O trabalho é um sistema de gerenciamento de funcionários. Um funcionário base e duas subclasses que representam o gerente e o operário. Os funcionários terão ações diferentes para aplicar o polimorfismo. Também foi usado encapsulamento para proteger as informações de cada funcionário.

Este sistema simula uma gestão de funcionários, onde temos três tipos de funcionários:

A classe Funcionario.java é a base para as outras duas classes. Ela contém atributos e métodos comuns, como nome e salário, e um método abstrato realizarAtividade(), que será implementado pelas subclasses.

Funcionario:

```
private String nome;  4 usages
private double salario;  5 usages

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

public String getNome() {  no usages
    return nome;
}

public void setNome(String nome) {  no usages
    this.nome = nome;
}

public double getSalario() {  no usages
    return salario;
}

public void setSalario(double salario) {  no usages
    this.salario = salario;
}

public abstract void realizarAtividade();  no usages

public double calcularPagamento() {  no usages  2 over
    return salario;
```

A classe Gerente.java herda de Funcionario, mas adiciona um bônus ao pagamento e implementa a atividade específica para um gerente.

Gerente:

```
private double bonus;  3 usages

public Gerente(String nome, double salario, double bonus) {  no usag
    super(nome, salario);
    this.bonus = bonus;
}

@Override  no usages
public void realizarAtividade() {
    System.out.println(getNome() + " está gerenciando a equipe.");
}

@Override  no usages
public double calcularPagamento() {
    return getSalario() + bonus;
}

@Override
public String toString() {
    return super.toString() + ", Bônus: R$" + bonus;
```

A classe Operario.java também herda de Funcionario, mas inclui um atributo para as horas extras e calcula o pagamento com base nisso.

Funcionario:

```java
private double horasExtras;  3 usages

public Operario(String nome, double salario, double horasExtras) {  no usages
    super(nome, salario);
    this.horasExtras = horasExtras;
}

@Override  no usages
public void realizarAtividade() {
    System.out.println(getNome() + " está realizando trabalho operacional.");
}

@Override  no usages
public double calcularPagamento() {
    return getSalario() + (horasExtras * 20); // R$20 por hora extra
}

@Override
public String toString() {
    return super.toString() + ", Horas extras: " + horasExtras;
```

E a classe Main cria objetos, testa e executa o programa.

Main:

```java
public static void main(String[] args) {
    // Criando instâncias dos funcionários
    Funcionario gerente = new Gerente( nome: "Alice",  salario: 5000,  bonus: 1000);
    Funcionario operario = new Operario( nome: "Bob",  salario: 2000,  horasExtras: 30);

    // Imprimindo os dados dos funcionários
    System.out.println(gerente);
    System.out.println(operario);

    // Realizando as atividades
    gerente.realizarAtividade();
    operario.realizarAtividade();

    // Calculando o pagamento
    System.out.println("\nPagamento do Gerente: R$" + gerente.calcularPagamento());
    System.out.println("Pagamento do Operário: R$" + operario.calcularPagamento());
```

Herança:

As classes Gerente e Operario herdam da classe Funcionario.

A classe funcionário seria como uma base, e as subclasses herdam as características básicas e expandem conforme necessário. Exemplo: Gerente e Operario podem receber pagamento extra, mas um como bônus e o outro como horas extras.

Polimorfismo:

Os métodos realizarAtividade() e calcularPagamento() são exemplos de polimorfismo porque dependem do tipo do funcionário.

Exemplo: Uma lista de funcionários, uns são gerentes e outros operários, ambos são podem executar o mesmo método, mas podendo ter resultados diferentes.

Encapsulamento:

Os dados privados de Funcionario(nome e salário) são acessados por meio de métodos públicos.

Não seria bom deixar qualquer um ver o atributo salário, então usei um getSalario() e um setSalario(), que equivaleriam a pedir permissão para ver ou alterar uma informação do atributo salário.

Abstração:

A classe Funcionario é abstrata não podendo ser instanciada diretamente. Ela define o método abstrato realizarAtividade(), o que obriga as subclasses a fornecerem sua própria implementação.

A classe funcionário seria como uma ideia genérica, e as subclasses detalham a ela de acordo com a sua área.

Victor do Espirito Santo Farias 202403557908