Relatorio WePayU

Tatiane Monteiro de Araujo

O sistema de pagamentos WePayU é uma aplicação projetada para simplificar e automatizar a administração de funcionários em uma empresa. Ele gerencia detalhes como horários, salários, comissões, filiação sindical e métodos de pagamento dos funcionários.

O projeto adota uma abordagem de design que se baseia na separação de classes para diferentes funcionalidades do sistema. Isso permite uma organização do código e facilita a manutenção e expansão futura da aplicação.

As principais características do sistema incluem:

- Gerenciamento de Informações dos Empregados:
- Automação de Cálculos de Folha de Pagamento:
- Funcionalidades de Persistência de Dados:

Classes implementadas e suas funcionalidades:

A classe Sistema é o núcleo do programa, responsável por coordenar todas as operações do sistema. Ela recebe as solicitações do usuário por meio da classe Facade e realiza chamadas aos métodos correspondentes. Sua função é garantir o funcionamento adequado do sistema, tratando entradas, criando instâncias dos objetos principais e determinando as ações a serem tomadas com base nas solicitações do usuário.

O Sistema mantém uma lista completa de todos os empregados do sistema, e uma lista com aqueles que são membros do único sindicato existente. Ele é encarregado de criar, modificar e remover empregados, além de gerenciar atributos específicos de cada empregado, como vendas, horas trabalhadas e taxa do sindicato. A classe Sistema assegura que os dados recebidos sejam armazenados corretamente nos funcionários e em seus atributos específicos.

Além disso, a classe Sistema é responsável por fornecer as informações solicitadas pelo usuário, como os atributos dos funcionários, quando solicitado. Sua função é

garantir a integridade e o bom funcionamento do sistema, atendendo às solicitações do usuário.

A classe Empregado é uma superclasse. O conceito de superclasse é uma estrutura fundamental na programação orientada a objetos por que serve como um molde a partir do qual as subclasses podem ser derivadas.

Ela é responsável por estruturar o modelo dos diferentes tipos de empregados e definir alguns de seus comportamentos genéricos. Além disso, promove a reutilização de código para as classes filhas, que herdam suas características principais, como nome, ID e endereço, e permitem a sobrescrita de métodos que possuem características distintas em cada subclasses, como é o caso do método getTipo();

Dessa forma temos a implementação do conceito de herança que é um dos princípios fundamentais da orientação a objetos, pois permite a reutilização de código em suas subclasses. Isso evita a duplicação de código e promove a modularidade e a manutenção do código. Além disso, permite a generalização de comportamentos comuns a todos os empregados e o polimorfismo, que possibilita que objetos de diferentes classes sejam tratados de maneira uniforme.

Portanto a classe empregado mantém todos os métodos que são compartilhados entre os diferentes tipos de empregados.

Além disso, dentro da classe Empregado, temos instâncias das classes MetodoDePagamento e MembroSindicato.

Cada uma das subclasses representa um tipo específico de empregado, com suas próprias características e comportamentos distintos. Essa abordagem ajuda a organizar o código de forma mais eficiente e a facilitar a manutenção e expansão do sistema.

Temos as seguintes classes para os diferentes tipos de empregados:

- Classe Horista: Esta classe, além dos métodos herdados da superclasse "Empregados", mantém atributos específicos dos empregados horistas. Isso inclui a quantidade de horas trabalhadas por dia e as datas em que ele trabalhou. As horas são armazenadas em uma lista junto com as respectivas datas, representando as horas trabalhadas em cada dia específico.
- Classe Assalariado: Esta classe, além dos métodos herdados da superclasse "Empregados", mantém atributos específicos do tipo e da agenda de pagamento do funcionário.

- Classe Comissionado: Esta classe mantém os atributos da superclasse "Empregados" e inclui atributos específicos, como a comissão e as vendas realizadas. A classe "Venda" representa uma venda específica e mantém informações como a data e o valor da venda. Embora atualmente a classe "Venda" contenha apenas esses dois atributos, sua estrutura de classe permite uma melhor expansibilidade do sistema, como adicionar informações sobre os produtos vendidos. A classe "Comissionado" mantém uma lista de instâncias de vendas, representando todas as vendas feitas por cada empregado comissionado.

A classe **MetodoPagamento** é responsável por armazenar um tipo de pagamento para cada empregado. Quando o tipo de pagamento é feito através de um banco, a classe armazena uma instância da classe **Banco**, contendo todas as informações pertinentes do banco, como número da conta, nome e agência. Cada empregado mantém uma instância diferente do método de pagamento, garantindo assim a unicidade desse método para cada funcionário.

A importância da unicidade do método de pagamento fica evidente quando lidamos com informações sensíveis, como o número da conta bancária. Cada empregado deve ter seu próprio método de pagamento para garantir que suas informações bancárias permaneçam seguras e não sejam acessíveis a outros funcionários.

A classe **AgendaPagamentos**, desempenha um papel fundamental no sistema de pagamento. Ela é responsável por receber os dados do tipo de agenda de pagamento, armazenar os tipos e criar um calendário que mantém as datas de pagamento de todos os funcionários. Essa classe calcula todas as datas de pagamento de um funcionário durante um período de um ano e as armazena no calendário. O calendário, por sua vez, contém as datas de pagamento de todos os empregados do sistema.

A classe **CalculaFolha** é responsável por retornar a folha de pagamento em uma data específica. Ela utiliza a instância da classe **AgendaPagamentos**, recebida do sistema, para encontrar a lista de funcionários que devem ser pagos na data especificada. Em seguida, calcula o total da folha de pagamento para esse dia específico. Além disso, essa classe é responsável por formatar e escrever a folha de pagamento.

O sistema possui ainda duas classes que lidam com arquivos. A classe **LerDados** que é responsável por ler os dados dos funcionários que foram armazenados em

arquivos (que simulam um banco de dados). E a classe **SalvarDados** armazena os dados de todos os funcionários nos arquivos. Ambas possuem relação com a classe **Sistema** pois necessitam da lista de empregados armazenados em Sistema.

Interações entre as classes.

- **Horista** (subclasse de **Empregado**):
    - Herança: A classe Horista herda características e comportamentos da classe Empregado.
- **Assalariado** (subclasse de **Empregado**):
    - Herança: A classe Assalariado também herda características e comportamentos da classe Empregado.
- **Comissionado** (subclasse de **Empregado**):
    - Herança: A classe Comissionado também herda características e comportamentos da classe Empregado.
- **Banco** depende de **MetodoPagamento**:
    - Dependência: A classe Banco depende da existência da classe MetodoPagamento para ser instanciada ou para realizar suas operações.
- **Sistema** agrega **Empregado**:
    - Agregação: O Sistema possui uma lista de Empregados.
- **Empregado** agrega **MétodoDePagamento**:
    - Agregação: Cada empregado possui um método de pagamento específico.
- **Empregado** agrega **MembroSindicato**:
    - Agregação: A filiação ao sindicato é mantida para cada empregado.
- **Comissionado** agrega **Venda**:
    - Agregação: Se o empregado for do tipo comissionado, ele pode ter várias vendas associadas a ele.
- **CalculoFolha** usa **AgendaPagamento**:
    - A classe CalculoFolha não está diretamente associada à classe AgendaPagamento. Mas ela usa um de seus métodos.
- **SalvarDados** e **LerDados** associam-se com **Sistema**:
    - Associação: Às classes SalvarDados e LerDados estão associadas à classe Sistema.