Unidade Acadêmica: Faculdade de Computação – FACOM
Disciplina: Programação Orientada a Objetos I
Professor: Fabiano Azevedo Dorça

Prática 01

Objetivos: Modelar um sistema orientado a objetos simples a partir de uma situação problema (domínio do problema). Exercitar a extração e reconhecimento de objetos existentes no mundo real, seus atributos e seus métodos, propondo uma solução para o problema. Iniciar o uso do ambiente BlueJ para implementar a solução.

1) Situação problema:
   1. Domínio do problema: Gestão acadêmica
   2. Descrição do problema: Uma universidade necessita de um sistema que facilite a sua gestão acadêmica. Deseja-se um controle de quais disciplinas são ministradas por cada professor e em qual curso. Também é necessário um controle de quais disciplinas são cursadas por um aluno. Sabe-se que um professor é um funcionário. Além de professores, tem-se funcionários que são técnicos administrativos. Para cada funcionário, independente de ser professor ou técnico, é necessário saber seu nome, endereço, telefone, cpf, número da CTPS e salario. Especificamente para professores, é necessário saber sua titulação e sua área de pesquisa. Para cada curso é necessário registrar seu código, nome e duração. Para cada disciplina é necessário registrar seu código, nome e carga horária.

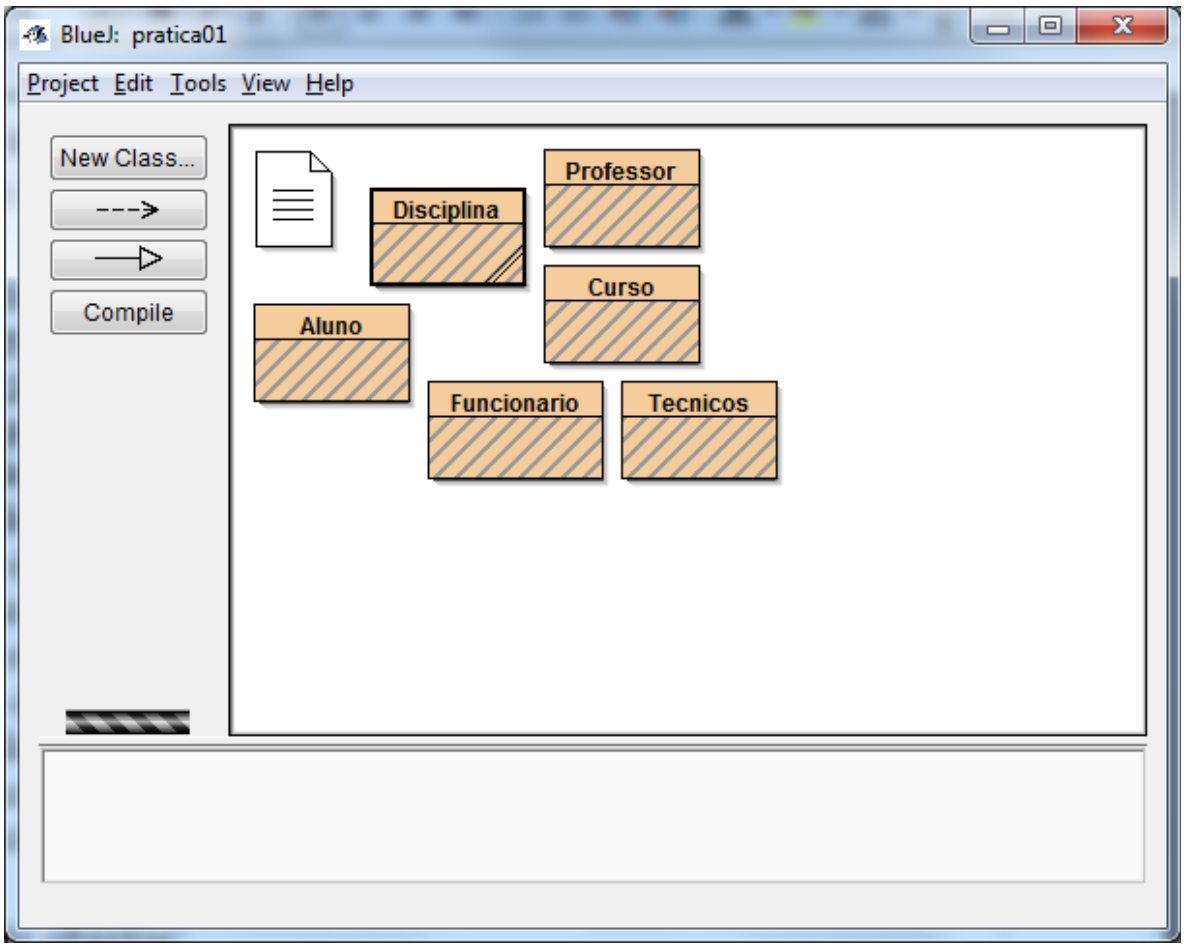
2) Identificação de objetos:
   1. Definição: Objetos são essencialmente componentes de software reutilizáveis que modelam itens do mundo real. Ou seja, um objeto é uma entidade representativa que pode caracterizar algo concreto ou abstrato do mundo real. Atributos são informações que caracterizam um objeto.
   2. Extraindo objetos a partir da descrição do problema: Vamos marcar de vermelho os objetos do problema e em azul os atributos:
      1. Uma universidade necessita de um sistema que facilite a sua gestão acadêmica. Deseja-se um controle de quais **disciplinas** são ministradas por cada **professor** e em qual **curso**. Também é necessário um controle de quais disciplinas são cursadas por um **aluno**. Sabe-se que um professor é um **funcionário**. Além de professores, tem-se funcionários que são **técnicos administrativos**. Para cada funcionário, independente de ser professor ou técnico, é necessário saber seu **nome**, **endereço**, **telefone**, **cpf, número da CTPS** e **salario**. Especificamente para

professores, é necessário saber sua **titulação** e sua **área de pesquisa**. Especificamente para **técnicos**, é necessário saber seu **cargo** e **departamento**. Para cada **curso** é necessário registrar seu **código**, **nome** e **duração**. Para cada **disciplina** é necessário registrar seu **código**, **nome** e **carga horária**.
Para cada **aluno**, deve-se armazenar seu **nome, matrícula, cpf e curso**.

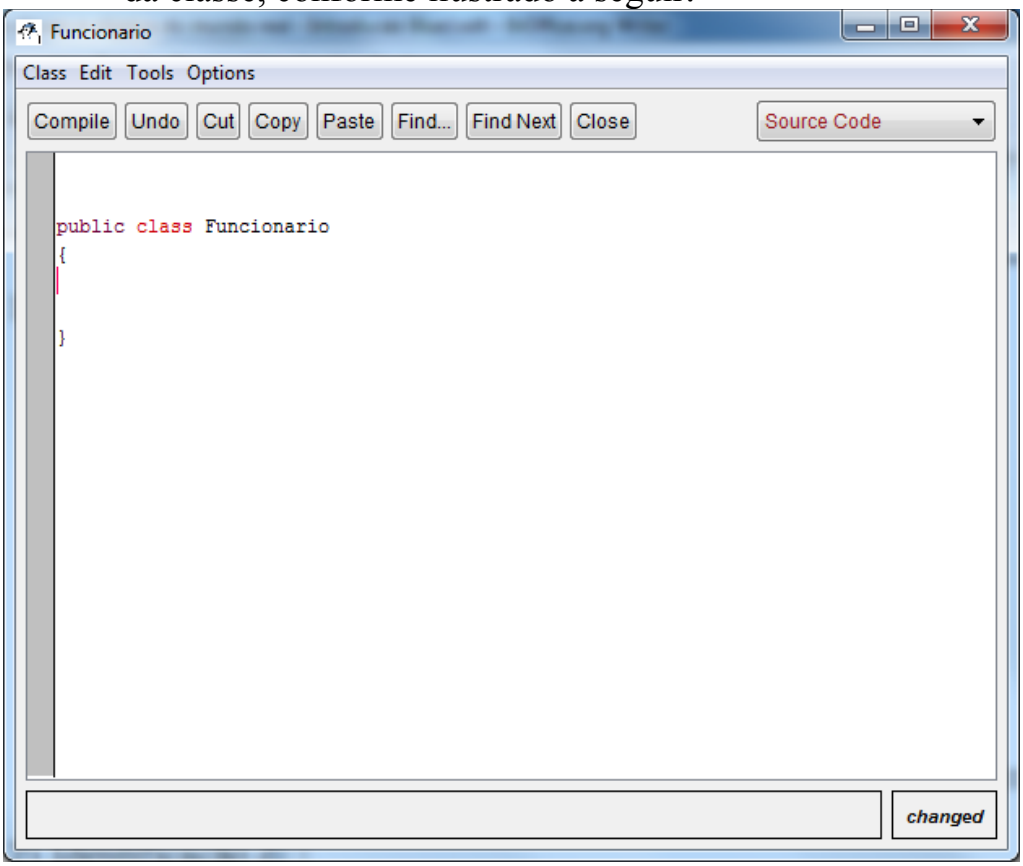
3) Criação de classes de objetos no BlueJ:

1. Definição: Uma classe é uma matriz (modelo ou forma) a partir da qual os objetos são criados (instanciados). Cada objeto tem a mesma estrutura e comportamento da classe a partir da qual ele foi instanciado.

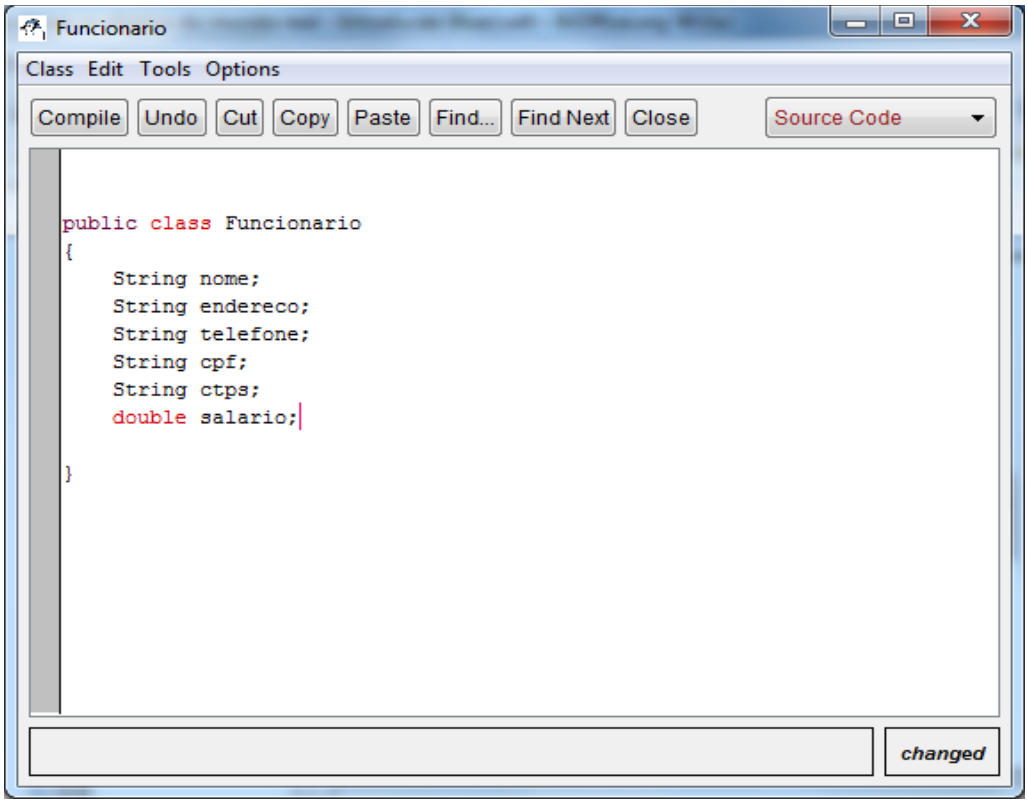
2. Implementação:
   1. Abra o BlueJ
   2. Acesse o menu Project → New Project
   3. Selecione o diretório onde deseja criar o projeto e nomeie-o como "Pratica01"
   4. Clique em "New Class" e nomeie a nova classe como "Disciplina". Faça o mesmo para as demais classes do sistema (marcadas em vermelho no item anterior) resultando em:



Vamos criar os atributos das classes. Para isto é necessário editar o código fonte das classes. De um clique duplo na classe “Funcionario”. Apague os comentários e demais códigos, deixando apenas a declaração da classe, conforme ilustrado a seguir:



5. Crie os atributos da classe da seguinte forma:



Note que para declarar um atributo para a classe, especificamos o tipo do atributo seguido por seu nome.

Note que foi utilizado o tipo String. Além do tipo String o Java oferece vários outros tipos de dados:

byte: 8 bits (-128 até 127)
short: 16 bits (-32.768 até 32.767)
int: 32 bits (-2.147.483.648 até 2.147.483.647)
long e double: 64 bits (-9223372036854775808 até 9223372036854775807)
boolean: false, true
char: qualquer caracter.
Dentre outros...

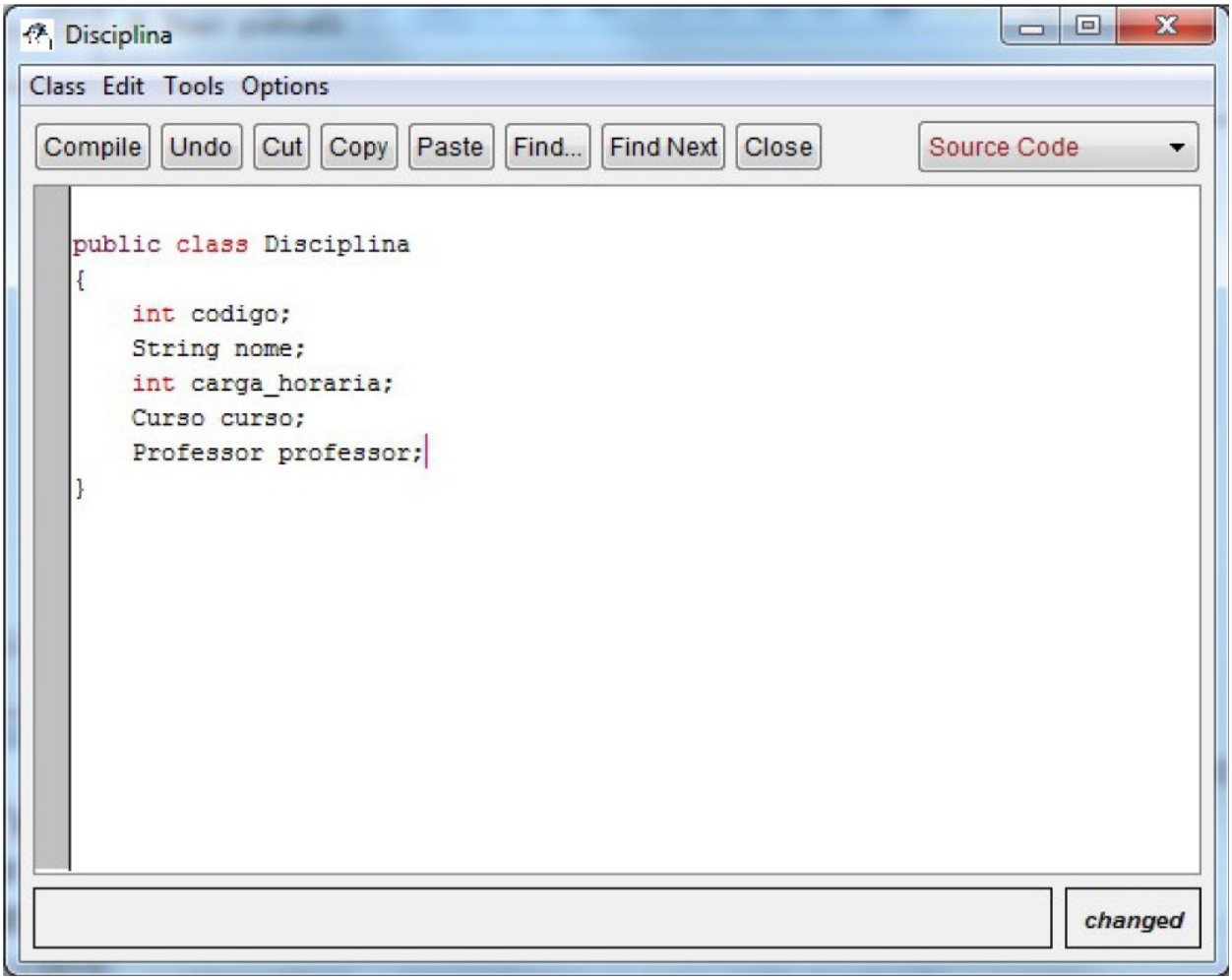
Declare todas as demais classes de objetos encontradas no domínio problema. Agora, compile o sistema clicando em "Compile" na barra de ferramentas lateral. Caso haja erros de sintaxe, eles serão apresentados.

Note que após a compilação, um arquivo .class é gerado para cada arquivo .java no diretório do projeto. Os arquivos .class são as classes compiladas, em bytecodes, passíveis de serem executados pela JVM.

Uma característica importante do problema descrito é a necessidade de se registrar qual professor é responsável por uma disciplina e em qual curso ela é ministrada.

Ou seja, é importante relacionar um objeto disciplina com o professor que a ministra, e com o curso a qual pertence.

Para isto, podemos modificar a classe "Disciplina" para conter estes atributos, resultando na seguinte declaração:



Observe que foram inseridos dois atributos na classe “Disciplina” além dos atributos já mencionados pela descrição do problema:

- um atributo curso do tipo Curso e
- um atributo professor do tipo Professor.

Note que Curso e Professor são classes que criamos em nosso projeto e que podem ser usados como tipos na declaração de atributos. Na programação orientada a objetos, toda classe que criamos é um tipo de dado. Em nosso exemplo, agora estamos aptos a armazenar um curso e um professor em cada disciplina que criarmos.

Além disto, é importante que se relacione cada aluno com o curso o qual está matriculado. Para isto, criamos um relacionamento entre as classes Aluno e Curso, da mesma forma que foi feito com Disciplina e Professor (no item anterior): colocamos um atributo curso do tipo Curso na classe Aluno. Por exemplo:

```
public class Aluno{
     String nome;
     String matricula;
     String cpf;
     Curso curso;
}
```

Agora vamos criar um método na classe “Funcionario” que permita ler a partir do teclado (dispositivo de entrada padrão) os valores dos atributos do funcionário. Para isto, implemente o código apresentado a seguir na classe “Funcionario”, logo após a declaração de seus atributos:

```
public void lerDados() {
   Scanner s = new Scanner(System.in);

   System.out.println("Digite o nome do funcionário:");
    this.nome = s.nextLine();

   System.out.println("Digite o endereço do funcionário:");
    this.endereco = s.nextLine();

   System.out.println("Digite o telefone do funcionário:");
    this.telefone = s.nextLine();

   System.out.println("Digite o CPF do funcionário:");
    this.cpf = s.nextLine();

    System.out.println("Digite a CTPS do funcionário:");
    this.ctps = s.nextLine();

    System.out.println("Digite o salário do funcionário:");
    this.salario = s.nextDouble();
}
```

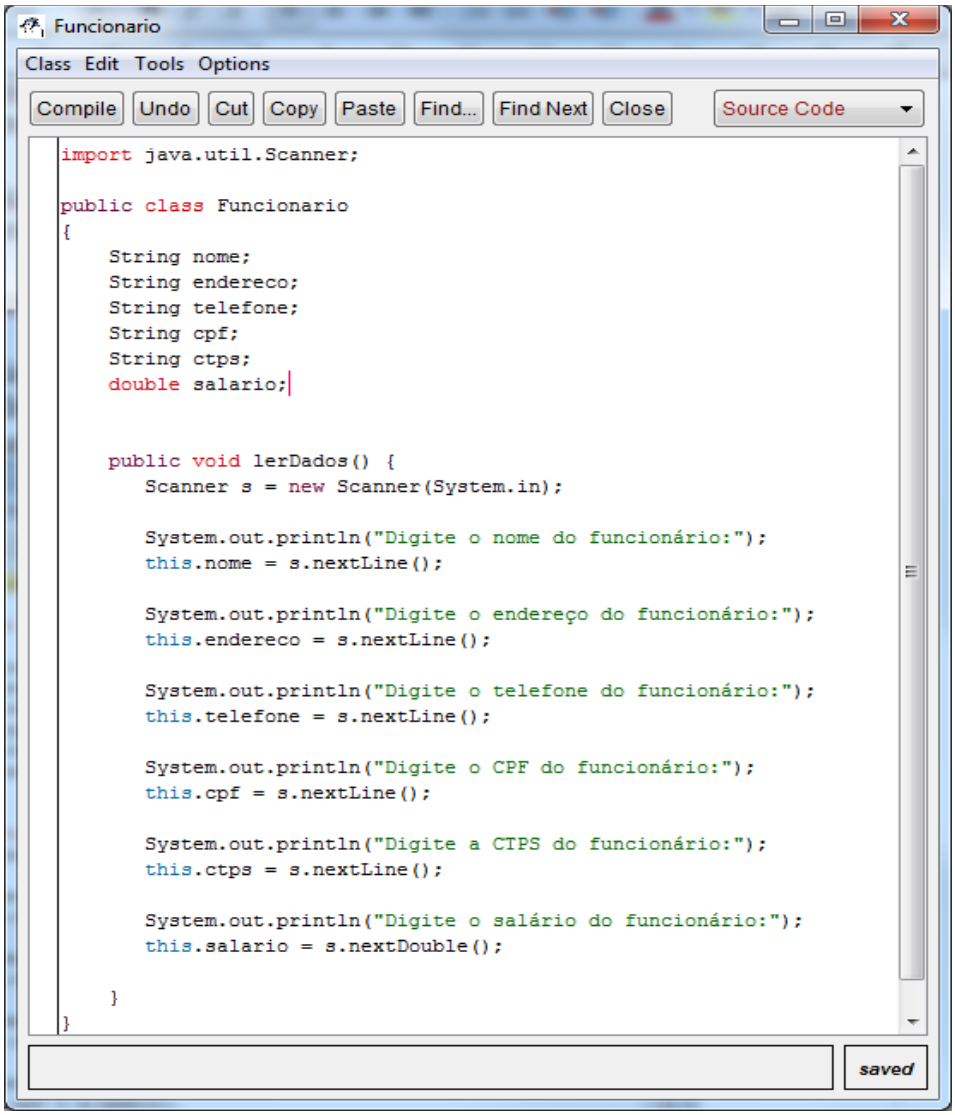
Note que este método utiliza a classe Scanner para implementar a leitura de dados. Para isto, instanciamos um objeto **s** do tipo Scanner e utilizamos o método nextLine() para ler uma string, nextDouble para ler um double. Ainda, podemos usar o método nextInt() para ler inteiros, nextFloat() para ler floats, e assim por diante.

O operador “this” serve para referenciar atributos da classe, e resolver conflitos de nomes, caso haja variáveis internas do método com o mesmo nome que o atributo da classe.

Para utilizar a classe Scanner, é necessário importa-la. Para isto, insira a seguinte linha ANTES da declaração da classe:

import java.util.Scanner;

Veja que para referenciar os atributos da própria classe, usamos a palavra reservada “this” para evitar conflitos com nomes de variáveis e parâmetros do método com o mesmo nome do atributo. O resultado é o seguinte:



Agora implemente o método lerDados() para as demais classes do sistema.

Ao se instanciar e ler os dados de um objeto que se relaciona com outro objeto, por exemplo, aluno e curso, uma instância de Curso deve ser atribuída ao atributo curso da classe Aluno. Uma solução plausível é que tivéssemos uma lista de objetos do tipo curso onde poder-se-ia buscar um curso através de seu

código e atribuir o objeto encontrado ao atributo curso de Aluno. Mas isto será feito posteriormente.

Por exemplo:
class Aluno {

….atributos

```
public void lerDados() {
   Scanner s = new Scanner(System.in);

   System.out.println("Digite o nome do aluno:");
    this.nome = s.nextLine();

   System.out.println("Digite a matricula do aluno:");
    this.matricula = s.nextLine();

   System.out.println("CPF do aluno:");
    this.CPF = s.nextLine();

   System.out.println("Digite o curso do aluno:");

   this.curso = new Curso();

   this.curso.lerDados();

}
```

Implemente também um método mostrarDados() em cada classe. O código a seguir ilustra a implementação do método mostrarDados() para a classe funcionário. Teste os métodos implementados.

```
public void mostrarDados() {
   System.out.println("Nome: "+this.nome);
   System.out.println("Endereço: "+this.endereco);
   System.out.println("Telefone: "+this.telefone);
   System.out.println("CPF: "+this.cpf);
   System.out.println("CPF: "+this.ctps);
   System.out.println("CPF: "+this.salario);
}
```

Agora crie uma classe principal com um método main para testar suas classes.
Exemplo:

class Principal {

```
    public static void main(String args[]){
            Aluno a = new Aluno();
            a.lerDados();
            a.mostrarDados();
            ….teste as demais classes....
    }
}
```

IMPORTANTE::

A linha de código:

```
Aluno a = new Aluno();
```

instancia um novo objeto da classe Aluno (new) e atribui sua referência à variável "a" do tipo Aluno. Desta forma, este objeto passa a ser acessível a partir da variável "a", e seus atributos e métodos podem ser acessados através de um "."

Exemplos:

a.lerDados(); -- invoca o método lerDados() do objeto referenciado pela variável "a";

a.nome = "João"; -- acessa o atributo "nome" do objeto referenciado por "a" e atribui a string.

Observe que neste caso, Aluno é tanto uma classe quanto um tipo;
Na verdade, toda classe é um tipo.

Copie o diretório do seu projeto para um local seguro e tenha-o sempre à mão. Este projeto será utilizado nas próximas aulas práticas.

Nunca se esqueça: O que é um paradigma?

É uma forma de abordar um problema, segundo um conjunto de procedimentos, valores ou conceitos que direcionam o pensamento.
