Apostila de Java: Do Básico ao Avançado

Sumário

---

1. Introdução ao Java

O que é Java?

Java é uma linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems (atualmente Oracle) em 1995. É uma das linguagens mais populares do mundo, utilizada em diversos tipos de aplicações, desde aplicativos móveis até sistemas empresariais complexos.

Características do Java

- **Orientada a objetos**: Tudo em Java é objeto (com exceção dos tipos primitivos)
- **Portável**: "Write once, run anywhere" - o código compilado (bytecode) pode ser executado em qualquer plataforma com uma JVM
- **Robusta**: Possui verificação de erros em tempo de compilação e execução
- **Segura**: Fornece um ambiente de execução seguro
- **Multithread**: Suporte nativo para programação concorrente

JVM, JRE e JDK

- **JVM (Java Virtual Machine)**: Máquina virtual que executa o bytecode Java
- **JRE (Java Runtime Environment)**: Ambiente de execução que inclui a JVM e bibliotecas padrão
- **JDK (Java Development Kit)**: Kit de desenvolvimento que inclui o JRE, compilador e ferramentas de desenvolvimento

2. Configurando o Ambiente

Instalando o JDK

1. Acesse o site da Oracle (https://www.oracle.com/java/technologies/javase-downloads.html)
2. Baixe a versão mais recente do JDK
3. Execute o instalador e siga as instruções
4. Configure a variável de ambiente JAVA_HOME apontando para o diretório de instalação do JDK
5. Adicione o caminho do binário do JDK à variável PATH

Verificando a instalação

Abra o terminal ou prompt de comando e digite:

```
java -version
javac -version
```

Primeiro programa em Java

```
public class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Olá, Mundo!");
    }
}
```

Compilando e executando

```
javac OlaMundo.java
java OlaMundo
```

3. Fundamentos da Linguagem

Variáveis e Tipos de Dados

```
public class TiposDados {
    public static void main(String[] args) {
        // Tipos primitivos
        byte idade = 25;                    // 8 bits
        short ano = 2023;                   // 16 bits
        int populacao = 210000000;          // 32 bits
        long habitantesTerra = 7800000000L; // 64 bits

        float altura = 1.75f;               // 32 bits
        double peso = 68.5;                 // 64 bits

        char letra = 'A';                   // 16 bits (Unicode)
        boolean estaChovendo = false;       // 1 bit (true/false)

        // Tipo de referência (String não é primitivo)
        String nome = "João Silva";
```

```
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade);
        System.out.println("Altura: " + altura);
    }
}
```

Operadores

```
public class Operadores {
    public static void main(String[] args) {
        // Operadores aritméticos
        int a = 10, b = 3;
        System.out.println("a + b = " + (a + b));  // 13
        System.out.println("a - b = " + (a - b));  // 7
        System.out.println("a * b = " + (a * b));  // 30
        System.out.println("a / b = " + (a / b));  // 3
        System.out.println("a % b = " + (a % b));  // 1

        // Operadores de atribuição
        int x = 5;
        x += 3;  // Equivalente a x = x + 3
        System.out.println("x = " + x);  // 8

        // Operadores de comparação
        System.out.println("a == b: " + (a == b));  // false
        System.out.println("a != b: " + (a != b));  // true
        System.out.println("a > b: " + (a > b));    // true

        // Operadores lógicos
        boolean cond1 = true, cond2 = false;
        System.out.println("cond1 && cond2: " + (cond1 && cond2));  // false
        System.out.println("cond1 || cond2: " + (cond1 || cond2));  // true
        System.out.println("!cond1: " + (!cond1));                  // false
    }
}
```

Exercícios

1. Crie um programa que converta temperatura de Celsius para Fahrenheit
2. Escreva um programa que calcule a área de um círculo dado o raio
3. Crie um programa que receba dois números e exiba a soma, subtração, multiplicação e divisão

4. Estruturas de Controle

Condicionais: if-else

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

```
        int nota = 85;

        if (nota >= 90) {
            System.out.println("Conceito A");
        } else if (nota >= 80) {
            System.out.println("Conceito B");
        } else if (nota >= 70) {
            System.out.println("Conceito C");
        } else if (nota >= 60) {
            System.out.println("Conceito D");
        } else {
            System.out.println("Conceito F");
        }

        // Operador ternário
        String status = (nota >= 60) ? "Aprovado" : "Reprovado";
        System.out.println("Status: " + status);
    }
}
```

Switch

```
public class SwitchExemplo {
    public static void main(String[] args) {
        int dia = 3;
        String nomeDia;

        switch (dia) {
            case 1:
                nomeDia = "Domingo";
                break;
            case 2:
                nomeDia = "Segunda-feira";
                break;
            case 3:
                nomeDia = "Terça-feira";
                break;
            case 4:
                nomeDia = "Quarta-feira";
                break;
            case 5:
                nomeDia = "Quinta-feira";
                break;
            case 6:
                nomeDia = "Sexta-feira";
                break;
            case 7:
                nomeDia = "Sábado";
                break;
            default:
                nomeDia = "Dia inválido";
        }
```

```
        System.out.println("Dia " + dia + ": " + nomeDia);

        // Switch expression (Java 14+)
        nomeDia = switch (dia) {
            case 1 -> "Domingo";
            case 2 -> "Segunda";
            case 3 -> "Terça";
            case 4 -> "Quarta";
            case 5 -> "Quinta";
            case 6 -> "Sexta";
            case 7 -> "Sábado";
            default -> "Dia inválido";
        };

        System.out.println("Dia " + dia + ": " + nomeDia);
    }
}
```

Loops: for, while, do-while

```
public class Loops {
    public static void main(String[] args) {
        // Loop for
        System.out.println("Contagem crescente:");
        for (int i = 1; i <= 5; i++) {
            System.out.println("i = " + i);
        }

        // Loop while
        System.out.println("Contagem regressiva:");
        int j = 5;
        while (j > 0) {
            System.out.println("j = " + j);
            j--;
        }

        // Loop do-while
        System.out.println("Números pares até 10:");
        int k = 0;
        do {
            System.out.println("k = " + k);
            k += 2;
        } while (k <= 10);

        // For each (para arrays e coleções)
        int[] numeros = {1, 2, 3, 4, 5};
        System.out.println("Elementos do array:");
        for (int numero : numeros) {
            System.out.println("Número: " + numero);
        }
```

```
    }
}
```

Break e Continue

```
public class BreakContinue {
    public static void main(String[] args) {
        // Exemplo com break
        System.out.println("Números até encontrar o primeiro múltiplo de 7:");
        for (int i = 1; i <= 20; i++) {
            if (i % 7 == 0) {
                System.out.println("Encontrado: " + i);
                break; // Sai do loop
            }
            System.out.println("Testando: " + i);
        }

        // Exemplo com continue
        System.out.println("Números ímpares entre 1 e 10:");
        for (int i = 1; i <= 10; i++) {
            if (i % 2 == 0) {
                continue; // Pula para a próxima iteração
            }
            System.out.println("Ímpar: " + i);
        }
    }
}
```

Exercícios

1. Escreva um programa que verifique se um número é primo
2. Crie um programa que exiba a tabuada de um número informado
3. Implemente a sequência de Fibonacci até o n-ésimo termo
4. Crie um programa que classifique triângulos (equilátero, isósceles, escaleno) com base nos lados

5. Arrays e Coleções

Arrays

```
public class ArraysExemplo {
    public static void main(String[] args) {
        // Declaração e inicialização de arrays
        int[] numeros = new int[5]; // Array de 5 inteiros
        String[] frutas = {"Maçã", "Banana", "Laranja"}; // Array inicializado

        // Acessando e modificando elementos
        numeros[0] = 10;
        numeros[1] = 20;
        numeros[2] = 30;
```

```
        numeros[3] = 40;
        numeros[4] = 50;

        System.out.println("Primeiro elemento: " + numeros[0]);
        System.out.println("Tamanho do array: " + numeros.length);

        // Percorrendo arrays
        System.out.println("Elementos do array:");
        for (int i = 0; i < numeros.length; i++) {
            System.out.println("numeros[" + i + "] = " + numeros[i]);
        }

        // Array multidimensional (matriz)
        int[][] matriz = new int[3][3];
        int valor = 1;
        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[i].length; j++) {
                matriz[i][j] = valor++;
            }
        }

        System.out.println("Matriz 3x3:");
        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[i].length; j++) {
                System.out.print(matriz[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

ArrayList

```
import java.util.ArrayList;
import java.util.List;

public class ArrayListExemplo {
    public static void main(String[] args) {
        // Criando um ArrayList
        List<String> nomes = new ArrayList<>();

        // Adicionando elementos
        nomes.add("Alice");
        nomes.add("Bob");
        nomes.add("Carol");
        nomes.add("David");

        System.out.println("Lista de nomes: " + nomes);
        System.out.println("Tamanho da lista: " + nomes.size());
        System.out.println("Primeiro elemento: " + nomes.get(0));

        // Modificando elementos
```

```
        nomes.set(1, "Roberto");
        System.out.println("Após modificar: " + nomes);

        // Removendo elementos
        nomes.remove(2);
        System.out.println("Após remover: " + nomes);

        // Verificando se contém elemento
        System.out.println("Contém 'Alice'? " + nomes.contains("Alice"));

        // Percorrendo a lista
        System.out.println("Elementos da lista:");
        for (String nome : nomes) {
            System.out.println(nome);
        }

        // Convertendo para array
        String[] arrayNomes = nomes.toArray(new String[0]);
        System.out.println("Array convertido:");
        for (String nome : arrayNomes) {
            System.out.println(nome);
        }
    }
}
```

Exercícios

1. Crie um programa que some todos os elementos de um array de inteiros
2. Escreva um programa que encontre o maior e menor valor em um array
3. Implemente uma função que inverta a ordem dos elementos em um array
4. Crie um programa que remova elementos duplicados de um ArrayList

6. Introdução à Orientação a Objetos

Classes e Objetos

```
// Definição de uma classe
public class Carro {
    // Atributos (campos)
    String marca;
    String modelo;
    int ano;
    double velocidade;

    // Construtor
    public Carro(String marca, String modelo, int ano) {
        this.marca = marca;
        this.modelo = modelo;
        this.ano = ano;
        this.velocidade = 0;
    }
```

```
    // Métodos
    public void acelerar(double incremento) {
        this.velocidade += incremento;
        System.out.println("Acelerando... Velocidade atual: " + this.velocidade +
" km/h");
    }

    public void frear(double decremento) {
        if (this.velocidade - decremento >= 0) {
            this.velocidade -= decremento;
        } else {
            this.velocidade = 0;
        }
        System.out.println("Freando... Velocidade atual: " + this.velocidade + "
km/h");
    }

    public void exibirInfo() {
        System.out.println("Marca: " + this.marca);
        System.out.println("Modelo: " + this.modelo);
        System.out.println("Ano: " + this.ano);
        System.out.println("Velocidade: " + this.velocidade + " km/h");
    }
}

// Classe principal para testar
public class TesteCarro {
    public static void main(String[] args) {
        // Criando objetos (instâncias da classe)
        Carro meuCarro = new Carro("Toyota", "Corolla", 2022);
        Carro seuCarro = new Carro("Honda", "Civic", 2020);

        // Usando métodos dos objetos
        meuCarro.exibirInfo();
        meuCarro.acelerar(50);
        meuCarro.frear(20);

        System.out.println();

        seuCarro.exibirInfo();
        seuCarro.acelerar(70);
        seuCarro.frear(30);
    }
}
```

Encapsulamento e Modificadores de Acesso

```
public class ContaBancaria {
    // Atributos privados (encapsulamento)
    private String numeroConta;
    private String titular;
    private double saldo;
```

```
    // Construtor
    public ContaBancaria(String numeroConta, String titular, double saldoInicial)
{
        this.numeroConta = numeroConta;
        this.titular = titular;
        this.saldo = saldoInicial;
    }

    // Métodos públicos para acessar e modificar os atributos
    public String getNumeroConta() {
        return numeroConta;
    }

    public String getTitular() {
        return titular;
    }

    public void setTitular(String titular) {
        this.titular = titular;
    }

    public double getSaldo() {
        return saldo;
    }

    // Métodos para operações bancárias
    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
            System.out.println("Depósito de R$" + valor + " realizado. Novo saldo:
R$" + saldo);
        } else {
            System.out.println("Valor de depósito inválido.");
        }
    }

    public void sacar(double valor) {
        if (valor > 0 && valor <= saldo) {
            saldo -= valor;
            System.out.println("Saque de R$" + valor + " realizado. Novo saldo:
R$" + saldo);
        } else {
            System.out.println("Valor de saque inválido ou saldo insuficiente.");
        }
    }

    public void transferir(ContaBancaria destino, double valor) {
        if (valor > 0 && valor <= saldo) {
            this.sacar(valor);
            destino.depositar(valor);
            System.out.println("Transferência de R$" + valor + " para " +
destino.getTitular() + " realizada.");
        } else {
            System.out.println("Transferência não realizada. Verifique o valor e
```

```
saldo.");
        }
    }
}

// Testando a classe
public class TesteConta {
    public static void main(String[] args) {
        ContaBancaria conta1 = new ContaBancaria("12345-6", "João Silva", 1000.0);
        ContaBancaria conta2 = new ContaBancaria("65432-1", "Maria Santos",
500.0);

        System.out.println("Saldo inicial de " + conta1.getTitular() + ": R$" +
conta1.getSaldo());
        System.out.println("Saldo inicial de " + conta2.getTitular() + ": R$" +
conta2.getSaldo());

        conta1.depositar(200.0);
        conta1.sacar(150.0);
        conta1.transferir(conta2, 300.0);

        System.out.println("Saldo final de " + conta1.getTitular() + ": R$" +
conta1.getSaldo());
        System.out.println("Saldo final de " + conta2.getTitular() + ": R$" +
conta2.getSaldo());
    }
}
```

Exercícios

1. Crie uma classe `Pessoa` com atributos nome, idade e métodos para exibir informações
2. Implemente uma classe `Retangulo` com métodos para calcular área e perímetro
3. Desenvolva uma classe `Produto` com encapsulamento para gerenciar estoque
4. Crie uma classe `Calculadora` com métodos estáticos para operações matemáticas

7. Herança e Polimorfismo

Herança

```
// Classe base (superclasse)
public class Animal {
    protected String nome;
    protected int idade;

    public Animal(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public void emitirSom() {
        System.out.println("Animal emitindo som...");
    }
```

```
    public void dormir() {
        System.out.println(nome + " está dormindo.");
    }

    public void exibirInfo() {
        System.out.println("Nome: " + nome);
        System.out.println("Idade: " + idade + " anos");
    }
}

// Subclasse que herda de Animal
public class Cachorro extends Animal {
    private String raca;

    public Cachorro(String nome, int idade, String raca) {
        super(nome, idade); // Chama o construtor da superclasse
        this.raca = raca;
    }

    // Sobrescrita de método (override)
    @Override
    public void emitirSom() {
        System.out.println(nome + " diz: Au Au!");
    }

    // Método específico da subclasse
    public void abanarRabo() {
        System.out.println(nome + " está abanando o rabo!");
    }

    @Override
    public void exibirInfo() {
        super.exibirInfo(); // Chama o método da superclasse
        System.out.println("Raça: " + raca);
    }
}

// Outra subclasse
public class Gato extends Animal {
    private boolean temPelagemLonga;

    public Gato(String nome, int idade, boolean temPelagemLonga) {
        super(nome, idade);
        this.temPelagemLonga = temPelagemLonga;
    }

    @Override
    public void emitirSom() {
        System.out.println(nome + " diz: Miau!");
    }

    public void arranharMoveis() {
        System.out.println(nome + " está arranhando os móveis!");
    }
```

```
    @Override
    public void exibirInfo() {
        super.exibirInfo();
        System.out.println("Tem pelagem longa: " + (temPelagemLonga ? "Sim" :
"Não"));
    }
}

// Testando a herança
public class TesteAnimais {
    public static void main(String[] args) {
        Cachorro cachorro = new Cachorro("Rex", 3, "Labrador");
        Gato gato = new Gato("Mimi", 2, true);

        System.out.println("=== CACHORRO ===");
        cachorro.exibirInfo();
        cachorro.emitirSom();
        cachorro.dormir();
        cachorro.abanarRabo();

        System.out.println("\n=== GATO ===");
        gato.exibirInfo();
        gato.emitirSom();
        gato.dormir();
        gato.arranharMoveis();

        // Polimorfismo
        System.out.println("\n=== POLIMORFISMO ===");
        Animal[] animais = new Animal[2];
        animais[0] = new Cachorro("Toby", 4, "Vira-lata");
        animais[1] = new Gato("Luna", 1, false);

        for (Animal animal : animais) {
            animal.emitirSom(); // Comportamento específico de cada subclasse
            animal.dormir();    // Comportamento herdado da superclasse
            System.out.println();
        }
    }
}
```

Classes e Métodos Finais

```
// Classe final não pode ser herdada
public final class Circulo {
    private final double raio; // Constante (não pode ser modificada)
    private static final double PI = 3.14159; // Constante estática

    public Circulo(double raio) {
        this.raio = raio;
    }
```

```
    // Método final não pode ser sobrescrito
    public final double calcularArea() {
        return PI * raio * raio;
    }

    public double calcularCircunferencia() {
        return 2 * PI * raio;
    }
}

// Esta classe não compilaria:
// public class CirculoColorido extends Circulo { ... }
```

Exercícios

1. Crie uma hierarquia de classes para veículos (Carro, Moto, Caminhão) com métodos específicos
2. Implemente uma classe `Funcionario` com subclasses `Gerente` e `Desenvolvedor` com diferentes cálculos de bônus
3. Crie uma classe `Forma` com subclasses `Circulo`, `Retangulo` e `Triangulo` que implementam cálculo de área
4. Desenvolva um sistema de biblioteca com classes `Livro`, `Revista` e `Artigo` herdando de `ItemBiblioteca`

8. Tratamento de Exceções

Try-Catch e Exceções Comuns

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class TratamentoExcecoes {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Digite o primeiro número: ");
            int num1 = scanner.nextInt();

            System.out.print("Digite o segundo número: ");
            int num2 = scanner.nextInt();

            int resultado = dividir(num1, num2);
            System.out.println("Resultado da divisão: " + resultado);

        } catch (InputMismatchException e) {
            System.out.println("Erro: Valor digitado não é um número inteiro
válido.");
            scanner.next(); // Limpa o buffer do scanner

        } catch (ArithmeticException e) {
            System.out.println("Erro: Divisão por zero não é permitida.");
```

```
        } catch (Exception e) {
            System.out.println("Erro inesperado: " + e.getMessage());

        } finally {
            System.out.println("Bloco finally sempre é executado.");
            scanner.close();
        }

        System.out.println("Programa continua após o tratamento de exceções.");
    }

    public static int dividir(int a, int b) {
        return a / b;
    }
}
```

Criando Exceções Personalizadas

```
// Exceção personalizada (checked exception)
public class SaldoInsuficienteException extends Exception {
    private double saldoAtual;
    private double valorSaque;

    public SaldoInsuficienteException(double saldoAtual, double valorSaque) {
        super("Saldo insuficiente. Saldo atual: R$" + saldoAtual + ", Valor do
saque: R$" + valorSaque);
        this.saldoAtual = saldoAtual;
        this.valorSaque = valorSaque;
    }

    public double getSaldoAtual() {
        return saldoAtual;
    }

    public double getValorSaque() {
        return valorSaque;
    }
}

// Exceção personalizada (unchecked exception)
public class ValorNegativoException extends RuntimeException {
    public ValorNegativoException(String mensagem) {
        super(mensagem);
    }
}

// Classe que usa as exceções personalizadas
public class ContaExcecoes {
    private double saldo;

    public ContaExcecoes(double saldoInicial) {
```

```
        this.saldo = saldoInicial;
    }

    public void depositar(double valor) {
        if (valor < 0) {
            throw new ValorNegativoException("Valor de depósito não pode ser
negativo: " + valor);
        }
        saldo += valor;
        System.out.println("Depósito de R$" + valor + " realizado. Novo saldo: R$"
+ saldo);
    }

    public void sacar(double valor) throws SaldoInsuficienteException {
        if (valor < 0) {
            throw new ValorNegativoException("Valor de saque não pode ser
negativo: " + valor);
        }

        if (valor > saldo) {
            throw new SaldoInsuficienteException(saldo, valor);
        }

        saldo -= valor;
        System.out.println("Saque de R$" + valor + " realizado. Novo saldo: R$" +
saldo);
    }

    public double getSaldo() {
        return saldo;
    }
}

// Testando as exceções
public class TesteExcecoes {
    public static void main(String[] args) {
        ContaExcecoes conta = new ContaExcecoes(1000.0);

        try {
            conta.depositar(500.0);
            conta.sacar(200.0);
            conta.depositar(-100.0); // Lançará ValorNegativoException

        } catch (ValorNegativoException e) {
            System.out.println("Erro: " + e.getMessage());

        } catch (SaldoInsuficienteException e) {
            System.out.println("Erro: " + e.getMessage());
            System.out.println("Saldo atual: R$" + e.getSaldoAtual());
            System.out.println("Valor tentado: R$" + e.getValorSaque());

        } catch (Exception e) {
            System.out.println("Erro inesperado: " + e.getMessage());
        }
```

```
        try {
            conta.sacar(2000.0); // Lançará SaldoInsuficienteException

        } catch (SaldoInsuficienteException e) {
            System.out.println("Erro: " + e.getMessage());
            System.out.println("Saldo atual: R$" + e.getSaldoAtual());
            System.out.println("Valor tentado: R$" + e.getValorSaque());

        } catch (Exception e) {
            System.out.println("Erro inesperado: " + e.getMessage());
        }

        System.out.println("Saldo final: R$" + conta.getSaldo());
    }
}
```

Try-with-Resources

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class TryWithResources {
    public static void main(String[] args) {
        // Try-with-resources (Java 7+)
        // Os recursos são automaticamente fechados ao final do bloco
        try (BufferedReader reader = new BufferedReader(new
FileReader("arquivo.txt"))) {
            String linha;
            while ((linha = reader.readLine()) != null) {
                System.out.println(linha);
            }
        } catch (IOException e) {
            System.out.println("Erro ao ler arquivo: " + e.getMessage());
        }
        // Não é necessário chamar reader.close() - é feito automaticamente
    }
}
```

Exercícios

1. Crie um programa que solicite números ao usuário e trate exceções de entrada inválida
2. Implemente uma exceção personalizada para idade inválida (menor que 0 ou maior que 150)
3. Desenvolva um método que converta string para int com tratamento de NumberFormatException
4. Crie um sistema de reservas com exceções para datas inválidas e conflitos de reserva

9. Classes Abstratas e Interfaces

Classes Abstratas

```java
// Classe abstrata - não pode ser instanciada diretamente
public abstract class Funcionario {
    private String nome;
    private String matricula;
    private double salarioBase;

    public Funcionario(String nome, String matricula, double salarioBase) {
        this.nome = nome;
        this.matricula = matricula;
        this.salarioBase = salarioBase;
    }

    // Método abstrato - deve ser implementado pelas subclasses
    public abstract double calcularSalario();

    // Método concreto - já tem implementação
    public String getNome() {
        return nome;
    }

    public String getMatricula() {
        return matricula;
    }

    public double getSalarioBase() {
        return salarioBase;
    }

    public void setSalarioBase(double salarioBase) {
        this.salarioBase = salarioBase;
    }

    // Método concreto
    public void exibirDados() {
        System.out.println("Nome: " + nome);
        System.out.println("Matrícula: " + matricula);
        System.out.println("Salário Base: R$" + salarioBase);
        System.out.println("Salário Final: R$" + calcularSalario());
    }
}

// Subclasse concreta
public class Desenvolvedor extends Funcionario {
    private int horasExtras;
    private double valorHoraExtra;

    public Desenvolvedor(String nome, String matricula, double salarioBase,
                         int horasExtras, double valorHoraExtra) {
        super(nome, matricula, salarioBase);
        this.horasExtras = horasExtras;
        this.valorHoraExtra = valorHoraExtra;
    }

    // Implementação do método abstrato
```

```
    @Override
    public double calcularSalario() {
        return getSalarioBase() + (horasExtras * valorHoraExtra);
    }

    public int getHorasExtras() {
        return horasExtras;
    }

    public void setHorasExtras(int horasExtras) {
        this.horasExtras = horasExtras;
    }

    public double getValorHoraExtra() {
        return valorHoraExtra;
    }

    public void setValorHoraExtra(double valorHoraExtra) {
        this.valorHoraExtra = valorHoraExtra;
    }
}

// Outra subclasse concreta
public class Gerente extends Funcionario {
    private double bonus;

    public Gerente(String nome, String matricula, double salarioBase, double
bonus) {
        super(nome, matricula, salarioBase);
        this.bonus = bonus;
    }

    // Implementação do método abstrato
    @Override
    public double calcularSalario() {
        return getSalarioBase() + bonus;
    }

    public double getBonus() {
        return bonus;
    }

    public void setBonus(double bonus) {
        this.bonus = bonus;
    }
}

// Testando as classes
public class TesteFuncionarios {
    public static void main(String[] args) {
        // Não é possível instanciar uma classe abstrata:
        // Funcionario f = new Funcionario("João", "123", 2000); // ERRO!

        Funcionario dev = new Desenvolvedor("Maria", "DEV001", 3000, 10, 50);
        Funcionario gerente = new Gerente("Carlos", "GER001", 5000, 1000);
```

```
        System.out.println("=== DESENVOLVEDOR ===");
        dev.exibirDados();

        System.out.println("\n=== GERENTE ===");
        gerente.exibirDados();

        // Polimorfismo com classes abstratas
        Funcionario[] funcionarios = new Funcionario[2];
        funcionarios[0] = dev;
        funcionarios[1] = gerente;

        System.out.println("\n=== FOLHA DE PAGAMENTO ===");
        double totalFolha = 0;
        for (Funcionario f : funcionarios) {
            f.exibirDados();
            totalFolha += f.calcularSalario();
            System.out.println();
        }
        System.out.println("Total da fol
```