



**Instituto Infnet**

**Fundamentos de Desenvolvimento com  
Java:**

# **Desenvolvimento Back-end**

**2025T3**


**AT**

**Professor: Elberth Lins Costa de Moraes**

**Aluno: Lucas Amorim Porciuncula**

## Sumário

Sumário.....	2
<b>Exercício 1 – Instalando e configurando o ambiente Java.....</b>	<b>4</b>
Ambiente Utilizado:.....	4
Configuração no Eclipse:.....	4
Criação do Projeto e Código:.....	5
Build e Execução:.....	5
<b>Exercício 2 – Validação de Senha Segura.....</b>	<b>6</b>
Contexto:.....	6
Requisitos da Senha:.....	6
Código Java – Validação de Senha Segura:.....	6
<b>Exercício 3 – Calculadora de Impostos.....</b>	<b>8</b>
Contexto:.....	8
Tabela de Imposto de Renda (Base Anual):.....	8
Regras do Programa:.....	8
Código Java – Calculadora de Impostos:.....	8
<b>Exercício 4 – Simulador de Empréstimo Bancário.....</b>	<b>10</b>
Contexto:.....	10
Regras do Exercício:.....	10
Fórmula de Juros Compostos usada:.....	11
Código Java – Simulador de Empréstimo:.....	12
<b>Exercício 5 – Criando um Programa CGI em Java.....</b>	<b>13</b>
Contexto:.....	13
Código Java – Simulação de Script CGI:.....	13
<b>Exercício 6 – Cadastro de Veículos.....</b>	<b>14</b>
Contexto:.....	14
Estrutura da Classe Veiculo:.....	14
Código Java – Cadastro de Veículos:.....	15
<b>Exercício 7 – Gerenciador de Alunos.....</b>	<b>17</b>
Contexto:.....	17
Estrutura da Classe Aluno:.....	17
Métodos obrigatórios:.....	17
Código Java – Classe Aluno:.....	18

Classe Main – Captura de Dados e Execução:.....	19
<b>Exercício 8 – Sistema de Funcionários.....</b>	<b>20</b>
Contexto:.....	20
Código Java – Classe Funcionario (Classe Pai):.....	21
Classe Gerente (Subclasse):.....	21
Classe Estagiario (Subclasse):.....	21
Classe Main – Testando o Sistema:.....	22
<b>Exercício 9 – Conta Bancária com Encapsulamento.....</b>	<b>23</b>
Contexto:.....	23
Código Java – Classe ContaBancaria:.....	24
Classe Main – Testando o Sistema:.....	25
<b>Exercício 10 – Registro de Compras em Arquivo.....</b>	<b>26</b>
Contexto:.....	26
Código Java – Registro e Leitura de Compras:.....	26
<b>Exercício 11 – Simulação de Loteria.....</b>	<b>29</b>
Contexto:.....	29
Regras:.....	29
Código Java – Loteria:.....	29
<b>Exercício 12 – Sistema de Chat Simples com Arrays.....</b>	<b>31</b>
Contexto:.....	31
Código Java – Sistema de Chat:.....	31
 <b>Repositório no GitHub.....</b>	<b>32</b>
Lucas-1234567890/Fundamentos-de-Desenvolvimento-com-Java-.....	32

## Exercício 1 – Instalando e configurando o ambiente Java

### Ambiente Utilizado:

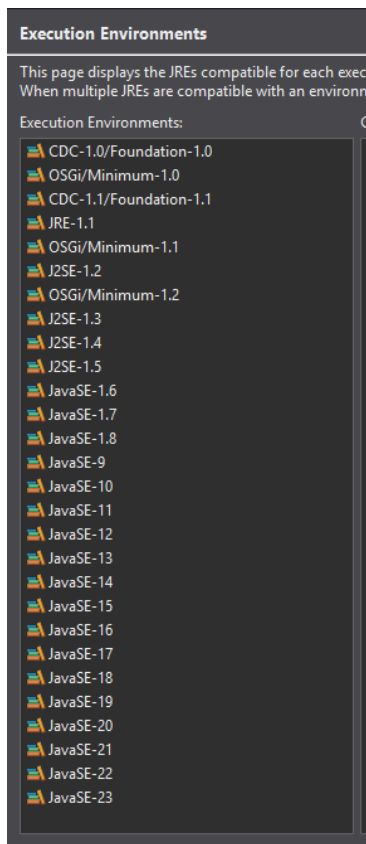
- **JDK:** Versão mais recente (verificado no terminal com `java -version`)

```
PS C:\Users\Lucas> java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
```

- **IDE:** Eclipse IDE for Java Developers (usei o Eclipse ao invés do IntelliJ, por preferência do professor)

### Configuração no Eclipse:

- Abri o Eclipse.
- Configurei o caminho do JDK dentro das configurações do Eclipse.



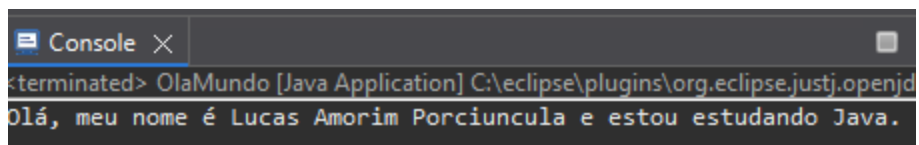
## Criação do Projeto e Código:

- Criei o projeto com o nome **MinhaPrimeiraApp**.
- Dentro do projeto, criei a classe **OlaMundo.java**.
- Código feito:

```
package br.edu.infnet;  
public class OlaMundo {  
    public static void main(String[] args) {  
        String meuNome = "Lucas Amorim Porciuncula";  
        System.out.println("Olá, meu nome é " + meuNome + " e estou estudando Java.");  
    }  
}
```

## Build e Execução:

- Compilei o projeto no Eclipse.
- Executei o programa pelo console da IDE.



The screenshot shows the Eclipse IDE's console window. The title bar reads "Console". The text in the console is as follows:  
x terminated> OlaMundo [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk  
Olá, meu nome é Lucas Amorim Porciuncula e estou estudando Java.

## Exercício 2 – Validação de Senha Segura

### Contexto:

O objetivo deste exercício é criar um programa simples de **validação de senha** que garanta que o usuário escolha uma senha forte antes de se cadastrar no sistema.

---

### Requisitos da Senha:

A senha precisa:

- ✓ Ter no mínimo 8 caracteres
- ✓ Conter pelo menos uma letra maiúscula
- ✓ Conter pelo menos um número
- ✓ Conter pelo menos um caractere especial (exemplo: @, #, \$, %, etc.)

Se a senha não atender a algum desses critérios, o programa deve **informar o motivo do erro** e pedir uma nova senha.

---

### Código Java – Validação de Senha Segura:



```
package br.edu.infnet;
import java.util.Scanner;
public class ex02 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String nome;
```

```

String senha;
boolean senhaValida = false;
System.out.print("Digite seu nome: ");
nome = scanner.nextLine();
while (!senhaValida) {
    System.out.print("Digite uma senha: ");
    senha = scanner.nextLine();
    boolean temTamanho = senha.length() >= 8;
    boolean temMaiuscula = senha.matches(".*[A-Z].*");
    boolean temNumero = senha.matches(".*\\d.*");
    boolean temEspecial = senha.matches(".*[!@#$%^&*().,?\\':{}|<>].*");
    if (!temTamanho) {
        System.out.println("Erro: A senha deve ter no mínimo 8 caracteres.");
    }
    if (!temMaiuscula) {
        System.out.println("Erro: A senha deve conter pelo menos uma letra maiúscula.");
    }
    if (!temNumero) {
        System.out.println("Erro: A senha deve conter pelo menos um número.");
    }
    if (!temEspecial) {
        System.out.println("Erro: A senha deve conter pelo menos um caractere especial.");
    }
    if (temTamanho && temMaiuscula && temNumero && temEspecial) {
        senhaValida = true;
        System.out.println("Senha cadastrada com sucesso! Bem-vindo, " + nome + "!");
    } else {
        System.out.println("Por favor, tente novamente.\n");
    }
}
scanner.close();
}

```

```

Console X
<terminated> ex02 [Java Application] C:\eclipse\plugins\org.eclipse.j
Digite seu nome: Lucas
Digite uma senha: lucas
Erro: A senha deve ter no mínimo 8 caracteres.
Erro: A senha deve conter pelo menos uma letra maiúscula.
Erro: A senha deve conter pelo menos um número.
Erro: A senha deve conter pelo menos um caractere especial.
Por favor, tente novamente.

Digite uma senha: lucas123456
Erro: A senha deve conter pelo menos uma letra maiúscula.
Erro: A senha deve conter pelo menos um caractere especial.
Por favor, tente novamente.

Digite uma senha: Lucas123456
Erro: A senha deve conter pelo menos um caractere especial.
Por favor, tente novamente.

Digite uma senha: Lucas12345$
Senha cadastrada com sucesso! Bem-vindo, Lucas!

```

## Exercício 3 – Calculadora de Impostos

### Contexto:

Agora o objetivo é criar um programa em Java que **calcule o Imposto de Renda Anual** com base no **salário mensal informado pelo usuário**, aplicando a tabela progressiva fornecida.

Tabela de Imposto de Renda (Base Anual):	
Tt Faixa de Renda Anual	Tt Alíquota
Até R\$ 22.847,76	Isento
De R\$ 22.847,77 a R\$ 33.919,80	7,5%
De R\$ 33.919,81 a R\$ 45.012,60	15%
Acima de R\$ 45.012,61	27,5%

### Regras do Programa:

- ✓ O usuário informa **nome** e **salário mensal**.
- ✓ O programa calcula o **salário anual**.
- ✓ Aplica a alíquota conforme a tabela.
- ✓ Mostra o **valor do imposto** e o **salário líquido anual** (ou seja: salário anual - imposto).

### Código Java – Calculadora de Impostos:



```

package br.edu.infnet;
import java.util.Scanner;
public class ex03 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Digite seu nome: ");
        String nome = scanner.nextLine();
        System.out.print("Digite seu salário mensal (R$): ");
        double salarioMensal = scanner.nextDouble();
        double salarioAnual = salarioMensal * 12;
        double imposto = 0;
        if (salarioAnual <= 22847.76) {
            System.out.println("Isento de imposto de renda.");
        } else if (salarioAnual <= 33919.80) {
            imposto = salarioAnual * 0.075;
        } else if (salarioAnual <= 45012.60) {
            imposto = salarioAnual * 0.15;
        } else {
            imposto = salarioAnual * 0.275;
        }
        double salarioLiquido = salarioAnual - imposto;
        System.out.println("\n--- Resultado ---");
        System.out.println("Nome: " + nome);
        System.out.printf("Salário Anual: R$ %.2f\n", salarioAnual);
        System.out.printf("Imposto a pagar: R$ %.2f\n", imposto);
        System.out.printf("Salário Líquido Anual: R$ %.2f\n", salarioLiquido);
        scanner.close();
    }
}

```

```

Console X
<terminated> ex03 [Java Application] C:\ecl
Digite seu nome: lucas
Digite seu salário mensal (R$): 1414
Isento de imposto de renda.

--- Resultado ---
Nome: lucas
Salário Anual: R$ 16968,00
Imposto a pagar: R$ 0,00
Salário Líquido Anual: R$ 16968,00

```

```

Console X
<terminated> ex03 [Java Application] C:\ecl
Digite seu nome: Lucas
Digite seu salário mensal (R$): 2500
|
--- Resultado ---
Nome: Lucas
Salário Anual: R$ 30000,00
Imposto a pagar: R$ 2250,00
Salário Líquido Anual: R$ 27750,00

```

```

Console X
<terminated> ex03 [Java Application] C:\eclipse\plugin
Digite seu nome: Lucas - Meu salario no futuro
Digite seu salário mensal (R$): 8000
|
--- Resultado ---
Nome: Lucas - Meu salario no futuro
Salário Anual: R$ 96000,00
Imposto a pagar: R$ 26400,00
Salário Líquido Anual: R$ 69600,00

```

## Exercício 4 – Simulador de Empréstimo Bancário

### Contexto:

Um banco precisa de um sistema simples para **simular empréstimos**, calculando o valor final com juros e o valor de cada parcela.

---

### Regras do Exercício:

✓ O usuário informa:

- Nome do cliente
- Valor do empréstimo
- Quantidade de parcelas (**mínimo 6, máximo 48**)

✓ O banco aplica **juros fixos de 3% ao mês**.

✓ O programa deve mostrar:

- **Valor total a pagar** (com juros)
  - **Valor de cada parcela**
-

**Fórmula de Juros Compostos usada:**

Como é um juros fixo por mês e acumulado, vamos usar a fórmula de **montante com juros compostos**:

mathematica

$$\text{Valor Total} = \text{Valor Emprestado} \times (1 + \text{Taxa})^{\text{Número de Parcelas}}$$

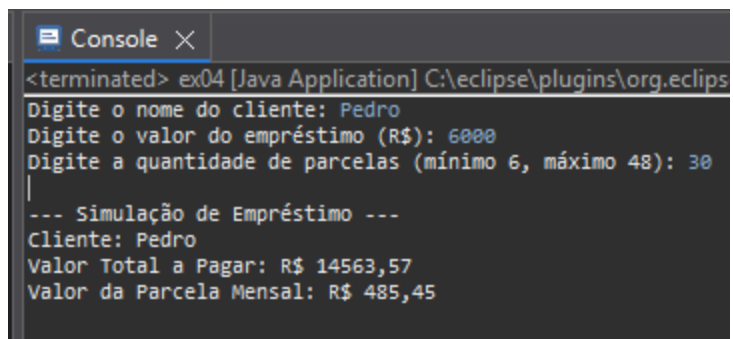
**Depois, pra achar a parcela mensal:**

$$\text{Parcela Mensal} = \text{Valor Total} / \text{Número de Parcelas}$$

---

## Código Java – Simulador de Empréstimo:

```
package br.edu.infnet;
import java.util.Scanner;
public class ex04 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Digite o nome do cliente: ");
        String nome = scanner.nextLine();
        System.out.print("Digite o valor do empréstimo (R$): ");
        double valorEmprestimo = scanner.nextDouble();
        int parcelas;
        do {
            System.out.print("Digite a quantidade de parcelas (mínimo 6, máximo 48): ");
            parcelas = scanner.nextInt();
            if (parcelas < 6 || parcelas > 48) {
                System.out.println("Erro: Número de parcelas inválido. Tente novamente.");
            }
        } while (parcelas < 6 || parcelas > 48);
        double taxaJuros = 0.03; // 3% ao mês
        double valorTotal = valorEmprestimo * Math.pow(1 + taxaJuros, parcelas);
        double valorParcela = valorTotal / parcelas;
        System.out.println("\n--- Simulação de Empréstimo ---");
        System.out.println("Cliente: " + nome);
        System.out.printf("Valor Total a Pagar: R$ %.2f\n", valorTotal);
        System.out.printf("Valor da Parcela Mensal: R$ %.2f\n", valorParcela);
        scanner.close();
    }
}
```



The screenshot shows a console window titled "Console" with the following output:

```
<terminated> ex04 [Java Application] C:\eclipse\plugins\org.eclips
Digite o nome do cliente: Pedro
Digite o valor do empréstimo (R$): 6000
Digite a quantidade de parcelas (mínimo 6, máximo 48): 30
|
--- Simulação de Empréstimo ---
Cliente: Pedro
Valor Total a Pagar: R$ 14563,57
Valor da Parcela Mensal: R$ 485,45
```

## Exercício 5 – Criando um Programa CGI em Java

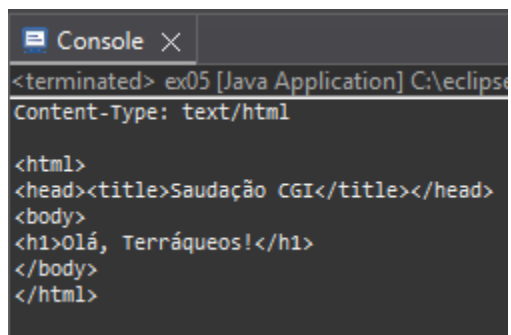
### Contexto:

Antes dos frameworks web modernos, os servidores HTTP usavam **CGI (Common Gateway Interface)** para gerar conteúdo dinâmico. O CGI executava programas externos que devolviam uma **resposta HTTP completa**, incluindo **headers** e **HTML**.

O foco desse exercício é **simular a saída de um script CGI em Java**, apenas **executando o programa pela linha de comando**, sem rodar num servidor.

### Código Java – Simulação de Script CGI:

```
package br.edu.infnet;
public class ex05 {
    public static void main(String[] args) {
        // Header HTTP
        System.out.println("Content-Type: text/html");
        System.out.println();
        // Corpo da resposta (HTML)
        System.out.println("<html>");
        System.out.println("<head><title>Saudação CGI</title></head>");
        System.out.println("<body>");
        System.out.println("<h1>Olá, Terráqueos!</h1>");
        System.out.println("</body>");
        System.out.println("</html>");
    }
}
```



```
Console X
<terminated> ex05 [Java Application] C:\eclipse
Content-Type: text/html

<html>
<head><title>Saudação CGI</title></head>
<body>
<h1>Olá, Terráqueos!</h1>
</body>
</html>
```

## Exercício 6 – Cadastro de Veículos

### Contexto:

A missão agora é simular um **sistema de gestão de veículos para uma locadora**, usando conceitos de **POO (Programação Orientada a Objetos)** em Java.

---

### Estrutura da Classe Veiculo:

A classe deve ter os seguintes **atributos**:

Estrutura da Classe Veiculo:	
Tt Atributo	Tt Tipo
Placa	String
Modelo	String
Ano de Fabricação	int
Quilometragem	double

métodos	
Tt Metodos	Tt Função
exibirDetalhes()	Mostra todas as informações do veículo
registrarViagem(double km)	Soma o km informado a quilometragem total

## Código Java – Cadastro de Veículos:

### Classe

```
package br.edu.infnet;
public class Veiculo {
    private String placa;
    private String modelo;
    private int anoFabricacao;
    private double quilometragem;
    // Construtor
    public Veiculo(String placa, String modelo, int anoFabricacao, double quilometragem) {
        this.placa = placa;
        this.modelo = modelo;
        this.anoFabricacao = anoFabricacao;
        this.kilometragem = quilometragem;
    }
    // Método para exibir detalhes
    public void exibirDetalhes() {
        System.out.println("Placa: " + placa);
        System.out.println("Modelo: " + modelo);
        System.out.println("Ano de Fabricação: " + anoFabricacao);
        System.out.println("Quilometragem: " + quilometragem + " km");
        System.out.println("-----");
    }
    // Método para registrar uma viagem
    public void registrarViagem(double km) {
        if (km > 0) {
            quilometragem += km;
            System.out.println("Viagem registrada: +" + km + " km");
        } else {
            System.out.println("Erro: Quilometragem inválida!");
        }
    }
}
```

## Main

```

package br.edu.infnet;
public class ex06Main {
    public static void main(String[] args) {
        // Criando dois veículos com dados fictícios
        Veiculo carro1 = new Veiculo("ABC-1234", "Toyota Corolla", 2018, 45000);
        Veiculo carro2 = new Veiculo("XYZ-5678", "Honda Civic", 2020, 32000);
        // Exibindo detalhes iniciais
        System.out.println("--- Detalhes Iniciais ---");
        carro1.exibirDetalhes();
        carro2.exibirDetalhes();
        // Registrando viagens
        System.out.println("--- Registrando Viagens ---");
        carro1.registrarViagem(150);
        carro2.registrarViagem(300);
        // Exibindo detalhes atualizados
        System.out.println("\n--- Detalhes Após as Viagens ---");
        carro1.exibirDetalhes();
        carro2.exibirDetalhes();
    }
}

```

```

Console X
<terminated> ex06Main [Java Application]
--- Detalhes Iniciais ---
Placa: ABC-1234
Modelo: Toyota Corolla
Ano de Fabricação: 2018
Quilometragem: 45000.0 km
-----
Placa: XYZ-5678
Modelo: Honda Civic
Ano de Fabricação: 2020
Quilometragem: 32000.0 km
-----
--- Registrando Viagens ---
Viagem registrada: +150.0 km
Viagem registrada: +300.0 km
--- Detalhes Após as Viagens ---
Placa: ABC-1234
Modelo: Toyota Corolla
Ano de Fabricação: 2018
Quilometragem: 45150.0 km
-----
Placa: XYZ-5678
Modelo: Honda Civic
Ano de Fabricação: 2020
Quilometragem: 32300.0 km
-----

```



## Exercício 7 – Gerenciador de Alunos

### Contexto:

Agora estamos criando um **sistema acadêmico simples**. O objetivo é **cadastrar alunos**, **calcular a média das notas** e **dizer se o aluno foi aprovado ou reprovado**.

---

Estrutura da Classe Aluno:	
Tt Atributo	Tt Tipo
nome	String
matricula	String
nota1	double
nota2	double
nota3	double

Métodos obrigatórios:	
Tt Método	Tt Função
calcularMedia()	Calcula e retorna a média das 3 notas
verificarAprovacao()	Exibe se o aluno foi aprovado ou reprovado

## Código Java – Classe Aluno:

```
package br.edu.infnet.models;

public class Aluno {

    // Atributos
    private String nome;
    private String matricula;
    private double nota1;
    private double nota2;
    private double nota3;

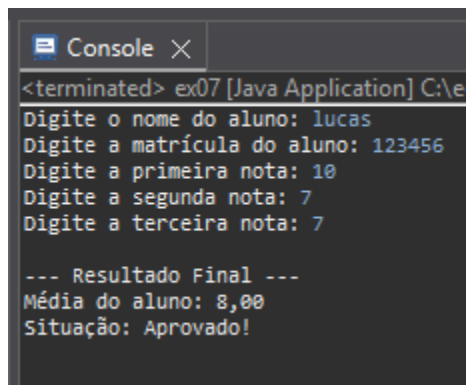
    // Construtor
    public Aluno(String nome, String matricula, double nota1, double nota2, double nota3) {
        this.nome = nome;
        this.matricula = matricula;
        this.nota1 = nota1;
        this.nota2 = nota2;
        this.nota3 = nota3;
    }

    // Método para calcular a média
    public double calcularMedia() {
        return (nota1 + nota2 + nota3) / 3;
    }

    // Método para verificar aprovação
    public void verificarAprovacao() {
        double media = calcularMedia();
        System.out.printf("Média do aluno: %.2f\n", media);
        if (media >= 7.0) {
            System.out.println("Situação: Aprovado!");
        } else {
            System.out.println("Situação: Reprovado!");
        }
    }
}
```

## Classe Main – Captura de Dados e Execução:

```
package br.edu.infnet;
import java.util.Scanner;
import br.edu.infnet.models.Aluno;
public class ex07 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Captura de dados do aluno
        System.out.print("Digite o nome do aluno: ");
        String nome = scanner.nextLine();
        System.out.print("Digite a matrícula do aluno: ");
        String matricula = scanner.nextLine();
        System.out.print("Digite a primeira nota: ");
        double nota1 = scanner.nextDouble();
        System.out.print("Digite a segunda nota: ");
        double nota2 = scanner.nextDouble();
        System.out.print("Digite a terceira nota: ");
        double nota3 = scanner.nextDouble();
        // Criando objeto Aluno
        Aluno aluno = new Aluno(nome, matricula, nota1, nota2, nota3);
        // Exibindo resultado
        System.out.println("\n--- Resultado Final ---");
        aluno.verificarAprovacao();
        scanner.close();
    }
}
```



```
Console X
<terminated> ex07 [Java Application] C:\e
Digite o nome do aluno: lucas
Digite a matrícula do aluno: 123456
Digite a primeira nota: 10
Digite a segunda nota: 7
Digite a terceira nota: 7

--- Resultado Final ---
Média do aluno: 8,00
Situação: Aprovado!
```

## Exercício 8 – Sistema de Funcionários

### Contexto:

Agora vamos entrar no mundo da Herança em Java (POO).

A ideia é criar um sistema de cálculo de salários, onde temos um Funcionario base, e dois tipos de funcionários com regras diferentes de cálculo:

Tipos	
Tt Atributo	Tt Tipo
Gerente	Ganha um bônus de 20% do salário base
Estagiário	Tem um desconto de 10% do salário base

Classe Pai: Funcionário	
Tt Atributo	Tt Tipo
nome	String
salarioBase	double

subclasses	
Tt Atributo	Tt Tipo
Gerente	Adiciona 20% do bônus

subclasses	
Tt Atributo	Tt Tipo
Estagiário	Aplica 10% de desconto

### Código Java – Classe Funcionario (Classe Pai):

```
package br.edu.infnet.models;
public class Funcionario {
    protected String nome;
    protected double salarioBase;
    public Funcionario(String nome, double salarioBase) {
        this.nome = nome;
        this.salarioBase = salarioBase;
    }
    public double calcularSalario() {
        return salarioBase; // Regra padrão (sem alteração)
    }
    public void exibirSalario() {
        System.out.printf("%s - Salário Final: R$ %.2f\n", nome, calcularSalario());
    }
}
```

### Classe Gerente (Subclasse):

```
package br.edu.infnet.models;
public class Gerente extends Funcionario {
    public Gerente(String nome, double salarioBase) {
        super(nome, salarioBase);
    }
    @Override
    public double calcularSalario() {
        return salarioBase * 1.20; // Salário + 20%
    }
}
```

### Classe Estagiario (Subclasse):

```
package br.edu.infnet.models;
public class Estagiario extends Funcionario {
    public Estagiario(String nome, double salarioBase) {
        super(nome, salarioBase);
    }
    @Override
```

```
public double calcularSalario() {  
    return salarioBase * 0.90; // Salário - 10%  
}  
}
```

## Classe Main – Testando o Sistema:

```
package br.edu.infnet;  
import br.edu.infnet.models.Estagiario;  
import br.edu.infnet.models.Funcionario;  
import br.edu.infnet.models.Gerente;  
public class ex08 {  
    public static void main(String[] args) {  
        // Criando um Gerente e um Estagiário  
        Funcionario gerente = new Gerente("Carlos - Gerente", 5000);  
        Funcionario estagiario = new Estagiario("Lucas - Estagiário", 2000);  
        // Exibindo os salários finais  
        System.out.println("--- Salários Finais ---");  
        gerente.exibirSalario();  
        estagiario.exibirSalario();  
    }  
}
```

```
Console X  
<terminated> ex08 [Java Application] C:\eclipse\plugin  
--- Salários Finais ---  
Carlos - Gerente - Salário Final: R$ 6000,00  
Lucas - Estagiário - Salário Final: R$ 1800,00  
|
```

## Exercício 9 – Conta Bancária com Encapsulamento

### Contexto:

Agora o foco é **Encapsulamento (POO)**.

Queremos proteger o atributo **saldo** para que **ninguém consiga alterar direto**. Só vai mudar o saldo através dos **métodos da própria classe**.

Classe ContaBancaria	
Tt Atributo	Tt Tipo
titular	String
saldo	double

Metodos obrigatórios	
Tt Método	Tt Função
depositar(valor)	Adiciona valor ao saldo
sacar(valor)	subtrai valor, mas só se tiver saldo suficiente

## Código Java – Classe ContaBancaria:

```
package br.edu.infnet.models;

public class ContaBancaria {
    public String titular;
    private double saldo;

    // Construtor
    public ContaBancaria(String titular) {
        this.titular = titular;
        this.saldo = 0.0;
    }

    // Método para depositar
    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
            System.out.printf("Depósito de R$ %.2f realizado com sucesso.\n", valor);
        } else {
            System.out.println("Valor de depósito inválido!");
        }
    }

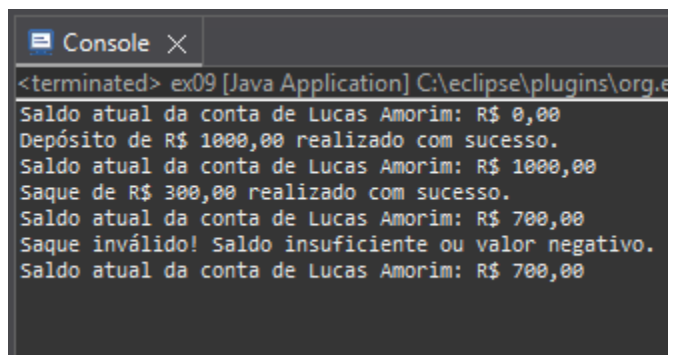
    // Método para sacar
    public void sacar(double valor) {
        if (valor > 0 && valor <= saldo) {
            saldo -= valor;
            System.out.printf("Saque de R$ %.2f realizado com sucesso.\n", valor);
        } else {
            System.out.println("Saque inválido! Saldo insuficiente ou valor negativo.");
        }
    }

    // Método para exibir saldo
    public void exibirSaldo() {
        System.out.printf("Saldo atual da conta de %s: R$ %.2f\n", titular, saldo);
    }
}
```



## Classe Main – Testando o Sistema:

```
package br.edu.infnet;  
import br.edu.infnet.models.ContaBancaria;  
public class ex09 {  
    public static void main(String[] args) {  
        // Criando a conta  
        ContaBancaria conta = new ContaBancaria("Lucas Amorim");  
        // Operações  
        conta.exibirSaldo();  
        conta.depositar(1000);  
        conta.exibirSaldo();  
        conta.sacar(300);  
        conta.exibirSaldo();  
        conta.sacar(800); // Teste de saque maior que o saldo  
        conta.exibirSaldo();  
    }  
}
```



Console X

<terminated> ex09 [Java Application] C:\eclipse\plugins\org.e

Saldo atual da conta de Lucas Amorim: R\$ 0,00  
Depósito de R\$ 1000,00 realizado com sucesso.  
Saldo atual da conta de Lucas Amorim: R\$ 1000,00  
Saque de R\$ 300,00 realizado com sucesso.  
Saldo atual da conta de Lucas Amorim: R\$ 700,00  
Saque inválido! Saldo insuficiente ou valor negativo.  
Saldo atual da conta de Lucas Amorim: R\$ 700,00

## Exercício 10 – Registro de Compras em Arquivo

### Contexto:

Agora vamos trabalhar com **gravação e leitura de arquivos em Java** (**FileWriter / FileReader / BufferedReader**).

### Objetivo:

1. ☒ Pegar os dados de **3 compras diferentes** via teclado.
2. ☒ Salvar tudo num arquivo chamado **compras.txt**.
3. ☒ Ler o arquivo e exibir o conteúdo no console.

Estrutura dos Dados da Compra	
Tt Campo	Tt Tipo
Produto	String
Quantidade	int
Preço Unitário	double

### Código Java – Registro e Leitura de Compras:

```

package br.edu.infnet;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class ex10 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String caminhoArquivo = "compras.txt";
        try {

            FileWriter writer = new FileWriter(caminhoArquivo);
            System.out.println("--- Cadastro de Compras ---");
            for (int i = 1; i <= 3; i++) {
                System.out.printf("\nCompra %d:\n", i);
                System.out.print("Produto: ");
                String produto = scanner.nextLine();
                System.out.print("Quantidade: ");
                int quantidade = Integer.parseInt(scanner.nextLine());
                System.out.print("Preço Unitário: ");
                double preco = Double.parseDouble(scanner.nextLine());
                // Grava no arquivo (Exemplo de linha: ProdutoX;5;10.50)
                writer.write(produto + ";" + quantidade + ";" + preco + "\n");
            }
            writer.close();
            System.out.println("\n✓ Dados salvos com sucesso em " + caminhoArquivo);
            // Parte 2: Leitura e exibição
            System.out.println("\n--- Compras Registradas ---");
            BufferedReader reader = new BufferedReader(new FileReader(caminhoArquivo));
            String linha;
            while ((linha = reader.readLine()) != null) {
                String[] partes = linha.split(";");
                String produto = partes[0];
                int quantidade = Integer.parseInt(partes[1]);
                double preco = Double.parseDouble(partes[2]);
                System.out.printf("Produto: %s | Quantidade: %d | Preço Unitário: R$ %.2f\n", produto, quantidade, preco);
            }
            reader.close();
        } catch (IOException e) {
            System.out.println("Erro ao acessar o arquivo: " + e.getMessage());
        }
    }
}

```

```
    } catch (NumberFormatException e) {  
        System.out.println("Erro de formato numérico: " + e.getMessage());  
    } finally {  
        scanner.close();  
    }  
}  
}
```

```
Console X  
<terminated> ex10 [Java Application] C:\eclipse\plugins\org.eclipse.justi  
--- Cadastro de Compras ---  
  
Compra 1:  
Produto: Macarrão  
Quantidade: 10  
Preço Unitário: 90  
  
Compra 2:  
Produto: arroz  
Quantidade: 90  
Preço Unitário: 80  
  
Compra 3:  
Produto: agua  
Quantidade: 90  
Preço Unitário: 3  
  
✓ Dados salvos com sucesso em compras.txt  
  
--- Compras Registradas ---  
Produto: Macarrão | Quantidade: 10 | Preço Unitário: R$ 90,00  
Produto: arroz | Quantidade: 90 | Preço Unitário: R$ 80,00  
Produto: agua | Quantidade: 90 | Preço Unitário: R$ 3,00
```

## Exercício 11 – Simulação de Loteria

### Contexto:

Agora é hora de brincar com:

- ✓ **Números Aleatórios**
  - ✓ **Arrays**
  - ✓ **Comparação de Dados**
- 

### Regras:

1. O sistema sorteia **6 números aleatórios entre 1 e 60** (tipo Mega-Sena).
  2. O usuário digita **6 palpites diferentes**.
  3. O programa compara e mostra **quantos acertos** o usuário teve.
- 

### Código Java – Loteria:

```
package br.edu.infnet;
import java.util.HashSet;
import java.util.Random;
import java.util.Scanner;
import java.util.Set;
public class ex11 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        Set<Integer> numerosSorteados = new HashSet<>();
        Set<Integer> palpitesUsuario = new HashSet<>();
        // Gerando os 6 números aleatórios
```

```

while (numerosSorteados.size() < 6) {
    int numero = random.nextInt(60) + 1; // Número entre 1 e 60
    numerosSorteados.add(numero);
}
System.out.println("--- Bem-vindo ao Jogo de Loteria ---");
System.out.println("Digite 6 números entre 1 e 60 (sem repetir:");
// Capturando os 6 palpites do usuário
while (palpitesUsuario.size() < 6) {
    System.out.print("Digite o número " + (palpitesUsuario.size() + 1) + ": ");
    int palpite = scanner.nextInt();
    if (palpite < 1 || palpite > 60) {
        System.out.println("Número inválido! Digite entre 1 e 60.");
    } else if (palpitesUsuario.contains(palpite)) {
        System.out.println("Número repetido! Escolha outro.");
    } else {
        palpitesUsuario.add(palpite);
    }
}
// Calculando os acertos
Set<Integer> acertos = new HashSet<>(palpitesUsuario);
acertos.retainAll(numerosSorteados);
// Exibindo resultado
System.out.println("\nNúmeros Sorteados: " + numerosSorteados);
System.out.println("Seus Palpites: " + palpitesUsuario);
System.out.println("Total de acertos: " + acertos.size());
System.out.println("Números que você acertou: " + acertos);
scanner.close();
}
}

```

```

<terminated> ex11 [Java Application] C:\eclipse\plug
--- Bem-vindo ao Jogo de Loteria ---
Digite 6 números entre 1 e 60 (sem repetir):
Digite o número 1: 6
Digite o número 2: 87
Número inválido! Digite entre 1 e 60.
Digite o número 2: 56
Digite o número 3: 43
Digite o número 4: 12
Digite o número 5: 3
Digite o número 6: 9

Números Sorteados: [33, 34, 37, 38, 40, 24]
Seus Palpites: [3, 6, 56, 9, 43, 12]
Total de acertos: 0
Números que você acertou: []
|

```

## Exercício 12 – Sistema de Chat Simples com Arrays

### Contexto:

Agora é um exercício que envolve:

- ✓ Arrays
- ✓ Laços de repetição
- ✓ Entrada de texto com Scanner
- ✓ Controle de alternância entre usuários

### Código Java – Sistema de Chat:

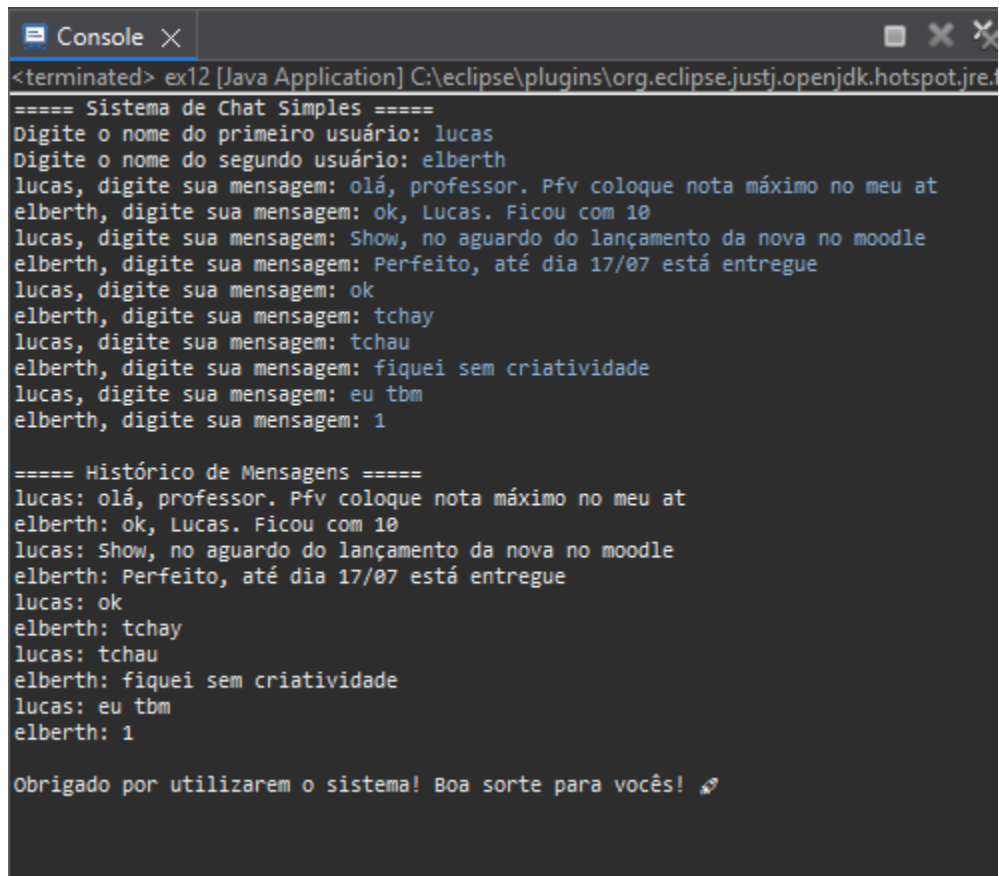
```
package br.edu.infnet;
import java.util.Scanner;
public class ex12 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] mensagens = new String[10];
        System.out.println("==== Sistema de Chat Simples =====");
        // Capturando os nomes dos usuários
        System.out.print("Digite o nome do primeiro usuário: ");
        String usuario1 = scanner.nextLine();
        System.out.print("Digite o nome do segundo usuário: ");
        String usuario2 = scanner.nextLine();
        // Variável pra controlar de quem é a vez
        String usuarioAtual;
        for (int i = 0; i < 10; i++) {
            // Alterna entre usuário1 e usuário2
            usuarioAtual = (i % 2 == 0) ? usuario1 : usuario2;
            System.out.print(usuarioAtual + ", digite sua mensagem: ");
            String mensagem = scanner.nextLine();
            // Salva a mensagem formatada no array
            mensagens[i] = usuarioAtual + ": " + mensagem;
        }
        // Exibindo o histórico de mensagens
        System.out.println("\n==== Histórico de Mensagens =====");
```

```

    for (String msg : mensagens) {
        System.out.println(msg);
    }

    // Mensagem de despedida
    System.out.println("\nObrigado por utilizarem o sistema! Boa sorte para vocês! 🚀");
    scanner.close();
}
}

```



```

<terminated> ex12 [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre...
===== Sistema de Chat Simples =====
Digite o nome do primeiro usuário: lucas
Digite o nome do segundo usuário: elberth
lucas, digite sua mensagem: olá, professor. Pfv coloque nota máximo no meu at
elberth, digite sua mensagem: ok, Lucas. Ficou com 10
lucas, digite sua mensagem: Show, no aguardo do lançamento da nova no moodle
elberth, digite sua mensagem: Perfeito, até dia 17/07 está entregue
lucas, digite sua mensagem: ok
elberth, digite sua mensagem: tchay
lucas, digite sua mensagem: tchau
elberth, digite sua mensagem: fiquei sem criatividade
lucas, digite sua mensagem: eu tbm
elberth, digite sua mensagem: 1

===== Histórico de Mensagens =====
lucas: olá, professor. Pfv coloque nota máximo no meu at
elberth: ok, Lucas. Ficou com 10
lucas: Show, no aguardo do lançamento da nova no moodle
elberth: Perfeito, até dia 17/07 está entregue
lucas: ok
elberth: tchay
lucas: tchau
elberth: fiquei sem criatividade
lucas: eu tbm
elberth: 1

Obrigado por utilizarem o sistema! Boa sorte para vocês! 🚀

```

## Repositório no GitHub

[Lucas-1234567890/Fundamentos-de-Desenvolvimento-com-Java-](https://github.com/Lucas-1234567890/Fundamentos-de-Desenvolvimento-com-Java-)