**Baseado nos conceitos estudados durante as aulas teóricas e prática sobre Programação Orientada a Objetos utilizando a linguagem de programação Java, responda as questões abaixo:**

1. Baseado nos conceitos estudados sobre a linguagem de programação **Java**, descreva os principais conceitos sobre:

* O que são os Construtores das classes? E porque é importante utilizar a sobrecarga de construtores?

Um construtor é um método especial que é chamado automaticamente quando um objeto é criado a partir de uma classe. A principal função de um construtor é inicializar os campos de dados do objeto, A sobrecarga de construtores é importante porque permite que uma classe tenha vários construtores com diferentes parâmetros. Isso é útil quando você precisa criar um objeto com diferentes valores de inicialização, ou quando uma classe tem várias variáveis de instância que precisam ser inicializadas de diferentes maneiras.

* O que são os Arrays? Quais são os seus tipos?

O array é um objeto com uma estrutura de dados que armazena um conjunto de elementos do mesmo tipo primitivo (int, byte, short, long, float, double, boolean, char), os arrays podem ser unidimensionais ou multidimensionais.

* O que é encapsulamento? Descreva abaixo os principais conceitos.

Encapsulamento é um dos princípios fundamentais da programação orientada a objetos que visa proteger os dados e comportamentos internos de uma classe, controlando seu acesso externo por meio de uma interface pública bem definida

Os principais conceitos relacionados ao encapsulamento em Java são:

Atributos privados: são as variáveis ​​de instância declaradas como privadas em uma classe, o que significa que elas não podem ser acessadas diretamente de fora da classe.

Métodos públicos: são os métodos da classe que são declarados como públicos e fornecem uma interface pública para interagir com os atributos da classe. Esses métodos são os únicos meios permitidos para acessar ou alterar os dados encapsulados dentro da classe.

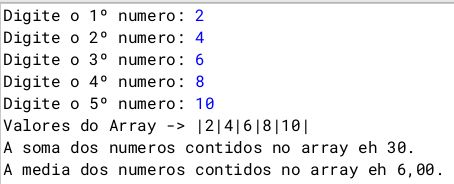
Getters e Setters: são os métodos públicos que permitem obter (get) ou alterar (set) o valor dos atributos privados de uma classe, garantindo assim que o acesso aos dados seja feito de maneira controlada e consistente.

Controle de acesso: é a capacidade de definir o nível de acesso a atributos e métodos de uma classe por meio de palavras-chave como private, protected e public, que determinam quem pode acessar esses elementos.

Ocultação de informação: é a ideia de que os detalhes internos da implementação de uma classe são ocultos do mundo externo, evitando que outras partes do programa dependam desses detalhes e aumentando a capacidade de manutenção e evolução do código.

Segurança e integridade dos dados: o encapsulamento é uma técnica para garantir que os dados sejam armazenados e manipulados de forma segura e consistente, protegendo-os de acessos indevidos ou modificações inadvertidas que possam comprometer a integridade do programa.

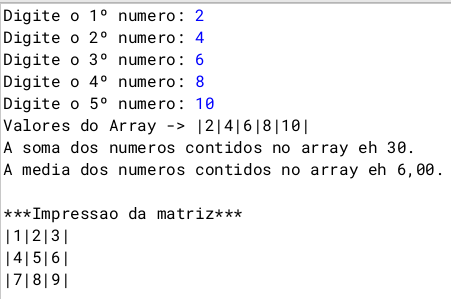
|  |
| --- |
| **Todos** os **códigos** solicitados nas atividades abaixo devem ser criados, compilados e executados usando o **BlueJ**.  Crie um novo projeto no BlueJ chamado “**Lista02-SeuNome**”, e faça as seguintes atividades. |

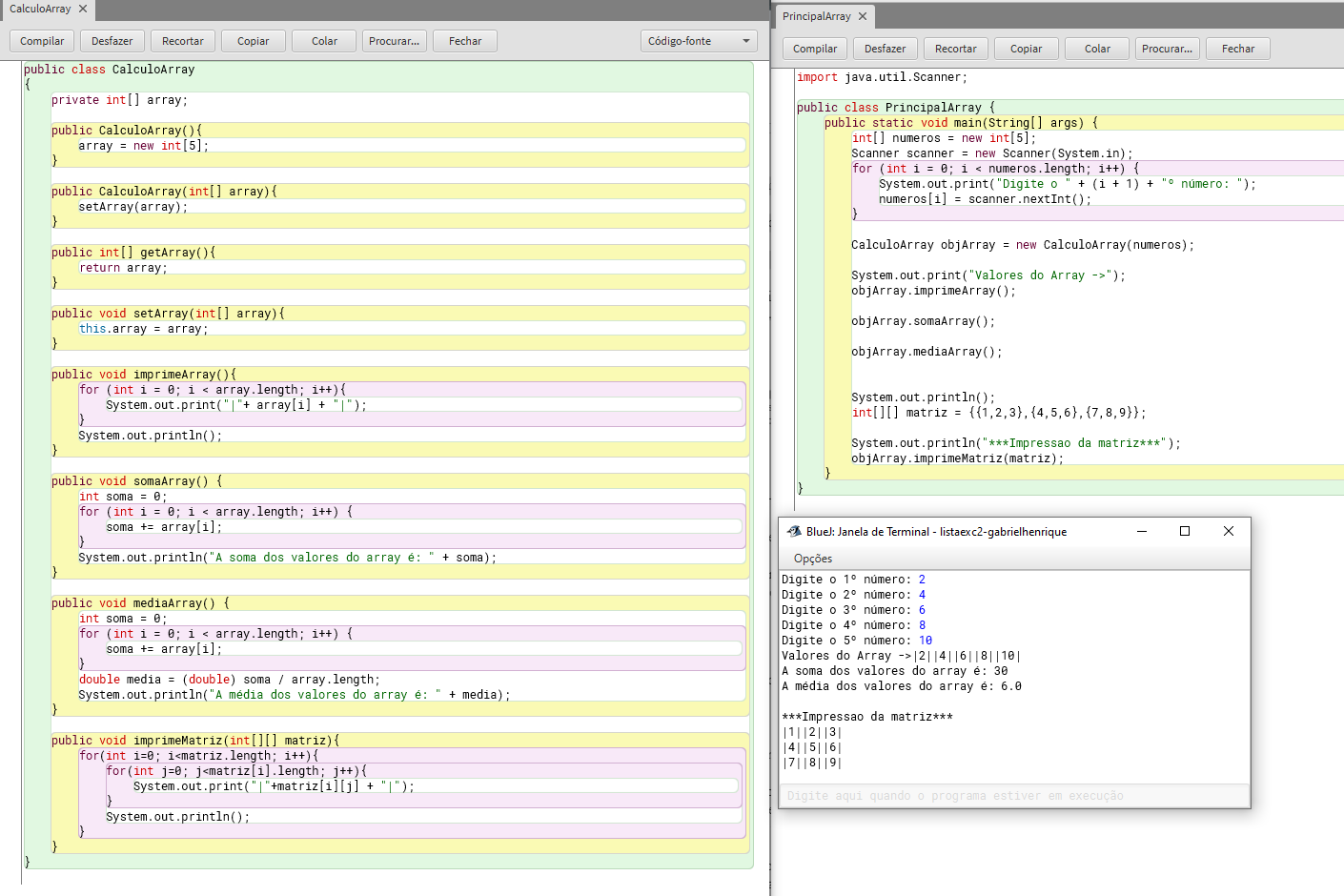
1. Crie um programa que receberá do usuário 5 números inteiros respeitando as seguintes orientações:
   * Você deve criar **DUAS** classes, a “CalculoArray” e “PrincipalArray”;
   * Crie na classe CalculoArray:
     + Um array de inteiros privado //private int[] array;
     + Dois construtores que vão criar 5 posições para o array; //Ex.: public CalculoArray(int[] array){...}
     + Os métodos **get** e **set** do array criado; //Ex.: public int[] getArray(){...} e public void setArray(int[] array){...}
     + Um método para imprimir os dados do array; //public void imprimeArray(){...}
     + Um método que vai imprimir o resultado da soma dos valores do array; //public void somaArray(){...}
     + Um método que vai imprimir a média dos valores do array; //public void mediaArray(){...}
   * Agora, na classe PrincipalArray, você deve armazenar em um array 5 números inteiros digitados pelo usuário;
     + Crie um objeto da classe CalculoArray; //Ex.: CalculoArray objArray = new CalculoArray();
     + Passe o array com os números digitados como parâmetro através do objeto da classe CalculoArray, use o método setArray();
     + Imprima os dados do vetor através do objeto da classe CalculoArray, use o método imprimeArray();
     + Imprima o resultado da soma dos valores do array através do objeto da classe CalculoArray, use o método somaArray();
     + Imprima o resultado da média dos valores do array através do objeto da classe CalculoArray, use o método médiaArray();
   * Segue uma imagem que ilustra os dados inseridos pelo usuário e as impressões pedidas na atividade:
   * Agora, na classe PrincipalArray, crie uma matriz com os seguintes números inteiros:
     + {{1,2,3},{4,5,6},{7,8,9}}
     + Crie na classe CalculoArray um método que vai imprimir os dados de qualquer matriz, que será passada por parâmetro;

//Ex.: public void imprimeMatriz(int[][] matriz){...}

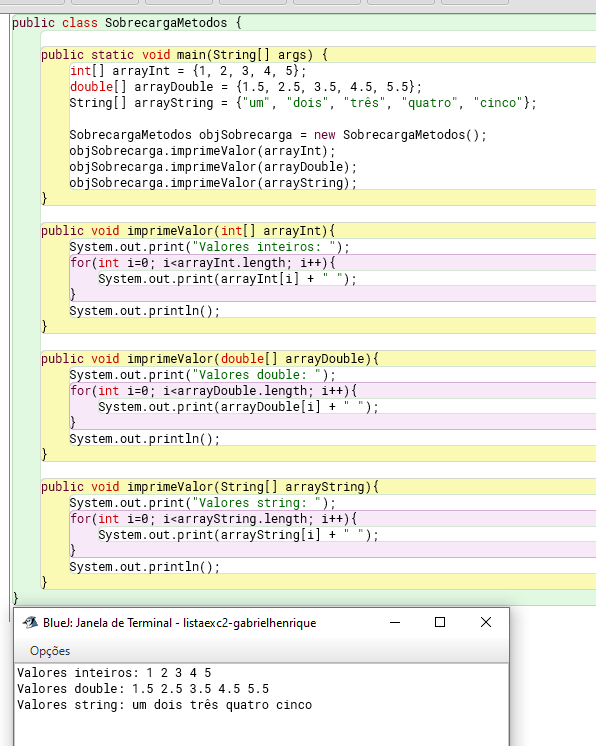
* + - Imprima os valores da matriz criada acima através do objeto da classe CalculoArray, use o método imprimeMatriz();

//Ex.: objArray.imprimeMatriz(matriz);

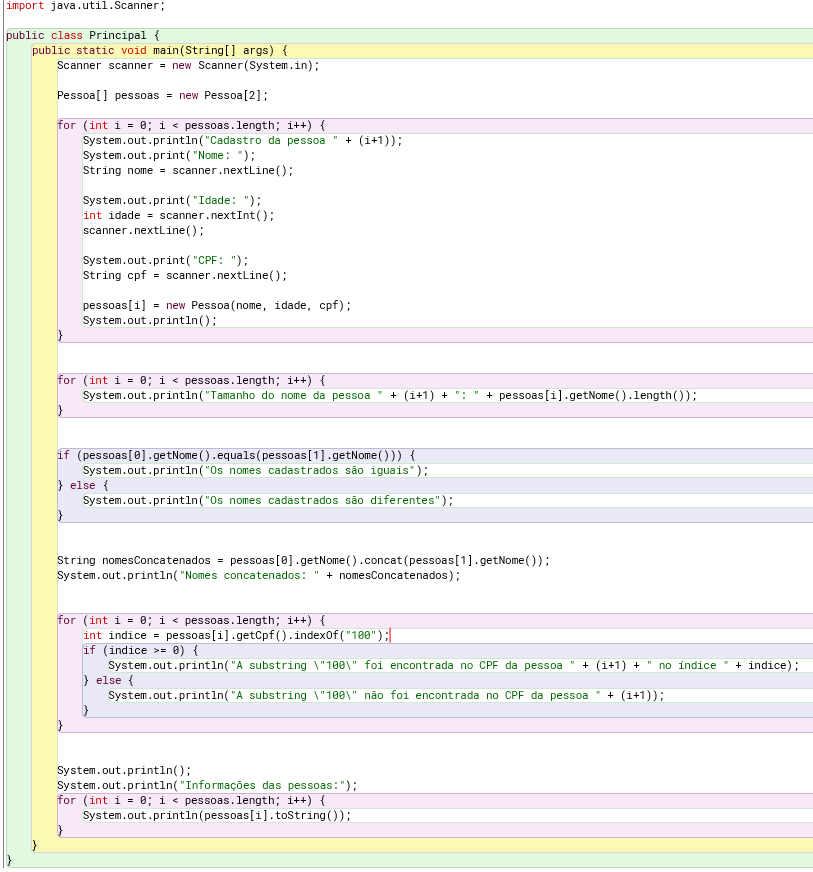
* + Segue uma imagem que ilustra os dados inseridos pelo usuário e as impressões pedidas na atividade:

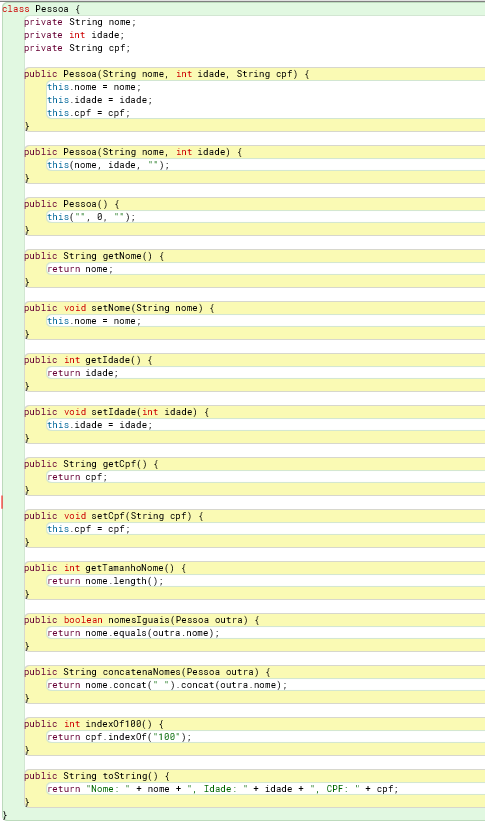


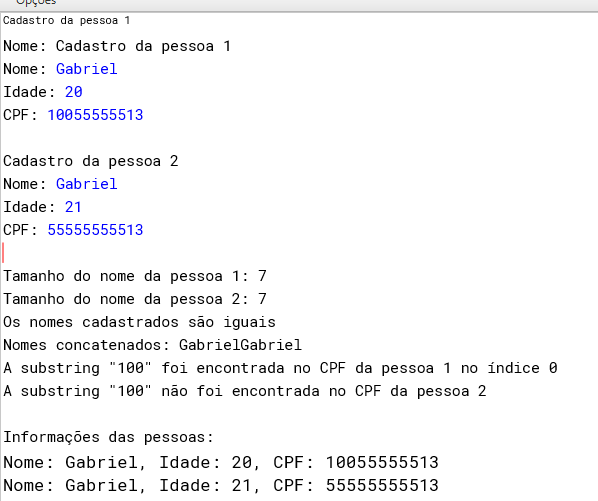
1. Baseados nos conceitos sobre sobrecarga de métodos, faça as seguintes atividades.
   * Crie uma classe chamada SobrecargaMetodos;
   * Crie nesta classe o método principal, que será o responsável por chamar os outros métodos;
   * Implemente nesta classe os **TRÊS** métodos abaixo, e realize a sobrecarga chamando os métodos através do método principal.
     + public void imprimeValor(int[] arrayInt){...}
     + public void imprimeValor(double[] arrayDouble){...}
     + public void imprimeValor(String[] arrayString){...}



1. Crie **DUAS** novasclasses, a primeira chamada **“Principal”** e a segunda chamada **“Pessoa”**.
   * Crie na classe Pessoa:
     + Os atributos nome (String), idade (int) e cpf (String);
     + Crie os métodos gets e set para cada um dos atributos;
     + Crie no mínimo dois construtores, usando a técnica de **sobrecarga de construtores**;
     + Crie o método toString; //Ex.: public String toString(){...}
   * Agora na classe “Principal” crie **UM** array com **DUAS** posições (objetos) do tipo “Pessoa”, e através da classe Scanner() receba do usuário e passe por parâmetro para os devidos métodos da classe “Pessoa” os valores dos atributos: nome, idade e cpf;
   * Após armazenar os dados digitados pelo usuário, chame o método toString usando os dois objetos criados. Com isso, as informações armazenadas devem ser exibidas na tela.
   * Ainda na classe Pessoa, use os dois objetos criados (array) e os métodos gets para:
     + Imprimir o tamanho dos nomes cadastrados em cada objeto do array, use o método **length()**;
     + Verificar se os nomes cadastrados em cada objeto do array são iguais, use o método **equals()**;
     + Imprimir a String resultante da concatenação entre os nomes dos dois objetos cadastrados, use o método **concat()**;
     + Verifique se existe a ocorrência da substring “100” e em qual índice ela inicia nos CPFs dos dois objetos cadastrados, use o método indexOf







**Boa Atividade!!!**