* **Exercício 1 (Básico) - Herança**

**Enunciado:**

**Crie uma classe chamada Animal que tenha um atributo nome e um método fazerSom(), que imprime "Som genérico de animal". Depois, crie uma classe Cachorro que herda de Animal e sobrescreve o método fazerSom() para imprimir "O cachorro late". No método main(), crie um objeto do tipo Cachorro e chame o método fazerSom().**

**public class Animal {**

**String nome;**

**void fazerSom() {**

**System.out.println("Som genérico de animal");**

**}**

**}**

**public class Cachorro extends Animal {**

**@Override**

**void fazerSom() {**

**System.out.println("O cachorro late");**

**}**

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

**Cachorro c = new Cachorro();**

**c.fazerSom();**

**}**

**}**

**Exercício 2 (Básico) - Polimorfismo**

**Enunciado:**

**Crie uma classe Veiculo com um método mover(), que imprime "O veículo está se movendo". Em seguida, crie duas classes Carro e Bicicleta, ambas herdando de Veiculo. Sobrescreva o método mover() para que Carro imprima "O carro está se movendo rapidamente" e Bicicleta imprima "A bicicleta está se movendo lentamente". No main(), crie um vetor de Veiculo com um Carro e uma Bicicleta e percorra o vetor chamando mover() para cada um.**

**public class Veiculo {**

**void mover() {**

**System.out.println("O veículo está se movendo");**

**}**

**}**

**public class Carro extends Veiculo {**

**@Override**

**void mover() {**

**System.out.println("O carro está se movendo rapidamente");**

**}**

**}**

**public class Bicicleta extends Veiculo {**

**@Override**

**void mover() {**

**System.out.println("A bicicleta está se movendo lentamente");**

**}**

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

**Veiculo[] veiculos = { new Carro(), new Bicicleta() };**

**for (Veiculo v : veiculos) {**

**v.mover();**

**}**

**}**

**}**

**Exercício 3 (Médio) - Herança e Polimorfismo Aplicados**

**Enunciado:**

Crie uma hierarquia de classes para representar funcionários de uma empresa. A classe Funcionario deve ter os atributos nome e salarioBase, além de um método calcularSalario(), que retorna o salarioBase.

Crie duas classes derivadas:

1. Gerente: recebe um bonus adicional ao salário.
2. Desenvolvedor: recebe um aumento de 10% sobre o salarioBase.

No main(), crie um Funcionario, um Gerente e um Desenvolvedor, defina seus salários e exiba os valores calculados pelo método calcularSalario().

public class Funcionario {

String nome;

double salarioBase;

double calcularSalario() {

return salarioBase;

}

}

public class Gerente extends Funcionario {

double bonus;

@Override

double calcularSalario() {

return salarioBase + bonus;

}

}

public class Desenvolvedor extends Funcionario {

@Override

double calcularSalario() {

return salarioBase \* 1.1;

}

public static void main(String[] args) {

Funcionario f = new Funcionario();

f.nome = "João";

f.salarioBase = 3000;

Gerente g = new Gerente();

g.nome = "Maria";

g.salarioBase = 5000;

g.bonus = 1500;

Desenvolvedor d = new Desenvolvedor();

d.nome = "Carlos";

d.salarioBase = 4000;

System.out.println(f.nome + ": " + f.calcularSalario());

System.out.println(g.nome + ": " + g.calcularSalario());

System.out.println(d.nome + ": " + d.calcularSalario());

}

}

**Exercício 4 (Médio) - Herança e Polimorfismo Aplicados**

**Enunciado:**

Crie um sistema que represente diferentes formas de pagamento usando herança e polimorfismo.

1. Crie uma classe base chamada Pagamento com um método realizarPagamento() que imprime "Processando pagamento...".
2. Crie duas classes derivadas:
   * PagamentoCartao que sobrescreve realizarPagamento() para imprimir "Pagamento com cartão aprovado!".
   * PagamentoDinheiro que sobrescreve realizarPagamento() para imprimir "Pagamento em dinheiro recebido!".
3. No main(), crie um **vetor de Pagamento** que contenha objetos de PagamentoCartao e PagamentoDinheiro e use um laço para chamar realizarPagamento() para cada um.

public class Pagamento {

void realizarPagamento() {

System.out.println("Processando pagamento...");

}

}

public class PagamentoCartao extends Pagamento {

@Override

void realizarPagamento() {

System.out.println("Pagamento com cartão aprovado!");

}

}

public class PagamentoDinheiro extends Pagamento {

@Override

void realizarPagamento() {

System.out.println("Pagamento em dinheiro recebido!");

}

public static void main(String[] args) {

Pagamento[] pagamentos = { new PagamentoCartao(), new PagamentoDinheiro() };

for (Pagamento p : pagamentos) {

p.realizarPagamento();

}

}

}