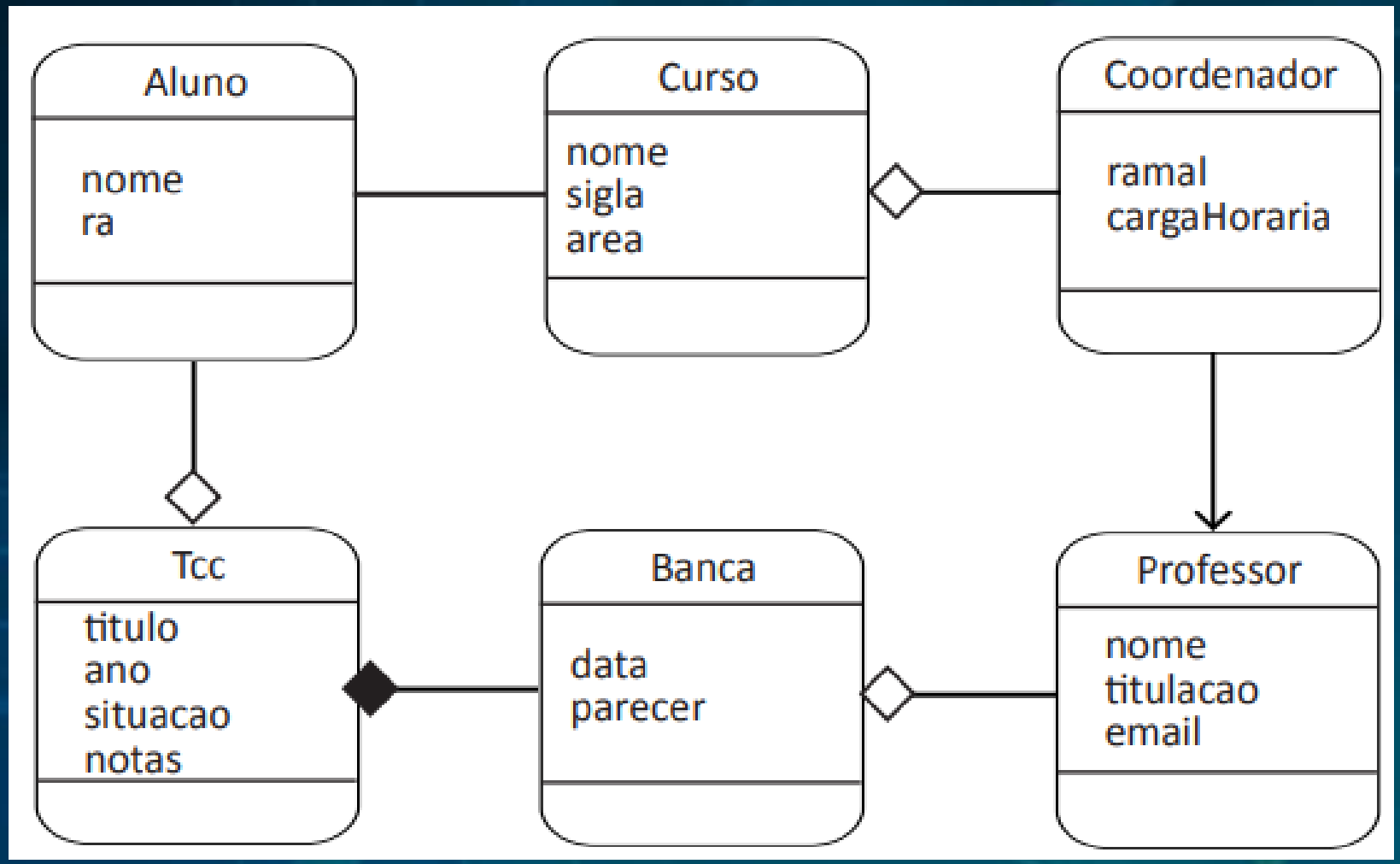
TÉCNICAS DE PROGRAMAÇÃO I

Construindo um novo projeto em BlueJ

JEANE A. MENEGUELI

DIAGRAMA DE CLASSES DE SISTEMA DE ORIENTAÇÕES E DEFESA DE TCC

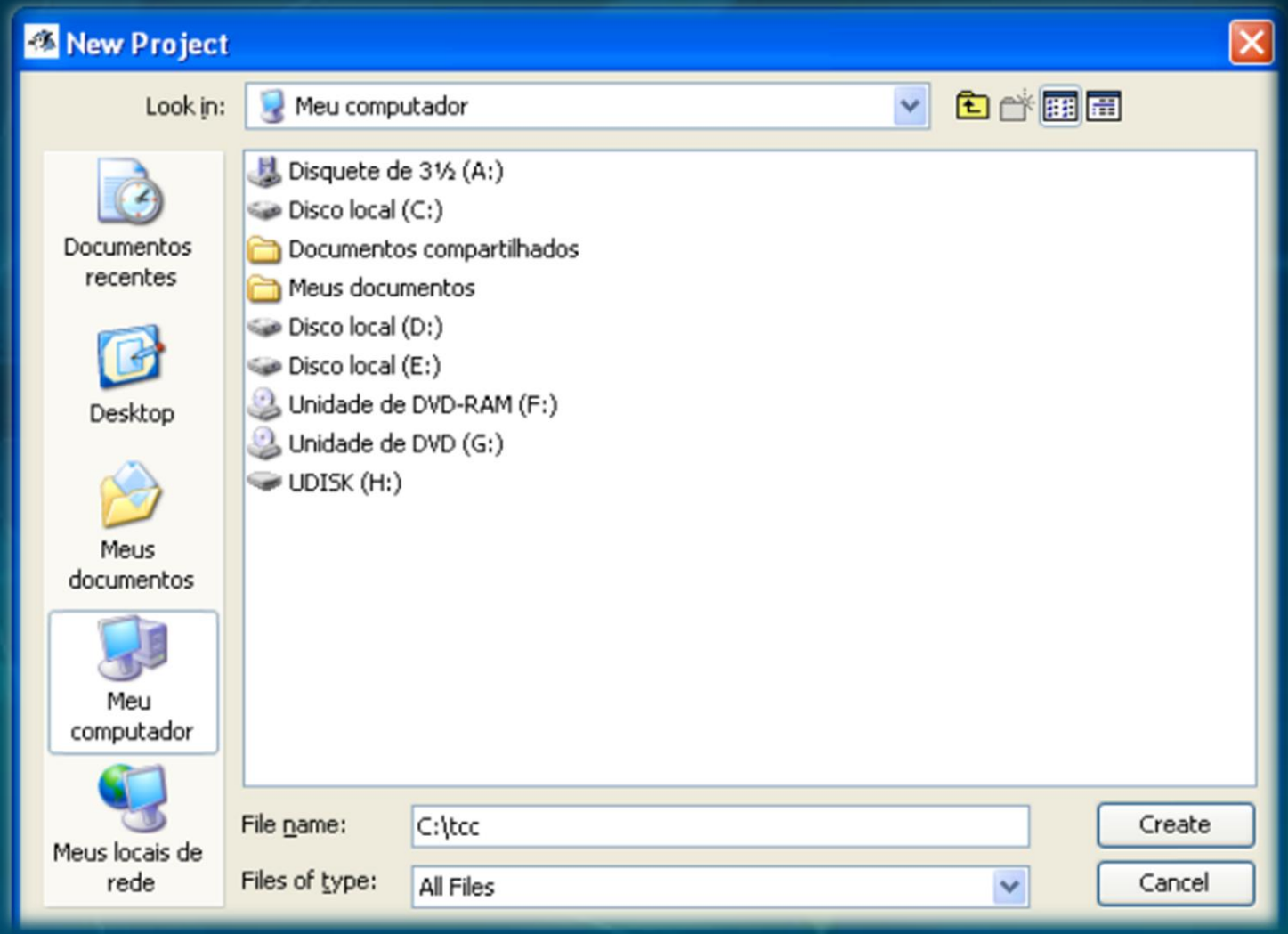


CRIANDO UM NOVO PROJETO

- Ao executar a ferramenta BlueJ, de início, serão exibidas as classes que foram criadas dentro do seu primeiro projeto.
- Como regra-padrão, o BlueJ sempre abre o último projeto que foi utilizado.
- Para não confundirmos a nova implementação com as classes já existentes, vamos definir um novo projeto chamado "***TCC***".
- Você pode fechar o projeto atual acessando o menu "Project" (Projeto) e escolhendo a opção "Close" (Fechar). Essa ação resultará no fechamento do projeto atual e exibirá a tela inicial do BlueJ (tela cinza).

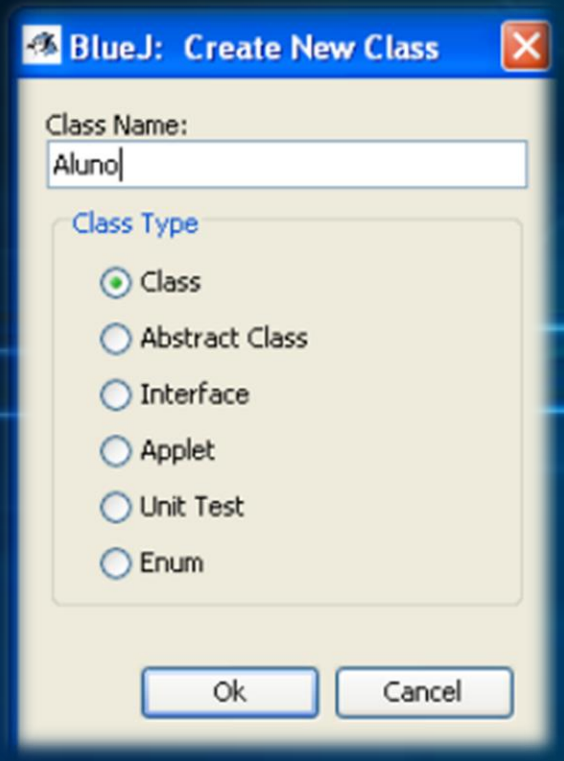
CRIANDO UM NOVO PROJETO

- escolha a opção "New Project“ (Novo Projeto). Ao realizar essa operação, será exibida uma nova janela (Figura), em que poderemos escolher a localização do novo projeto. Você pode optar por criar o projeto em qualquer pasta de seu computador.
- Como exemplo, vamos criar um novo projeto chamado “tcc” na pasta raiz do disco local (C:).
- Para isso, selecione o drive C e digite o nome do projeto "tcc" na opção "File Name“ (Nome do Arquivo), conforme exemplo exibido na Figura. Para confirmar a criação do projeto, clique no botão "Create” (Criar).

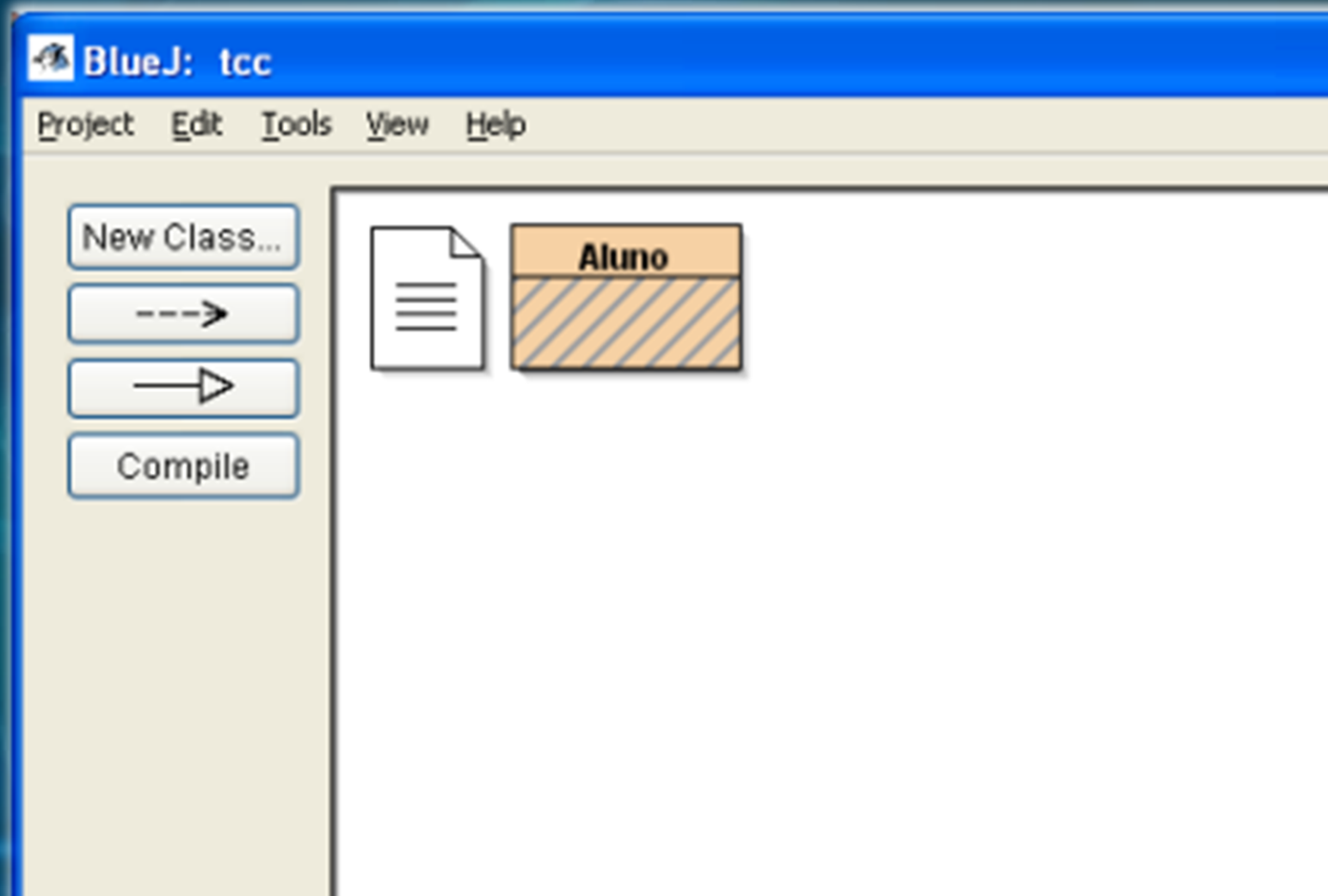


CRIANDO UM NOVO PROJETO

- Vamos criar a classe Aluno.
- Botão "New Class ...".
- (Class Name): Digite "Aluno“
- Tipo da classe (Class Type): Class
- Botão "Ok”.

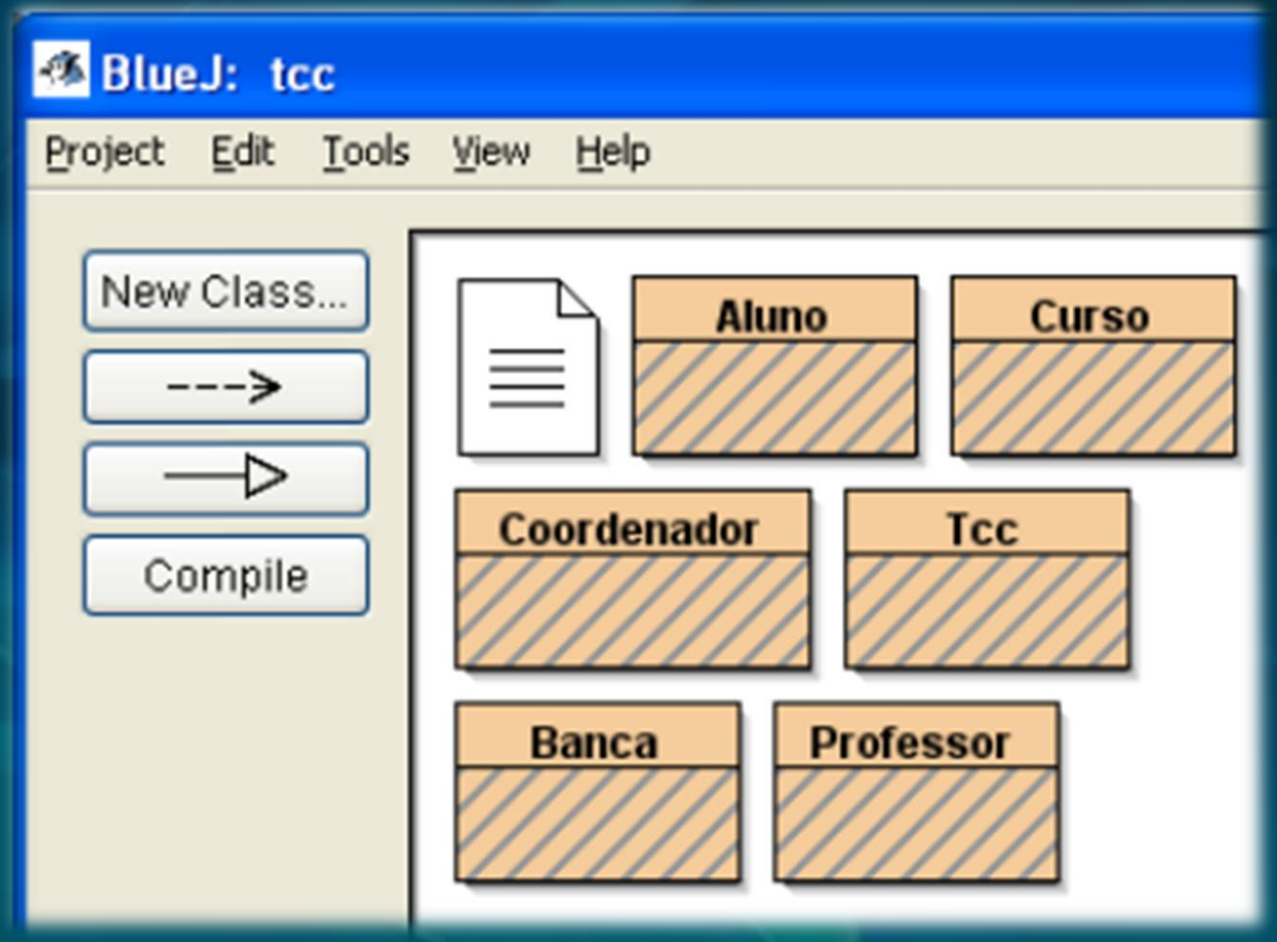


- Como resultado dessa ação, o BlueJ exibirá a classe Aluno em seu ambiente visual.



CRIANDO UM NOVO PROJETO

- Repetir os passos realizados no processo de criação da classe Aluno no BlueJ com todas as outras classes:
- Curso
- Coordenador
- Tcc
- Banca e
- Professor



ALTERAÇÕES DA CLASSE ALUNO

- Dê duplo clique com o botão esquerdo do mouse sobre a classe
- classe Aluno possui duas características que serão codificadas em Java em forma de dois atributos: nome e ra.
- Um atributo corresponde à declaração de uma variável, assim, definiremos os dois atributos como do tipo caractere usando o tipo String da linguagem Java.

```
// variáveis da instância que representam as características do Aluno
private String nome;
private String ra;
```

Atributos da classe (características)

ALTERAÇÕES DA CLASSE ALUNO

- Outra modificação importante a ser realizada no código da classe Aluno corresponde ao método construtor chamado Aluno.
- Na linguagem Java, todo método construtor deve possuir o mesmo nome da classe.
- Esse método será o responsável pela criação de objetos e terá o importante papel de realizar a iniciação dos valores dos atributos dos objetos.

```
 * Método Construtor responsável por criar objetos do tipo Aluno
 */
public Aluno(String nome, String ra)
{
    // atribui valores para as variáveis da instância (objeto)
    this.nome = nome;
    this.ra = ra;
}
```

Método Construtor (cria objetos do tipo Aluno)

ALTERAÇÕES DA CLASSE ALUNO

- Além das alterações descritas, também introduzimos no código da classe Aluno o método
- "exibeInformacoes()" que será responsável por mostrar na tela (terminal do BlueJ) as informações armazenadas pelos objetos. Foi utilizado nesse método a chamada "System.out.println",
- que corresponde aos comandos "writeln" da linguagem Pascal e "printf" da linguagem C.

```
 * Método que exibe informações sobre o Aluno
 */
public void exibeInformacoes( )
{
    System.out.println("Informações sobre Aluno:");
    System.out.println("Nome: "+nome);
    System.out.println("Ra: "+ra);
}
```

Método exibeInformacoes ( )
(permite visualizar as características dos objetos)

(Comportamentos)

ALTERAÇÕES DA CLASSE ALUNO

- O código ao lado, exibe, em negrito, as partes que devem ser alteradas no código-padrão da classe Aluno.
- Faça as mudanças indicadas, lembrando que a linguagem Java faz diferenciação entre letras minúsculas e maiúsculas (case-sensitive).
- Depois de realizar as alterações, você deverá compilar a classe clicando no botão "Compile", localizado na parte superior da janela de edição do código.

```
/**
 * Classe Aluno.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Aluno
{
    // variáveis da instância que representam as características do Aluno
    private String nome;
    private String ra;

    /**
     * Método Construtor responsável por criar objetos do tipo Aluno
     */
    public Aluno(String nome, String ra)
    {
        // atribui valores para as variáveis da instância (objeto)
        this.nome = nome;
        this.ra = ra;
    }

    /**
     * Método que exibe informações sobre o Aluno
     */
    public void exibeInformacoes( )
    {
        System.out.println("Informações sobre Aluno:");
        System.out.println("Nome: "+nome);
        System.out.println("Ra: "+ra);
    }
```

Atributos da classe (características)

Método Construtor (cria objetos do tipo Aluno)

Método exibeInformacoes ( ) (permite visualizar as características dos objetos)

(Comportamentos)

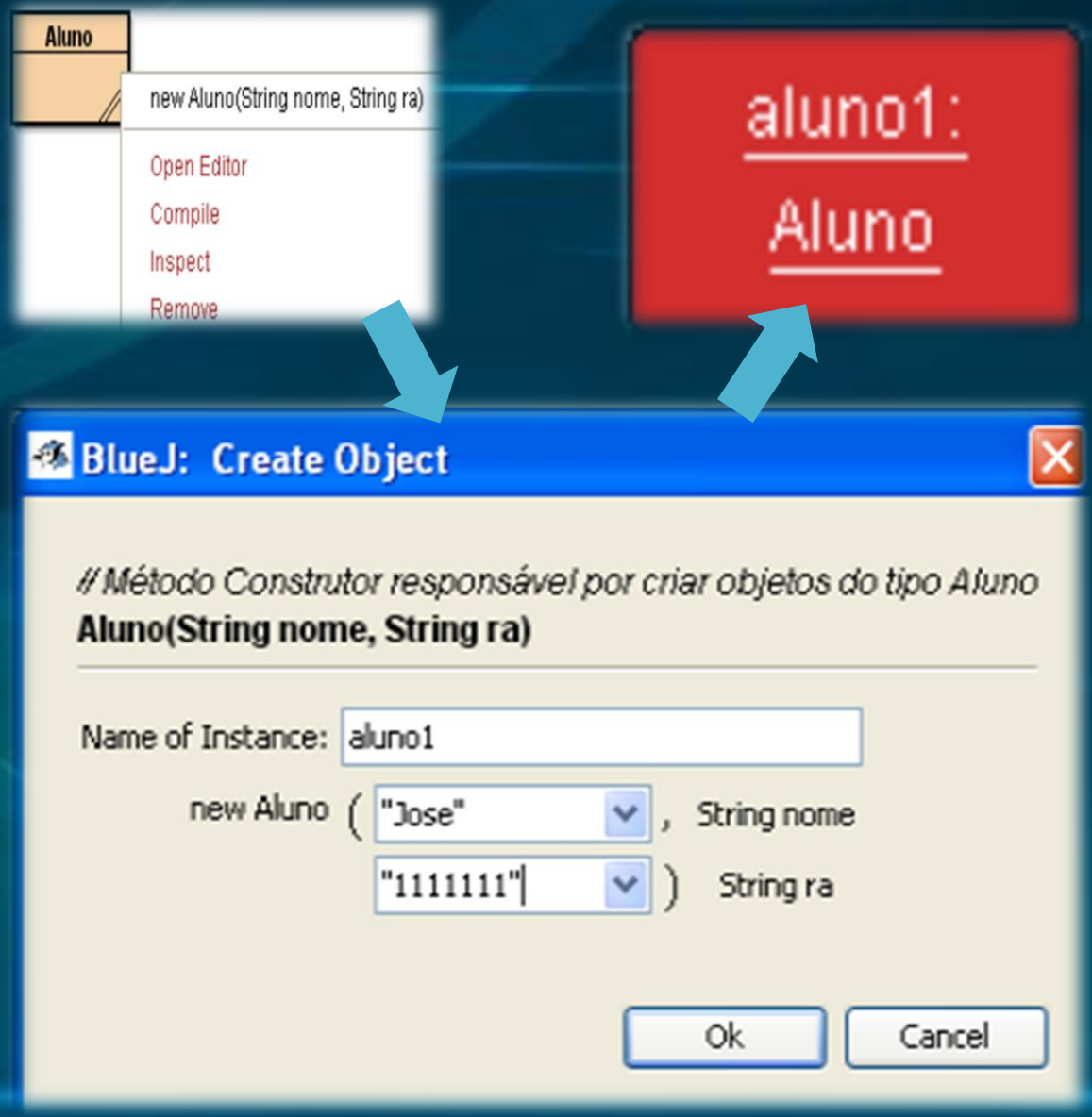
ALTERAÇÕES DA CLASSE ALUNO

- Note que removemos do código original da classe Aluno o método "sampleMethod" e toda a sua declaração. Essa remoção foi necessária, pois tal método fazia referência a variáveis que foram retiradas da declaração da classe Aluno.
- Mais à frente, realizaremos a codificação de outros métodos que tratarão dos comportamentos das classes.
- Uma vez que a classe esteja compilada, vamos testá-la criando objetos por meio da interface da ferramenta BlueJ.
- Para isso, clique com o botão direito do mouse sobre a representação visual da classe Aluno e escolha a opção ***"New Aluno (String nome, String ra)"*** (Novo Aluno).

Continua no próximo slide...

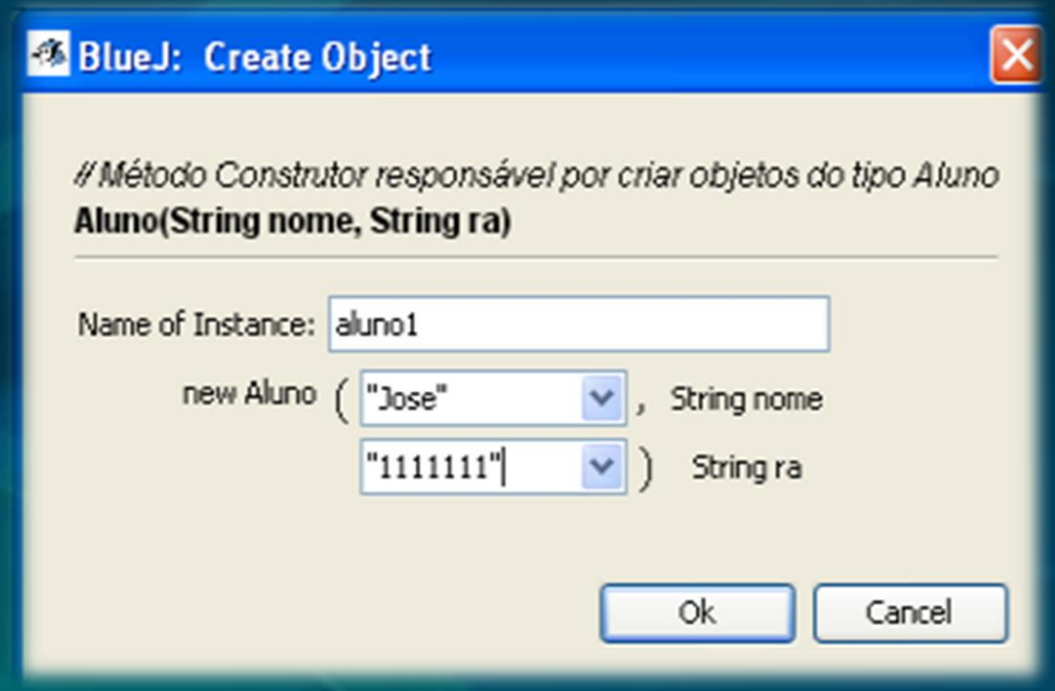
ALTERAÇÕES DA CLASSE ALUNO

- Essa operação abrirá uma nova janela, que solicitará a informação de *"Name of Instance"* (Nome da Instância), que corresponde à identificação (nome) do novo objeto Aluno que será criado.
- Cada objeto instanciado com base em uma classe deve receber um nome para identificação única.



ALTERAÇÕES DA CLASSE ALUNO

- mantivemos o nome da instância (ou nome do objeto) sugerido pelo BlueJ e adicionamos as informações do nome ("Jose") e ra ("1111111") do novo aluno (novo objeto).
- Você pode digitar qualquer outra informação, só não se esqueça de que como nome e ra foram definidos na linguagem Java como do tipo *String*, então as informações digitadas devem ser iniciadas e finalizadas com aspas duplas.
- Clique sobre o botão "Ok" para confirmar a criação do novo objeto.
- Para cada objeto criado (ou instanciado) no BlueJ será exibida, na parte inferior da ferramenta, uma representação visual dos objetos instanciados em memória (Figura 30).



crie mais objetos do tipo Aluno para praticar!

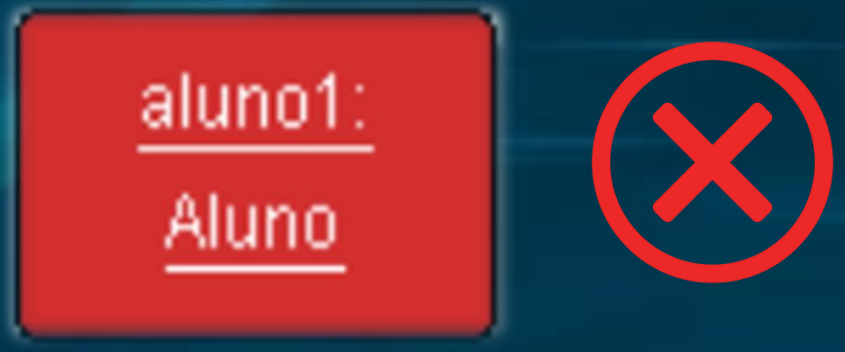
ALTERAÇÕES DA CLASSE ALUNO

- Uma vez que um objeto esteja instanciado em memória, podemos realizar a invocação de métodos que estão disponíveis para ele.
- Todos os métodos são codificados dentro da classe que define o objeto, no nosso caso, dentro da classe Aluno.
- Assim, para visualizar os métodos disponíveis para um objeto no BlueJ, você deverá clicar com o botão direito do mouse sobre o objeto (Figura).
- Isso acontece porque apenas este método foi
- codificado e encontra-se disponível dentro da classe Aluno (ver código da classe)

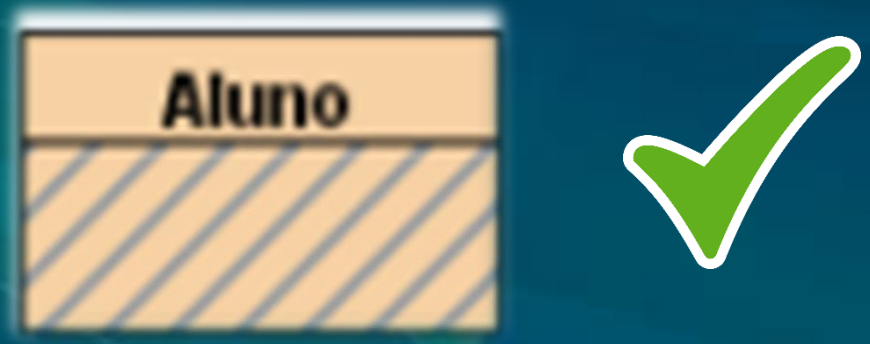
Figura 32 *BlueJ: menu sobre objeto do tipo Aluno.*

ALTERAÇÕES DA CLASSE ALUNO

- Mas onde está o método Construtor da classe Aluno? Ele não deveria aparecer também para o objeto “aluno1” junto com o método "exibeInformacoes()"?

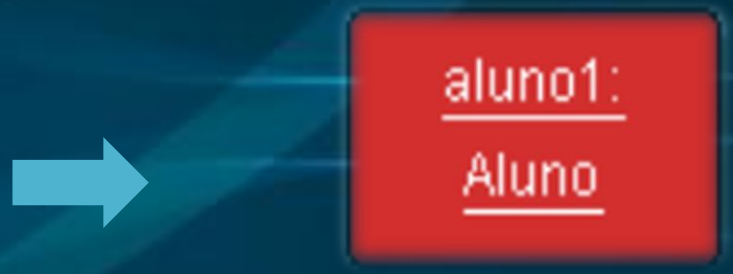


- A resposta para esse questionamento é: NÃO, pois o ***método construtor só poderá ser executado sobre a classe, e nunca sobre os objetos instanciados***.



ALTERAÇÕES DA CLASSE ALUNO

- Antes de invocarmos métodos sobre os objetos, executaremos a opção "Inspect" (Inspecionar) para visualizar o estado atual do Objeto "aluno1" (Figura).

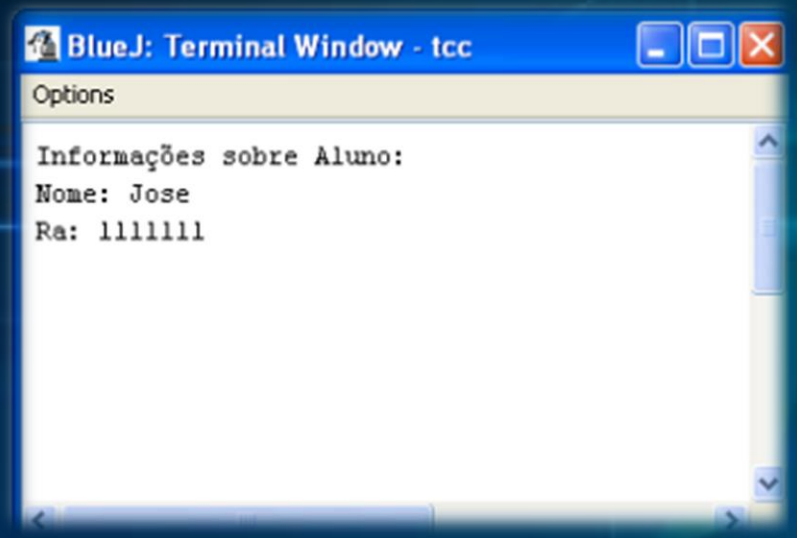


- Como resultado dessa ação, todas as informações referentes aos atributos (características) do objeto serão exibidas (Figura 32).

- Faça esse procedimento em todos os outros objetos que você tenha instanciado e note como os objetos possuem características diferentes. Relembramos que cada objeto possui um armazenamento distinto em memória para cada objeto instanciado.

ALTERAÇÕES DA CLASSE ALUNO

- Vamos invocar o método "exibeInformacoes()" sobre o objeto "aluno1" (Figura). Como resultado dessa ação, o Java executará o método sobre o objeto que recebeu a invocação.
- Nesse exemplo, o objeto "aluno1" será a referência para a execução do método "exibeInformacoes()" e, portanto, os atributos referenciados no método serão os correspondentes ao objeto "aluno1".



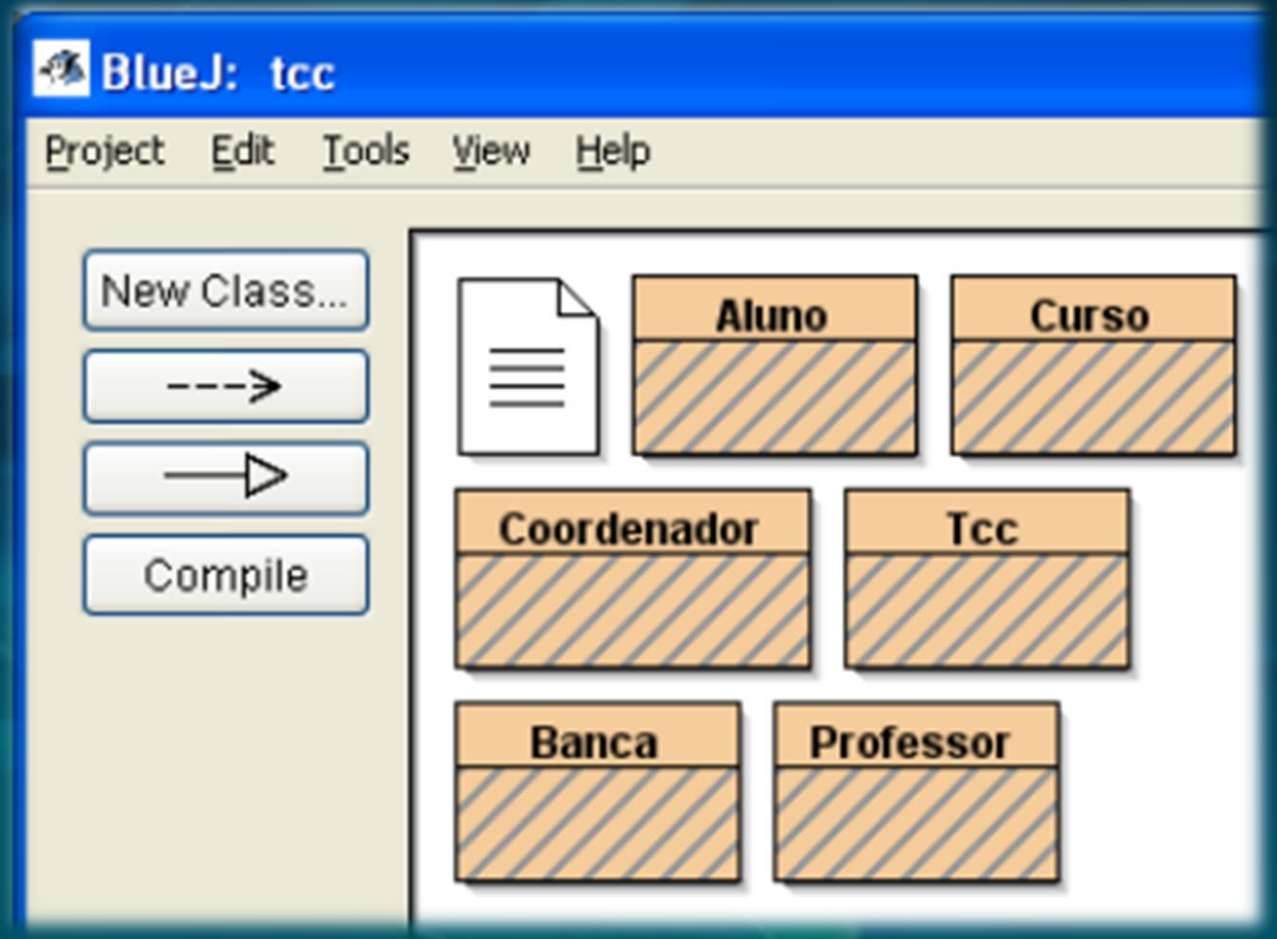
- O método "exibeInformacoes()" utiliza o comando "System.out.println" para enviar as informações de texto e os valores dos atributos para o terminal do BlueJ responsável por exibir as mensagens em formato texto (Figura 33).

ALTERAÇÕES DA CLASSE ALUNO

- É aconselhável que, após o término da utilização de um objeto, você faça a "destruição“ dele, o que resultará na consequente liberação da memória que o objeto alocava.
- Para isso, você tem à disposição, dentro da ferramenta BlueJ (Figura), a opção "Remove" (Remover) que indicará à linguagem Java a destruição do objeto.

CRIANDO UM NOVO PROJETO

- Agora realize as alterações sobre as demais classes existentes no projeto do sistema de controle de orientações e defesas de TCC:
- Curso
- Coordenador
- Tcc
- Banca e
- Professor



GLOSSÁRIO DE CONCEITOS

1) Abstração: mecanismo utilizado na análise de um domínio de aplicação em que se observa a realidade e dela se identificam informações consideradas essenciais para uma aplicação, excluindo todos os aspectos julgados irrelevantes.

2) Agregação: tipo de relação existente entre classes que permite a reutilização de classes já existentes para compor classes de estruturas mais complexas. A relação estabelecida é do tipo "parte-todo", em que se criam classes (agregadoras) que se utilizam de outras classes (agregadas) como parte de sua definição.

3) Associação: outro tipo de relação existente entre classes, em que as classes envolvidas não apresentam estruturas comuns como acontece na herança e na agregação. O que se observa normalmente é que, na associação, as classes relacionadas apresentam estruturas distintas, e o tipo de relação está vinculado a alguma regra de negócio do sistema.

4) Classe: abstração de um conjunto de objetos similares do mundo real, descrevendo a estrutura de dados e o comportamento de objetos similares.

5**) Classificação:** ação de criar classes por meio da abstração de conceitos existentes em objetos que possuem características e comportamentos iguais

GLOSSÁRIO DE CONCEITOS

6) Domínio de Aplicação: corresponde às informações do ambiente em que a aplicação está inserida. Por exemplo, para projetar um sistema de biblioteca, é necessário que o desenvolvedor compreenda todas as regras de negócio relacionadas ao funcionamento da biblioteca (controle de livros, empréstimo, devolução etc.). A compreensão do domínio da aplicação é pré-requisito para um bom projeto de software.

7) Encapsulamento: é uma técnica para minimizar interdependências entre objetos, por meio da definição de métodos que possibilitam o acesso aos dados do objeto. Assim, mudanças na definição e na implementação de uma classe, desde que preservem os métodos de acesso, não afetam outras classes presentes no restante do sistema.

8) Herança: tipo de relação existente entre classes que permite definir uma nova classe (subclasse) a partir de uma já existente (superclasse). Ao se estabelecer uma subclasse, ela herda todas as características da superclasse, ou seja, a especificação dos atributos e dos métodos da superclasse passa a fazer parte da especificação dos atributos e dos métodos da subclasse. Assim, a subclasse pode adicionar novos métodos, como também reescrever os métodos herdados da superclasse. A relação estabelecida é chamada de "é do tipo", em que uma subclasse "é do tipo" da superclasse.

9) Instanciação: corresponde à ação de criar objetos com base em uma classe. Todo objeto criado é uma instância de uma classe.

10) Objeto: abstração de uma entidade do mundo real representada por meio das características e dos comportamentos.

```
public class Executa { // Definição da classe chamada Executa

    public static void main(String arg) {// Método que inicia o programa

            System.out.println("Você passou o parâmetro :"+arg); // comando de escrita

    }//fecha o método main

}//fecha a definição da classe
```

```
public class Soma
{
 public static void main(String[] args) // método que inicia o programa
        {
        // converte os parâmetros em inteiros e os armazena em a e b
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = a + b; // realiza a soma
        System.out.println("O Resultado da soma é: " + c); // exibe a soma
        }
}
```

```
class EstacaoDoAno
{
 public static void main(int mes)
 {
 String estacao; // armazenara o nome do mes
 if (mes ==12 || mes == 1 || mes == 2)
 { // IF dos meses de Verao
 estacao = “Verão”;
 }
 else if (mes ==3 || mes == 4 || mes ==5)
 { // IF dos meses de Outono
 estacao = “Outono”;
 }
 else if (mes ==6 || mes == 7|| mes ==8)
 { // IF dos meses de Inverno
 estacao = “Inverno”;
 }
 else if (mes ==9|| mes == 10|| mes ==11)
 { // IF dos meses de Primavera
 estacao = “Primavera”;
 }
  else // ELSE para mes invalido
  estacao = “Desconhecida”;
  System.out.println(“Mês: “+ mes + “ - Estação: “ +
estacao + “.”); // exibe
estacao
  }
}
```

```
class EstacaoDoAno2
{
 public static void main(int mes)
 {
 String estacao; // armazenara o nome do mes
 switch (mes)
 {
 case 12:
 case 1:
 case 2:
 estacao = “Verão”;
 break;
 case 3:
 case 4:
 case 5:
 estacao = “Outono”;
 break;
 case 6:
 case 7:
 case 8:
 estacao = “Inverno”;
 break;
 case 9:

 case 10:

 case 11:
 estacao = “Primavera”;
 break;

 default:
 estacao = “Desconhecida”;
 }
 // exibe a estação conforme o mes
 System.out.println(“Mês: “+ mes + “ - Estação: “ +
estacao + “.”);
 }
}
```

```
class RepeticaoFor
{
 public static void main(String arg)
 {
 for (int i = 1; i <= 10; i++) // repete 10 vezes
 System.out.println("i = " + i + " / Parâmetro: "+ arg); }
}
```

```
class RepeticaoWhile
{
 public static void main(String arg)
 {
 int i = 1;// inicialização de i
 while (i <= 10)
 { // encerramento em 10
 System.out.println("i = " + i + " / Parâmetro: "+ arg); // imprime o
valor da variavel i
 i++; // iteracao para adicionar 1 em i
 }
 }
}
```