

INSTITUTO MAUÁ DE TECNOLOGIA



Linguagens I

Introdução a POO

Profº. Tiago Sanches da Silva

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]

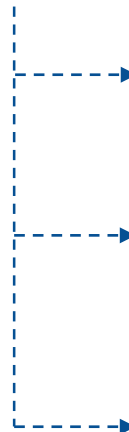


[Repositório local (clonado) (.git)]

Linguagens1_Projetos



pratica1



exercicio1



exercicio2



exercicio3

Exercício 1

Abra um arquivo e responda as seguintes perguntas:

1. Em qual pasta o jdk do java está instalado? (não incluir a pasta \bin nesta etapa)
2. E qual o caminho completo até os executáveis javac e java, que foram instalados do computador?
3. Como compilamos um programa por linha de comando utilizando o javac?
4. E como executamos este programa?
5. O que são variáveis do ambiente? Para que elas servem?
6. Qualquer um pode criar uma variável de ambiente?
7. Como eu crio um variável de ambiente no Windows/Linux (escolha apenas 1)?
8. O que é JAVA_HOME? Por que preciso dele?

Conan Mode



Exercício 2

Abra o bloco de notas e escreva seu primeiro código em Java.

PS1: “Sim no bloco de notas.”

```
class MeuPrograma {  
    public static void main(String[] args) {  
        // Mensagem de saida do sistema  
        System.out.println("Minha primeira aplicação Java!!");  
    }  
}
```

PS2: É uma imagem para você não copiar!

Utilizando o console, tente compilar o código.

1. Responda em um arquivo quais arquivos foram gerado(s)?
2. Como podemos executa-lo(s)?

Problemas nesse passo? Chame o professor!

Exercício 3

Modificar o exercício anterior para que exiba como saída as linhas:

“Olá mundo!”

“Estou programando no modo Conan. :)”

Compile e execute o programa.

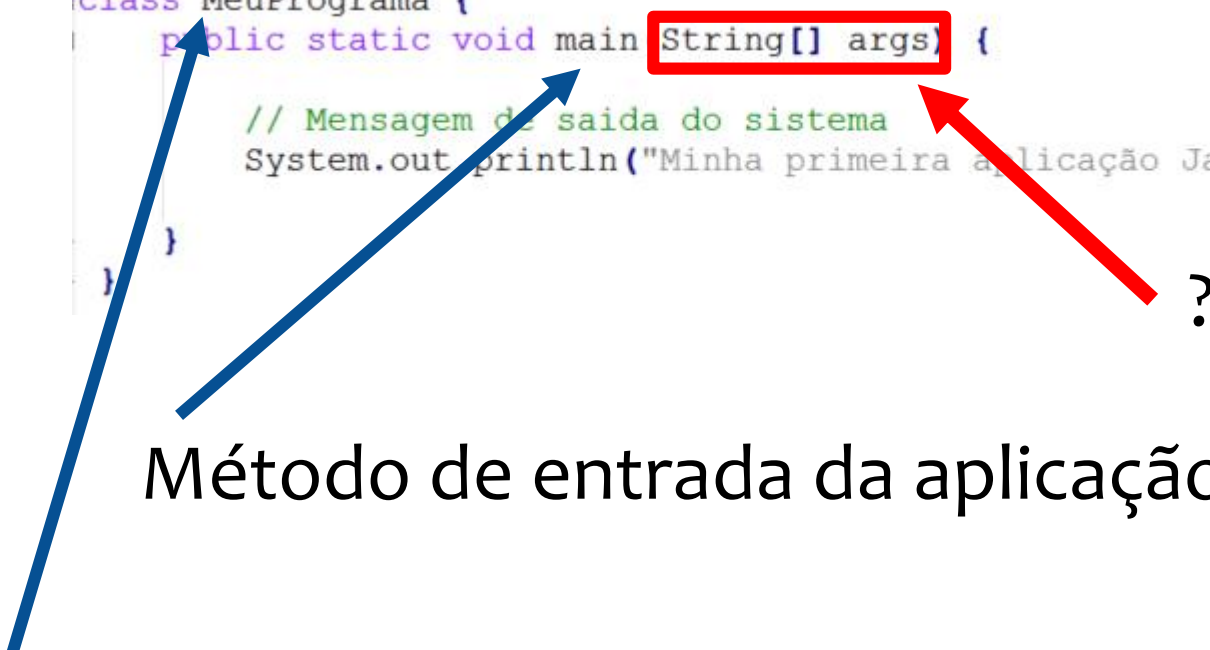
Leitura auxiliar: <http://www.devmedia.com.br/system-out-objeto-de-saida-em-java/25240>

PS: Crie a pasta “exercicio3”, mesmo que seja um exercício baseado no anterior.

Ex. Dirigido 1 – Método main

Para um programa Java executar, é necessário definir um método especial para ser o ponto de entrada do programa, ou seja, para ser o primeiro método a ser chamado quando o programa for executado. O método main precisa ser **public**, **static**, **void** e receber um array de strings como argumento.

```
class MeuPrograma {  
    public static void main(String[] args) {  
        // Mensagem de saída do sistema  
        System.out.println("Minha primeira aplicação Java!!");  
    }  
}
```



Método de entrada da aplicação.

Classe de entrada da aplicação. (nome da aplicação)

Ex. Dirigido 1 – args?

```
class ExemploArgs {  
    public static void main(String[] args) {  
  
        int i;  
        for( i = 0; i < args.length; i++) {  
            System.out.println( args[i] );  
        }  
    }  
}
```

Execução da virtual machine

Argumentos de entrada (args)

```
\exemplo_args>java ExemploArgs Aqui vão os argumentos, não esqueça que é um array
```

...

Nome da aplicação que quero rodar

arg[0] arg[1] arg[2]

arg[n-1]

Ex. Dirigido 1 – args?

```
class ExemploArgs {  
    public static void main(String[] args) {  
  
        int i;  
        for( i = 0; i < args.length; i++) {  
            System.out.println( args[i] );  
        }  
    }  
}
```

```
\exemplo_args>java ExemploArgs Aqui vão os argumentos, não esqueça que é um array
```

Como será a saída?

Ex. Dirigido 1 – args?

```
class ExemploArgs {  
    public static void main(String[] args) {  
  
        int i;  
        for( i = 0; i < args.length; i++) {  
            System.out.println( args[i] );  
        }  
    }  
}
```

```
\exemplo_args>java ExemploArgs Aqui vão os argumentos, não esqueça que é um array
```

```
Aqui  
vão  
os  
argumentos,  
não  
esqueça  
que  
é  
um  
array
```

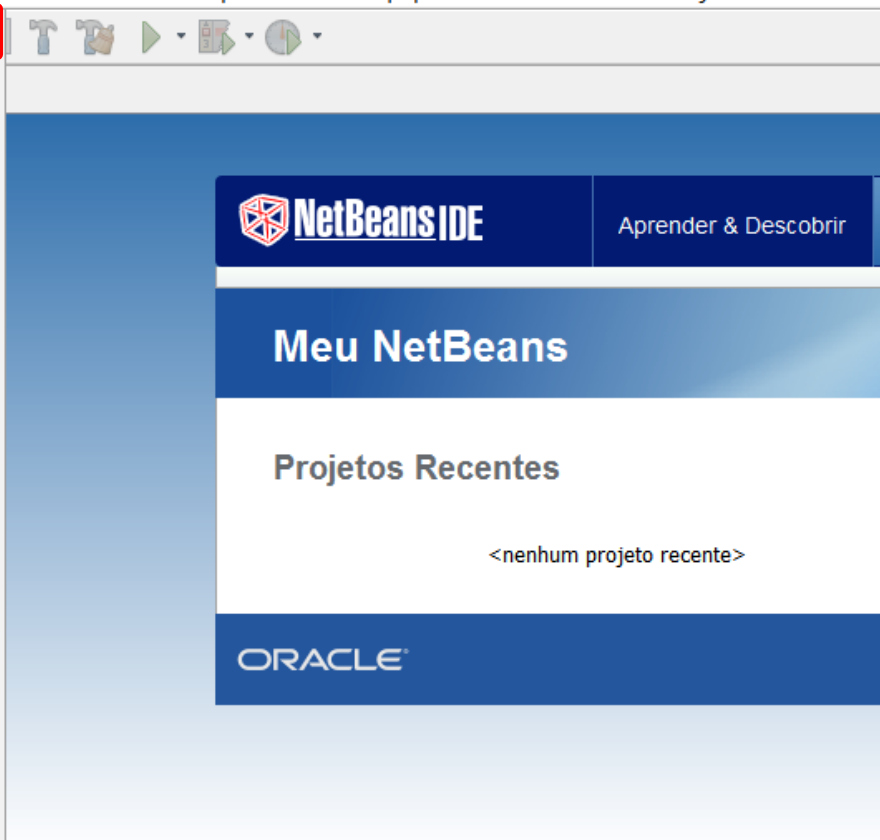
NetBeans

Ex. Dirigido – Criando uma aplicação Java

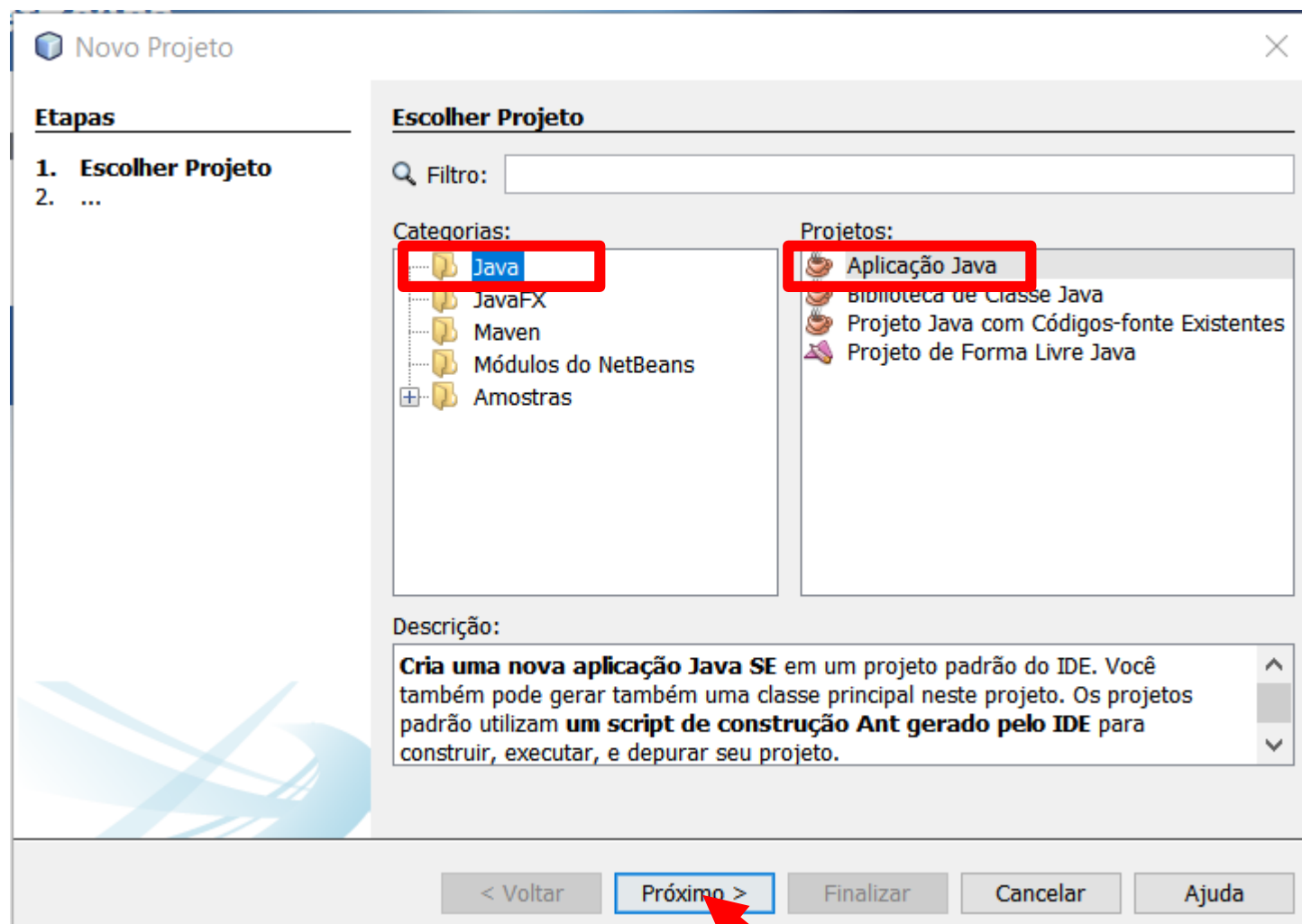
NetBeans IDE 8.2

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

- Novo Projeto...** Ctrl+Shift+N
- Novo Arquivo... Ctrl+N
- Abrir Projeto... Ctrl+Shift+O
 - Abrir Projeto Recente
 - Fechar Projeto
 - Fechar Outros Projetos
 - Fechar Todos os Projetos
- Abrir Arquivo...
 - Abrir Arquivo Recente
- Grupos de Projetos...
- Propriedades do Projeto ()
- Importar Projeto
- Exportar Projeto
- Salvar Ctrl+S
- Salvar como...
- Salvar Tudo Ctrl+Shift+S
- Configurar Página...
- Imprimir... Ctrl+Alt+Shift+P
- Imprimir em HTML...
- Sair



Ex. Dirigido 1 – Criando uma aplicação Java



Ex. Dirigido – Criando uma aplicação Java

Novo Aplicação Java

Etapas

1. Escolher Projeto
2. **Nome e Localização**

Nome e Localização

Nome do Projeto:

Localização do Projeto: Procurar...

Pasta do Projeto:

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

Pasta Bibliotecas: Procurar...

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar Classe Principal

Vai alterar automaticamente de acordo com o nome, deixe como está.

< Voltar Próximo > **Finalizar** Cancelar Ajuda

Ex. Dirigido – Criando uma aplicação Java

Seu projeto (nome da aplicação)

Nome do seu pacote, conterá todas classes da sua aplicação

Pacote da sua aplicação

The screenshot shows the NetBeans IDE 8.2 interface. The 'Projetos' (Projects) window on the left displays a project named 'Aula03'. Under 'Pacotes de Códigos-fonte' (Source Packages), there is a package named 'aula03'. Under 'Bibliotecas' (Libraries), there is a file named 'Aula03.java'. The 'Código-Fonte' (Source) window on the right shows the code for 'Aula03.java'. The code includes a package declaration 'package aula03;', a class declaration 'public class Aula03 {', and a main method 'public static void main(String[] args) {'. The main method contains a comment '// TODO code application logic here'.

```
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 Aula03 {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
```

Classe principal

Modelar uma classe para conta bancaria

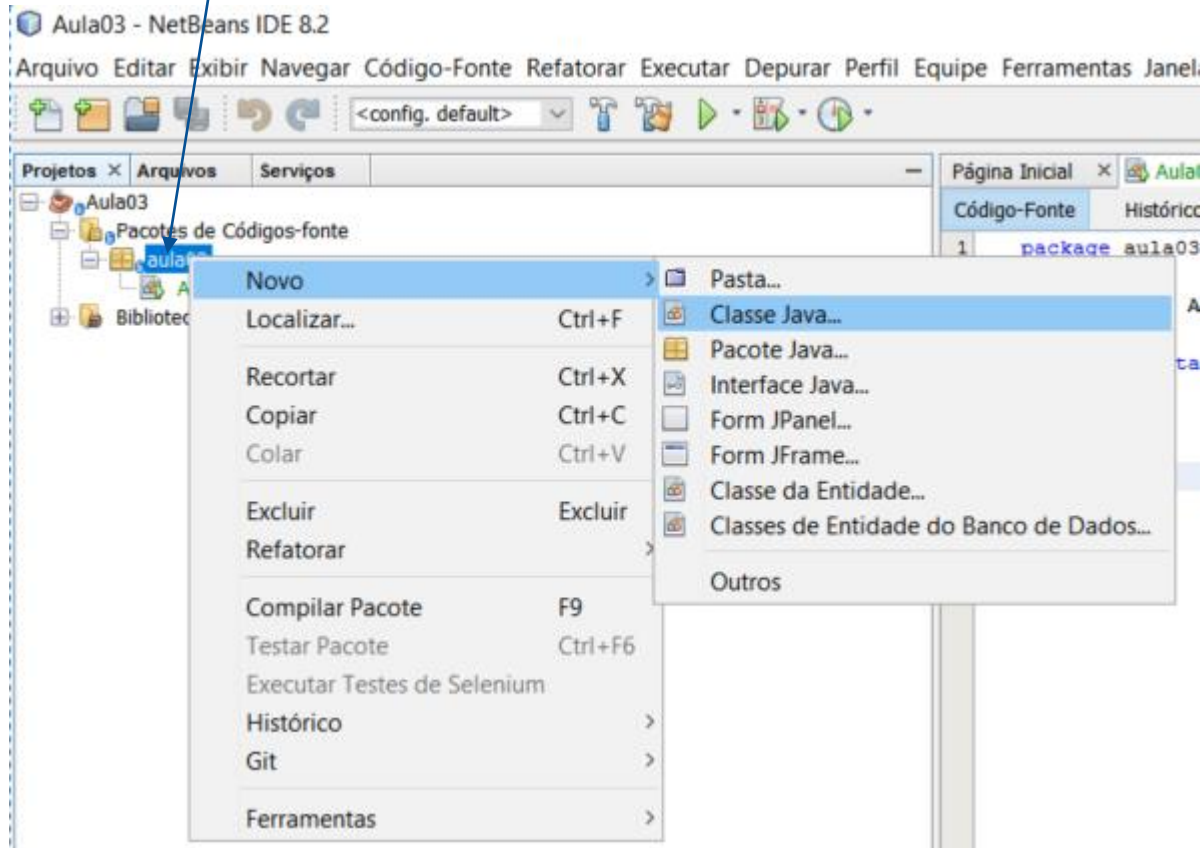
Conta
numero: inteiro titular: string saldo: real cpf: string
visualizarSaldo() depositar() sacar() transferirDinheiro()

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:

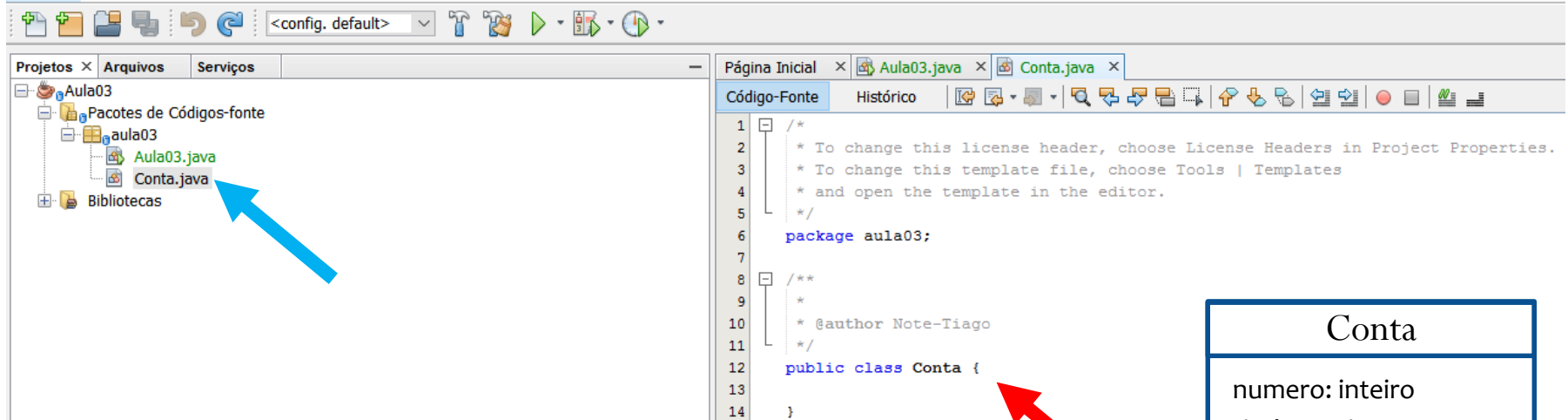
Pacote:

Arquivo Criado:

Ex. Dirigido

Aula03 - NetBeans IDE 8.2

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



Conta

numero: inteiro

titular: string

saldo: real

cpf: string

visualizarSaldo()

depositar()

sacar()

transferirDinheiro()

Ex. Dirigido

```
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:

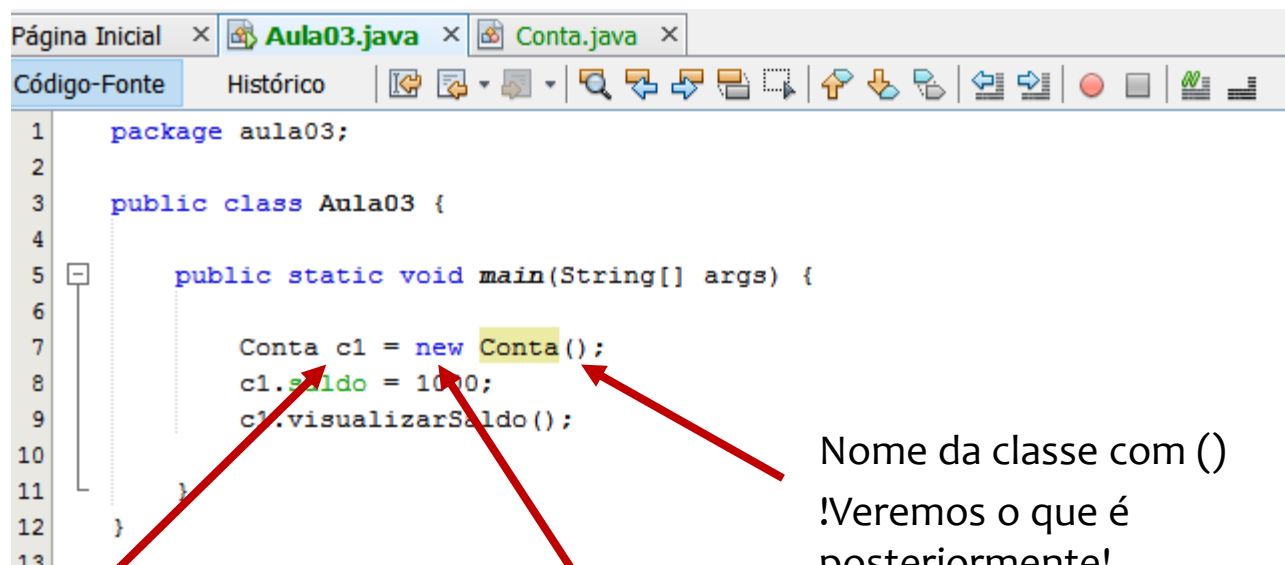
```
package aula03;  
  
public class Aula03 {  
    public static void main(String[] args) {  
        Conta c1;  
        c1 = new Conta();  
    }  
}
```



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
```

The screenshot shows an IDE with two tabs: 'Aula03.java' and 'Conta.java'. The 'Código-Fonte' (Source Code) view is active. The code in 'Aula03.java' is as follows:

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 comecemos 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");  
}
```

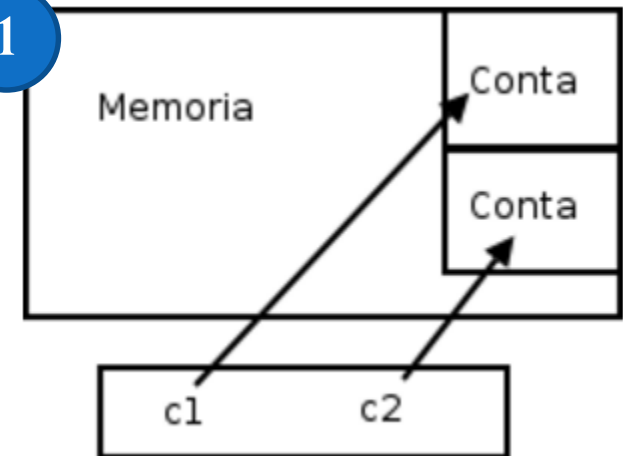
Apenas referência

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

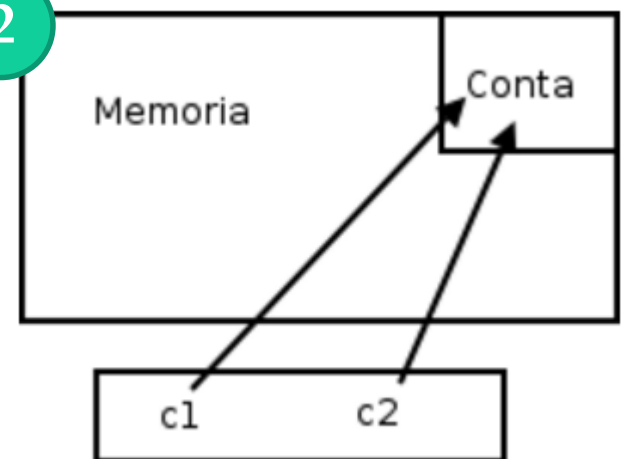
2

```
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



Perguntas?