

INSTITUTO MAUÁ DE TECNOLOGIA



Linguagens I

Introdução a POO
Com Java

Prof. Tiago Sanches da Silva
Prof. Murilo Zanini de Carvalho

Configure seu ambiente

Configuração do ambiente de trabalho

Crie um repositório no seu GitHub: “Linguagens1_Projetos”;

Configure o Gitbash (user.name/user.email);

Entre na sua pasta local no computador e clone o repositório criado;

Dentro do repositório **local** crie uma nova pasta chamada “pratica1”;
Essa será a pasta de trabalho para esse primeiro dia de pratica;

Dentro da pasta “pratica1” serão criadas novas pastas segundo necessidade, uma pra cada exercício. **Por exemplo**, se no dia de hoje tiverem 3 exercícios diferentes, dentro da pasta “pratica1” então deve conter 3 pastas nomeadas da seguinte forma: “exercicio1”, “exercicio2” e “exercicio3”.

Nova semana, nova pasta pratica!

Configuração do ambiente de trabalho



RA_Nome

[Pasta do aluno no computador local]

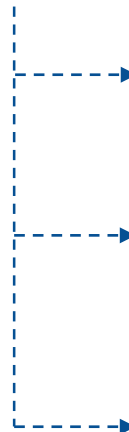


[Repositório local (clonado) (.git)]

Linguagens1_Projetos



pratica1



exercicio1



exercicio2



exercicio3

Modelar uma classe para conta bancaria

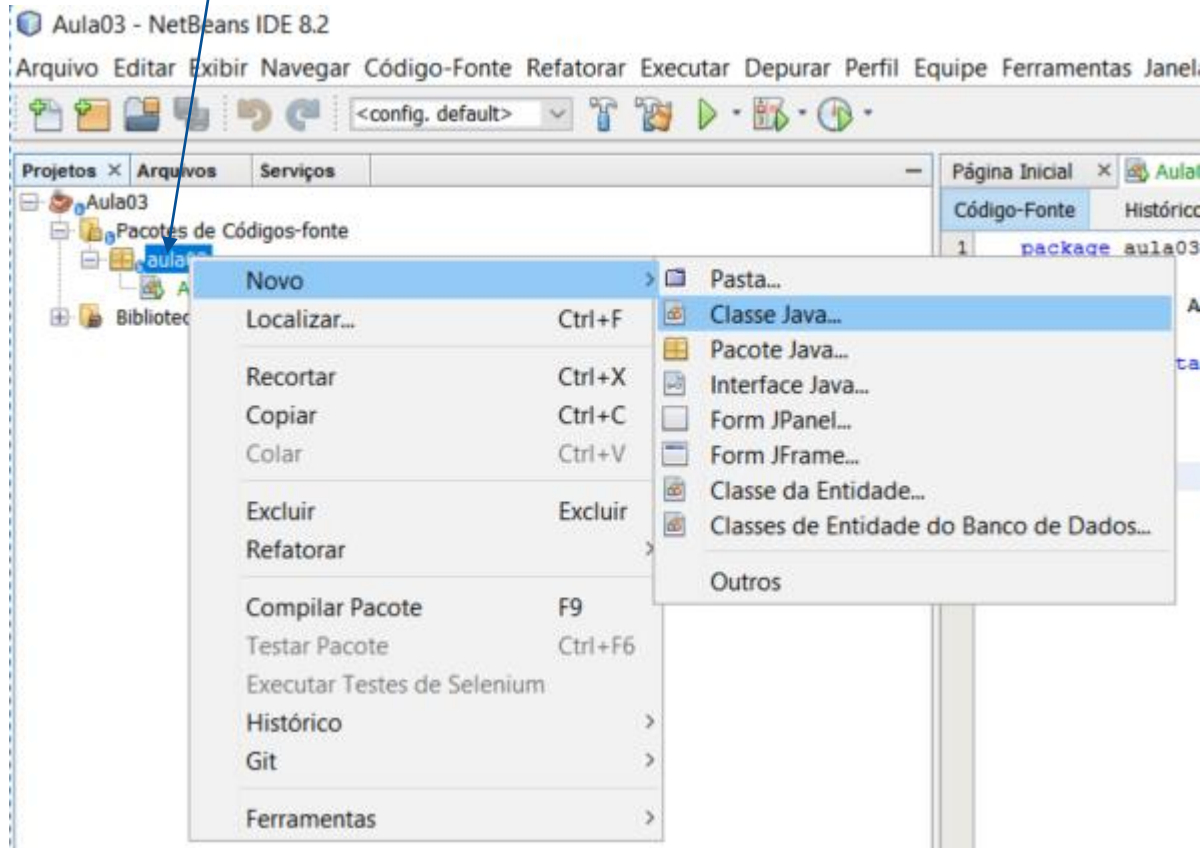
Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

Atributos dessa classe


Métodos dessa classe

Ex. Dirigido – Adicionando uma nova classe

Clique com direito em cima do pacote



Ex. Dirigido – Adicionando uma nova classe

 New Classe Java ✕

Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**

Nome e Localização

Nome da Classe:

Projeto:

Localização:

Pacote:

Arquivo Criado:

Ex. Dirigido

Aula03 - NetBeans IDE 8.2

Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda

The screenshot shows the NetBeans IDE 8.2 interface. On the left, the 'Projetos' (Projects) pane displays the project structure: 'Aula03' contains 'Pacotes de Códigos-fonte' (Source Packages), which includes 'aula03'. Inside 'aula03', there are two source files: 'Aula03.java' and 'Conta.java'. A blue arrow points to 'Conta.java'. The main editor window shows the 'Código-Fonte' (Source) view of 'Conta.java'. The code is as follows:

```
1  /*  
2   * To change this license header, choose License Headers in Project Properties.  
3   * To change this template file, choose Tools | Templates  
4   * and open the template in the editor.  
5   */  
6  package aula03;  
7  
8  /**  
9   *  
10   * @author Note-Tiago  
11   */  
12  public class Conta {  
13  
14  }
```

A red arrow points to the 'public class Conta {' line in the code editor.

Conta

numero: inteiro

titular: string

saldo: real

cpf: string

visualizarSaldo()

depositar()

sacar()

transferirDinheiro()

Ex. Dirigido

```
Página Inicial x Aula03.java x Conta.java x
Código-Fonte Histórico
1 package aula03;
2
3 public class Conta {
4     int numero;
5     String titular;
6     float saldo;
7     String cpf;
8
9     void visualizarSaldo() {
10
11     }
12
13     void depositar() {
14
15     }
16
17     void sacar() {
18
19     }
20
21     void transferirDinheiro() {
22
23     }
24
25
26 }
27
```

Atributos dessa classe

Métodos dessa classe

Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

Ex. Criar o objeto (Instanciar o objeto)

Para criar (construir, instanciar) uma Conta, basta usar a palavra chave new. Devemos utilizar também os parênteses, veremos mais pra frente o porque.

```
package aula03;  
  
public class Aula03 {  
    public static void main(String[] args) {  
        new Conta();  
    }  
}
```



Comando para criar o objeto na memória

Nome da classe com “()”
!Veremos o que é posteriormente!

Bem, o código acima cria um objeto do tipo Conta, mas como acessar esse objeto que foi criado? Precisamos ter alguma forma de nos referenciarmos a esse objeto. Precisamos de uma variável:

```
package aula03;  
  
public class Aula03 {  
    public static void main(String[] args) {  
        Conta c1;  
        c1 = new Conta();  
    }  
}
```



Referência



Ex. Criar o objeto (Instanciar o objeto)

```
1 package aula03;
2
3 public class Aula03 {
4
5     public static void main(String[] args) {
6
7         Conta c1 = new Conta();
8         c1.saldo = 1000;
9         c1.visualizarSaldo();
10
11     }
12 }
13
```

Declarou o objeto
como sendo da classe
Conta

Comando para criar o
objeto na memória

Nome da classe com ()
!Veremos o que é
posteriormente!

Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

Ex. Criar o objeto (Instanciar o objeto)

```
package aula03;

public class Conta {
    int numero;
    String titular;
    float saldo;
    String cpf;

    void visualizarSaldo() {
        System.out.println("Saldo= " + this.saldo);
    }

    void depositar() {

    }

    void sacar() {

    }

    void transferirDinheiro() {

    }
}
```

Auto-referencia

this = próprio objeto
que esta utilizando o
método

Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

E da pra adicionar outro objeto da mesma classe,
na nossa aplicação?

- Sim ou com certeza?

Ex. Dirigido

```
package aula03;
```

```
public class Aula03 {
```

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

```
        Conta c1 = new Conta();
```

```
        Conta minhaConta = new Conta();
```

```
        c1.saldo = 1000;
```

```
        c1.visualizarSaldo();
```

```
        minhaConta.saldo = 1800;
```

```
        minhaConta.visualizarSaldo();
```

```
    }
```

```
}
```

```
public class Conta {
```

```
    int numero;
```

```
    String titular;
```

```
    double saldo;
```

```
    String cpf;
```

```
    void visualizarSaldo() {
```

```
        System.out.println("Saldo= " + this.saldo);
```

```
    }
```

this = próprio objeto
que esta utilizando o
método

Nesse caso aqui, quem vai ser o
“this” referenciado lá na classe?

Vamos fazer **juntos** os métodos: sacar e depositar

Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

Métodos com retorno

Um método pode retornar um valor para o código que o chamou. No caso do nosso método **sacar**, podemos devolver um valor booleano indicando se a operação foi bem sucedida.

```
boolean sacar(double valor) {  
    if (this.saldo < valor) {  
        return false;  
    }  
    else {  
        this.saldo = this.saldo - valor;  
        return true;  
    }  
}
```


Façam sozinhos o método: transferirPara

Podemos ir conversando a respeito! 😊

Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

Atributos – Valores default

As variáveis do tipo atributo, diferentemente das variáveis temporárias (declaradas dentro de um método), recebem um valor padrão. No caso numérico, valem 0, no caso de boolean, valem false.

Você também pode dar valores **default**, como segue:

```
class Conta {  
    int numero = 1234;  
    String titular = "Ninguém";  
    double saldo = 0;  
}
```

Classes

Imagine que começemos a aumentar nossa classe **Conta** e adicionar nome, sobrenome e cpf do titular da conta.

Começaríamos a ter muitos atributos... e, se você pensar direito, uma **Conta** não tem nome, nem sobrenome nem cpf, quem tem esses atributos é um **Cliente**. Sugestão?

Podemos criar uma nova classe e fazer uma composição!

```
class Cliente {  
    String nome;  
    String sobrenome;  
    String cpf;  
}
```

Classes

```
class Cliente {  
    String nome;  
    String sobrenome;  
    String cpf;  
}  
  
class Conta {  
    int numero;  
    double saldo;  
    Cliente titular;  
    // ..  
}
```

Como utilizar isso? Lousa.

NullPointerException

Exibir no programa principal as informações formatadas sobre a conta.

É possível criar um método que retorne todas as informações sobre a conta de uma maneira formatada?

Apenas referência

Construa duas contas com o new e compare-os com o ==. E se eles tiverem os mesmos atributos?

1

```
Conta c1 = new Conta();  
c1.titular = "Danilo";  
c1.saldo = 100;
```

```
Conta c2 = new Conta();  
c2.titular = "Danilo";  
c2.saldo = 100;
```

```
if (c1 == c2) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferentes");  
}
```

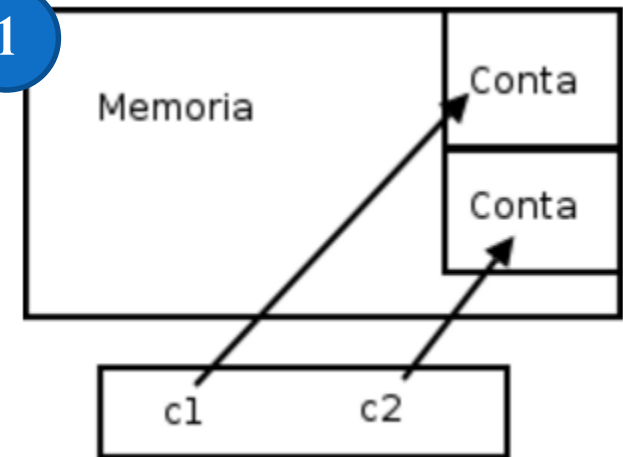
Apenas referência

Crie duas referências para a mesma conta, compare-os com o `==`. Tire suas conclusões. Para criar duas referências pra mesma conta:

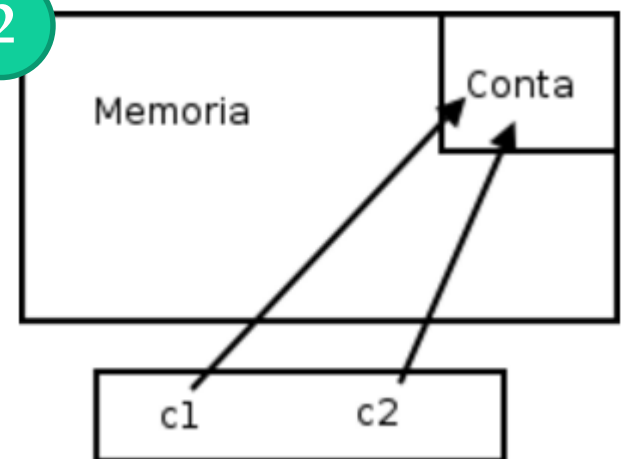
2

```
Conta c1 = new Conta();  
c1.titular = "Hugo";  
c1.saldo = 100;  
  
Conta c2 = c1;  
  
if (c1 == c2) {  
    System.out.println("iguais");  
} else {  
    System.out.println("diferentes");  
}
```

1



2



Exercício 4

- **Modelar** a classe Funcionarios de uma concessionaria.
- Crie um novo projeto: Concessionaria
- Inicie a implementação
- Discuta a solução com o professor se necessário
- RH deve conseguir acessar essas informações
 - (13º, comissão, férias, salario do mês)
- Comissão = 5% de todas as vendas
- salario do mês = salario base + horas extras + comissão

Funcionarios

Perguntas?

Exercícios!

Exercícios

1. Modelar e implementar uma classe que represente as possibilidades de jogo de “Pedra-Papel-Tesoura”. Em caso de dúvidas, consultar ([https://pt.wikipedia.org/wiki/Pedra, papel e tesoura](https://pt.wikipedia.org/wiki/Pedra,_papel_e_tesoura)).
2. Modelar e implementar uma classe que represente um jogador com seu nome e jogada de pedra-papel-tesoura.
3. Modelar e implementar uma classe Jogo, que permite que 2 jogadores possam disputar o jogo de pedra-papel-tesoura.
4. Modele e implemente uma classe que represente as jogadas de “Rock, Paper, Scissor, Lizard and Spoke” ([https://bigbangtheory.fandom.com/wiki/Rock, Paper, Scissors, Lizard, Spock](https://bigbangtheory.fandom.com/wiki/Rock,_Paper,_Scissors,_Lizard,_Spock)).
5. Modele e implemente uma classe jogo para o item 4.
6. Crie uma classe principal que permite que o jogador escolha qual jogo ele deseja jogar.