMGH, 7. ed., 2010. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788563308443/. Acesso em: 09 nov. 2022.

6) A partir da descrição do problema abaixo, apresente o devido diagrama de use case:

[RF01] - Devolver produto	
Ator Principal	Cliente
Atores Secundários	Funcionário e Caixa
Resumo	Este caso de uso descreve as etapas necessárias para que um cliente devolva um produto comprado.
Pré-condições	É necessário existir um produto.
Pós-condições	Escolher opção de devolução
Fluxo Principal	
Ações do Ator	**Ações do sistema**
1. Solicitar a devolução de um produto apresentando o produto.	
	2. Executar a Devolução de um produto
	3. Executar Caso de uso Atualizar estoque
	4. Efetuar troca ou ressarcir o valor do produto
Restrições /validações	
1. O produto só poderá ser devolvido pelo cliente que realizou a compra	
2. O produto só poderá ser devolvido se não apresentar avaria.	
Fluxo Alternativo I – devolução	
Ações do Ator	**Ações do sistema**
	1. Executar caso de uso Efetuar Troca
Fluxo Alternativo II – ressarcimento	
Ações do Ator	**Ações do sistema**
	1. Ressarcir ao cliente o valor pago pelo produto (não é considerado taxas de envio)

[RF02] – Atualizar estoque	
Ator Principal	Vendedor
Resumo	Este caso de uso descreve as etapas necessárias para atualizar o estoque
Pré-condições	É necessário haver uma demanda de atualização
Pós-condições	Verificar situação do produto
Fluxo Principal	
Ações do Ator	**Ações do sistema**
1. Atualizar a base com um produto	
	2. Executar a atualização de estoque
Restrições /validações	
1. Existência de produto requisitado para troca	

[RF03] – Efetuar Troca	
Ator Principal	Vendedor
Resumo	Este caso de uso descreve as etapas necessárias para efetuar a troca de um produto
Pré-condições	É necessário existir uma solicitação de troca de produto
Pós-condições	É necessário dar baixa do produto trocado no estoque
Fluxo Principal	
Ações do Ator	**Ações do sistema**
1. Efetuar a troca de um produto por outro	
	2. Executar o caso de uso Baixar Estoque
Restrições /validações	
1. Existência de produto requisitado para troca	

[RF04] – Ressarcir cliente	
Ator Principal	Caixa
Resumo	Este caso de uso descreve o ressarcimento de um cliente
Pré-condições	É necessário existir uma solicitação de ressarcimento
Fluxo Principal	
Ações do Ator	**Ações do sistema**
1. Realizar a devolução do ressarcimento de valores a cliente.	
	2. Executar o ressarcimento de valor a cliente
Restrições /validações	
1. Existência de uma conta bancária	
Fluxo Alternativo I – ressarcimento em conta	
Ações do Ator	**Ações do sistema**
	1. Executar a transferência de valor em conta bancária de um cliente

Fique atento!

Método construtor é um método que inicializa os atributos da classe. O nome do método construtor deverá ser o mesmo nome da classe. Por exemplo:

```
public class Pessoa{
    String nome;
    int rg;

    public Pessoa (String n, int rg){
        this.nome = n;
        this.rg = rg;
    }
}
```

Toda classe Java possui um construtor **default**, mas uma vez definido um método construtor para a existência do default é necessário explicitá-lo. Por exemplo:

```
public class Pessoa{
    String nome;
    int rg;

    public Pessoa (){}

    public Pessoa (String n, int rg){
        this.nome = n;
        this.rg = rg;
    }
}
```

7) A partir da codificação abaixo

```
//arquivo Classe.java
public class Classe {
    int atributo;
}

//arquivo App.java
public class App {
    public static void main(String[] args) {
            int variavel;
            Classe objeto = new Classe();
    }
}
```

7.1) Podemos dizer que o tamanho da área de memória de `variavel` é o mesmo de `objeto`, uma vez que `objeto` possui apenas um atributo do tipo inteiro?

Antes de responder, pesquise sobre o tamanho de memória para tipos primitivos e para objetos e:

7.2) Explane sobre como é realizada a alocação de memória em JAVA para uma variável do tipo primitiva e para uma instância (objeto).

8) Qual a diferença entre acesso direto e indireto a memória? Como fazemos na Linguagem de Programação C para acessar a memória direta e indiretamente? E como se dá alocação de memória em Java?

9) O que é um método construtor? É possível haver mais de um método construtor? Explique.

10) Em Java o "método construtor" realiza a alocação de uma área de memória. Então:

10.1) Como podemos identificar um método construtor?

10.2) Sempre se devem colocar parâmetros para atualizar todos os atributos em um método construtor?

11) Em diversas situações é preciso fazer o uso de informações `static` e `final`. Qual a diferença entre essas palavras-chaves em um sistema desenvolvido em Java?

12) O que acontece quando é colocado `static` na assinatura de um método de uma classe Java? E em um atributo?

13) A partir da codificação abaixo:

```
public class Classe {
    int valor1;
    static int valor2;
    public static void main(String[] args) {
        int valor3;
        Classe instancia1;
        Classe instancia2 = new Classe();
        //?
    }
}
```

Qual seria a saída para //? assumindo as instruções abaixo. Em caso de erro, explicar o porquê do erro.

Instrução //?	Saída
System.out.println(valor1);	
System.out.println(valor2);	
System.out.println(valor3);	
System.out.println(instancia1);	
System.out.println(instancia2);	
System.out.println(instancia1.valor1);	
System.out.println(instancia1.valor2);	
System.out.println(instancia1.valor3);	
System.out.println(instancia2.valor1);	
System.out.println(instancia2.valor2);	
System.out.println(instancia2.valor3);	

14) A partir das descrições abaixo, proponha uma solução representada em diagrama de classes e codificada em Java. Em todas as questões utilize a classe App para demonstrar a criação de objetos e chamadas de métodos.

14.1) Implemente uma classe chamada "Endereco" que possua atributos para armazenar um endereço. Codifique a criação de um endereço.

14.2) Crie uma classe chamada "Paciente" que possua atributos para armazenar o nome, a idade e sua temperatura. Em Hospital, implemente método para classificar a febre do paciente:

14.3) Implemente uma classe chamada "Aluno" que possua atributos para armazenar o nome, a matrícula e as três notas de um aluno. Em Universidade, adicione métodos para calcular a média das notas e verificar a situação do aluno (aprovado, reprovado ou fazer final). Adote o sistema da UFRPE, em que a menor das notas é descartada.

14.4) Implemente uma classe chamada "Circulo" que possua um atributo para armazenar o raio. Na classe “Geometria” crie métodos para calcular a área e o perímetro de um círculo. Codifique a criação de dois círculos e seus respectivos perímetros e áreas.

14.5) Crie uma classe chamada "Triangulo" com atributos para armazenar os três lados e ângulos internos do triângulo. Na classe “Geometria” crie métodos para verificar se é um triângulo válido e calcular sua área. Codifique a criação dos triângulos: acutângulo, retângulo, obtusângulo, escaleno, isósceles e equilátero. Demonstre os respectivos perímetros e áreas.

14.6) Implemente uma classe chamada "Retangulo" que possua atributos para armazenar a largura e a altura. Na classe “Geometria” implemente métodos para calcular a área e o perímetro de um retângulo. Codifique a criação de dois retângulos e seus respectivos perímetros e áreas.

14.7) A partir das questões 14.4, 14.5 e 14.6 é possível criar uma única classe Geometria que possua serviços para atender o que foi pedido? Se sim, demonstre sua codificação.

14.8) Implemente uma classe chamada "ContaBancaria" que possua atributos para armazenar o número da conta, nome do titular e saldo. Adicione os métodos para realizar depósitos e saques na classe Banco. São regras de negócio:

- RN01: um saque ou depósito só é realizado em uma conta existente.
- RN02: um saque só poderá ser realizado se o saldo de uma conta for igual ou superior ao valor a ser sacado.

14.9)

a) Implemente uma classe chamada "Carro" com atributos para armazenar a marca, o modelo e a velocidade atual do carro. Adicione métodos para acelerar (aumenta a velocidade em 1km/h), frear (reduz a velocidade em 1km/h até o carro estar parado) e exibir a velocidade atual. Codifique um VW fusca em 50 km/h acelerando até 120 km/h e freando até parar. A cada diferença de velocidade deve-se exibir a velocidade atual.

b) Qual a principal diferença entre essa questão e as questões 14.2 a 14.8?

15) **Desafio:** A partir da problemática de autenticação de usuário apresentada na aula da Semana 2, proponha uma solução representada em diagrama de classes e codificada em Java.