**ALUNO: GUSTAVO CLAY DO NASCIMENTO LIMA RGM:15588921**

**Lista de Exercício 03**

**OO/Classes/Objetos/Atributos/Métodos/Construtores**

Para cada novo exercício, sempre que possível, crie um novo arquivo com a extensão .java, declarando o cabeçalho:

**class** ExercicioX {

**public** **static** **void** main(String[] args) {

// seu exercício vai aqui

}

}

Lembre-se de seguir a convenção Java, isso é importantíssimo. Isto é, preste atenção nas maiúsculas e minúsculas, seguindo o seguinte exemplo: nomeDeAtributo, nomeDeMetodo, nomeDeVariavel, NomeDeClasse, ...

1. Explique com as suas palavras o que são Classes e Objetos. O que significa o termo INSTANCIAÇÃO.

Classe: Modelo de atributos e comportamentos que define um conjunto de objetos.

Objeto: instancia de um objeto.

Instanciação: Construção de um objeto.

1. Explique o que é o Principio da Abstração, baseado na *analogia biológica* definida pelo Alan Kay.

O objetivo da abstração é a construção de um modelo para representar a realidade. Para Alan Key seria como se o software funciona-se como um ser vivo, cada célula do corpo é um agente autônomo que interagissem entre si.

1. O que significa e para que serve a referencia **this** neste caso:

**public** MinhaClasse(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

**this** significa auto-referência ao próprio contexto. Serve para identificar que *x (this.x)* e *y (this.y)* é o atributo.

1. Explique o que acontece (com o máximo possível de detalhes) nas duas linhas de código abaixo:

MinhaClasse obj; *//Declaração de uma variável de referencia obj do tipo MinhaClasse*

obj = **new** MinhaClasse(); *//variável obj recebe a instancia de um objeto do tipo MinhaClasse. ”new” aloca memoria para um objeto e devolve um valor de referencia. O operador = copia o valor para a variável. Esse valor é a referencia(endereço) de onde o objeto se encontra na memoria principal.*

1. O que significa Encapsulamento? O que são métodos de acesso e para que servem? Fale sobre a relação entre a Interface e a Implementação de uma classe.

Encapsulamento: esconder todos os membros de uma classe como funcionam as rotinas.

Métodos de Acesso: controlar o acesso aos seus métodos, atributos e construtores através dos modificadores private e public;

Interface é o conjunto de métodos públicos de uma classe, é a única maneira de se comunicar com um objeto dessa classe.

Implementação é a forma como um método realiza a operação.

1. No que diz respeito ao encapsulamento, explique o que significa os modificadores **public** e **private**.

Uma declaração com o modificador public pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.

Os membros da classe definidos como private não podem ser acessados ou usados por nenhuma outra classe. Esse modificador não se aplica as classes, somente para seus métodos e atributos. Esses atributos e métodos também não podem ser visualizados pelas classes herdadas.

1. Determine se cada um dos seguintes exemplos é verdadeiro ou falso, explique por quê.
   1. Por convenção, os nomes de método iniciam com a primeira letra maiúscula e todas as palavras subsequentes no nome iniciam com a primeira letra maiúscula. FALSO – a primeira letra tem que ser obrigatoriamente minúscula no método.
   2. Parênteses vazios que se segue a um nome de um método em uma declaração de método indicam que o mesmo não recebe nenhum tipo de parâmetro. VERDADEIRO
   3. Variáveis ou métodos declarados com o modificador de acesso *private* só são acessíveis à metodos da classem em que são declarados. VERDADEIRO
   4. Qualquer classe que contém *public static void main(String args[])* pode ser utilizada para executar um aplicativo. VERDADEIRO
   5. O número de argumentos na chamada de um método deve corresponder ao número de parâmetros na lista de parâmetros que constam na declaração do método. FALSO – dentro do método podem ter parâmetros que não estão nos argumentos da chamado.
2. Crie uma classe Data com os dados: dia, mês e ano.
3. Crie um construtor e métodos acessórios para os atributos.
4. Crie também um método chamado mostraData que deve imprimir a data armazenada no objeto no formato DD/MM/AAAA.
5. Crie uma classe UsaData que contém um método main. Instancie um objeto da classe Data, preencha os atributos e teste o método mostraData.
6. Altere os métodos acessórios para aceitar apenas dias, meses e anos válidos (um dia deve estar entre 1 e 31, assim como um mês deve estar compreendido entre 1 e 12.)
7. Escreva uma classe chamada classe “Comparavel” que tem como atributo um valor inteiro, um construtor para inicializar o atributo e os métodos éMaiorOuIgual, éMenorOuIgual e éDiferenteDe que recebem um valor do tipo int como parâmetro e retornam true se o valor encapsulado for, respectivamente, maior ou igual, menor ou igual ou diferente do passado como parâmetro.
8. Crie uma classe chamada Calculadora. Esta classe deve conter 4 métodos: somar, subtrair, dividir e multiplicar para calcular e retornar o resultado de cada uma das operações. Crie também uma classe TestaCalculadora que contém um método main. Instancie um objeto da classe Calculadora e teste os métodos.
9. Crie a classe Pedido, considerando que esta deva possuir:
10. Construtor: Deve ter um construtor, cujos parâmetros são: Código do Pedido, Nome do Cliente e Descrição.
11. Atributos

Código de Pedido

Nome do Cliente

Descrição do Pedido

1. Métodos

Modifica o nome do Cliente

Modifica a descrição do pedido

Retorna a descrição do pedido

Retorna o nome do cliente.

**public** **class** Pedido {

//atributos

**public** **int** codigoPedido;

**public** String nomeCliente;

**public** String descricaoPedido;

//Construtor

**public** Pedido(**int** codigoPedido, String nomeCliente, String descricaoPedido) {

**super**();

**this**.codigoPedido = codigoPedido;

**this**.nomeCliente = nomeCliente;

**this**.descricaoPedido = descricaoPedido;

}

//getter e setter

**public** **int** getCodigoPedido() {

**return** codigoPedido;

}

**public** **void** setCodigoPedido(**int** codigoPedido) {

**this**.codigoPedido = codigoPedido;

}

**public** String getNomeCliente() {

**return** nomeCliente;

}

**public** **void** setNomeCliente(String nomeCliente) {

**this**.nomeCliente = nomeCliente;

}

**public** String getDescricaoPedido() {

**return** descricaoPedido;

}

**public** **void** setDescricaoPedido(String descricaoPedido) {

**this**.descricaoPedido = descricaoPedido;

}

}

1. Crie um programa que instancie dois objetos da classe Pedido e que modifique os valores dos atributos: Nome do Cliente e Descrição do Pedido. Antes e após as modificações, deverá imprimir os valores dos mesmos atributos, utilizando os métodos de retorno da classe

**public** **class** Principal {

**public** **static** **void** main(String[] args) {

Pedido Pedido1 = **new** Pedido();

Pedido Pedido2 = **new** Pedido();

System.***out***.println(Pedido1.getNomeCliente());

System.***out***.println(Pedido1.getDescricaoPedido());

System.***out***.println(Pedido2.getNomeCliente());

System.***out***.println(Pedido2.getDescricaoPedido());

Pedido1.setNomeCliente("Joao");

Pedido1.setDescricaoPedido("Sorvetes");

Pedido2.setNomeCliente("Maria");

Pedido2.setDescricaoPedido("Picoles");

System.***out***.println(Pedido1.getNomeCliente());

System.***out***.println(Pedido1.getDescricaoPedido());

System.***out***.println(Pedido2.getNomeCliente());

System.***out***.println(Pedido2.getDescricaoPedido());

}

}

1. Crie uma classe Cliente, considerando que esta deva possuir :
2. Construtor: O Constutor da classe deve possuir como parâmetro somente o código do Cliente
3. Atributos

Código do Cliente

Nome Completo

1. Métodos

Inserção do nome do cliente

Retorna do nome do cliente

Retorna Código do Cliente

**public** **class** Cliente {

//atributos

**public** String nomeCliente;

**public** **int** codigoCliente;

//construtores

**public** Cliente(**int** codigoCliente){

**super**();

}

//getter e setter

**public** **int** getCodigoCliente() {

**return** codigoCliente;

}

**public** **void** setCodigoCliente(**int** codigoCliente) {

**this**.codigoCliente = codigoCliente;

}

**public** String getNomeCliente() {

**return** nomeCliente;

}

**public** **void** setNomeCliente(String nomeCliente) {

**this**.nomeCliente = nomeCliente;

}

}

1. Crie um programa que instancie dois objetos Cliente, atribua o valor ao atributo Nome do Cliente. Antes e após a atribuição, imprima o Nome do Cliente utilizando a função de retorno.

**public** **class** Principal {

**public** **static** **void** main(String[] args) {

Cliente Cliente1 = **new** Cliente();

Cliente Cliente2 = **new** Cliente();

System.***out***.println(Cliente1.getNomeCliente());

System.***out***.println(Cliente2.getNomeCliente());

Cliente1.setNomeCliente("Joao");

Cliente2.setNomeCliente("Maria");

System.***out***.println(Cliente1.getNomeCliente());

System.***out***.println(Cliente2.getNomeCliente());

}

}

SAIDA

null

null

Joao

Maria

1. Implemente uma classe “Livro” que tenha como atributos um titulo, um autor e um preço, além dos métodos acessórios. Crie também dois construtores: um vazio (default) e outro que recebe os três atributos de um livro e preenche os atributos do objeto.

**public** **class** Livro {

//atributos

**private** String titulo;

**private** String autor;

**private** **float** preco;

//construtores

**public** Livro(){

**super**();

}

**public** Livro(String titulo, String autor, **float** preco){

**super**();

setAutor(autor);

setTitulo(titulo);

setPreço(preco);

}

// getter e setter

**public** String getTitulo() {

**return** titulo;

}

**public** **void** setTitulo(String titulo) {

**this**.titulo = titulo;

}

**public** String getAutor() {

**return** autor;

}

**public** **void** setAutor(String autor) {

**this**.autor = autor;

}

**public** **float** getPreco() {

**return** preco;

}

**public** **void** setPreço(**float** preco) {

**this**.preco = preco;

}

}