Relatório Milestone 2 - P2

WePayU

Aluno: José Matheus Santana Alves (Zé)

Descrição Geral da Arquitetura

O sistema proposto é um sistema de folha de pagamento de uma empresa. O cálculo da folha de pagamento apresenta diferenças de acordo com o tipo de empregado a receber do seu trabalho e algumas características atreladas ao empregado. Um empregado pode ser horista, comissionado ou assalariado, e portanto a arquitetura do sistema é desenhada em torno dessa dinâmica. As principais características (atributos) necessários para os cálculos serão guardados dentro da classe Empregado, que serve como classe mãe para as classes EmpregadoHorista, EmpregadoComissionado e EmpregadoAssalariado. A partir dessa herança é possível designar métodos e atributos diferentes, para diferentes tipos de empregados, possibilitando que o cálculo da folha de pagamento considere as individualidades da situação atual de cada empregado.

Para que seja possível criar, manipular e analisar os empregados, é implementado uma business logic que possibilita ações que permitem as operações do CRUD (Create, Read, Update & Delete), além de operações específicas para cada tipo de empregado para que sejam registrados no sistema as informações de seu trabalho, como horas trabalhadas, que serão levadas em consideração durante o cálculo da folha de pagamento.

A business logic do sistema se sustenta em diversas classes utilitárias criadas para dar suporte às operações necessárias para funcionamento do sistema, como armazenamento e manipulação dos objetos das classes Empregado, persistência dos dados, tratamento de entrada e tratamento de exceções. Esses artefatos controlam o fluxo do sistema, mantém seu estado e garantem a entrega correta das informações.

Principais componentes e suas interações

Como supracitado, o sistema funciona em torno da classe de Empregado, que guarda as principais informações para entrega da folha de pagamento e garante, com a herança, o tratamento individualizado para tipos de empregados.

Empregado

Guarda os atributos:

- ID;
- Nome;
- Endereço;
- Tipo de Empregado;

- Salário;
- Se é sindicalizado ou não;
- Informações sindicais; e
- Método de pagamento utilizado.

E possui os métodos responsáveis para acesso e registro desses atributos (getters e setters). Essa classe serve como classe mãe para as classes EmpregadoHorista, EmpregadoComissionado e EmpregadoAssalariado.

EmpregadoHorista

Guarda os mesmos atributos de Empregado e guarda uma lista de cartões de pontos, representados pela classe CartaoDePonto, que mantém as informações da data de lançamento do cartão e a quantidade de horas trabalhadas. Possui getters e setters, além de método para adicionar novo cartão de ponto à lista e métodos para contabilizar a quantidade de horas trabalhadas dentro de determinado intervalo de tempo.

EmpregadoComissionado

Guarda os atributos de Empregado e guarda uma taxa de comissão, além de uma lista de vendas realizadas, representadas pela classe ResultadoDeVenda, que armazena a data e o valor de uma venda. Além de getters e setters, possui métodos para adicionar uma nova venda e para contabilizar o valor das vendas realizadas dentro de um determinado período.

EmpregadoAssalariado

Não guarda nenhum atributo além dos guardados em Empregado. Existe para que a diferenciação entre empregados seja possível de maneira mais dinâmica.

MetodoPagamento

Classe utilizada para guardar as informações relacionadas aos métodos de pagamento utilizados em Empregado. É classe mãe de Banco, Correios e EmMaos, que representam os métodos de pagamento diferentes possíveis.

MembroSindicato

Guarda as informações referentes ao sindicato dos empregados. Guarda:

- Identificador (ID) de membro;
- Taxa sindical; e
- Lista de taxas de serviço;
    - Taxa de serviço representada pela classe TaxaServico que guarda data e valor do serviço.

Possui getters e setters, métodos para adicionar uma nova taxa de serviço e para contabilizar o valor das taxas dentro de determinado período.

BusinessLogic

Classe que realiza toda a lógica de negócio do sistema. Ela fornece identificadores aos empregados, se utilizada das classes Persistence e Xstream, utilizadas para persistência dos dados.
Possui os métodos para:

- zerar o sistema;
- encerrar o sistema;
- criar novo empregado;
- consultar atributo de empregado;
- alterar atributo de empregado;
- consultar empregado pelo nome;
- remover empregado do sistema;
- lançar novo cartão de pontos;
- lançar nova venda;
- lançar nova taxa de serviço;
- consultar vendas;
- consultar horas dos cartões de pontos;
- consultar taxas de serviço;
- calcular o total da folha de pagamento.

TratamentoEntrada

Classe utilitária que auxilia no tratamento de entradas no sistema e no tratamento de exceções durante o funcionamento do programa.

ListaEmpregados

Classe utilitária que auxilia na manipulação dos empregados armazenados no sistema.

Atualizações Milestone 2

Undo e Redo

Agora é possível desfazer e refazer ações realizadas pelo cliente durante o funcionamento do sistema. Para a implementação dessas funcionalidades, seguiu-se o padrão de projeto memento. Com o auxílio de duas pilhas, é possível armazenar os estados do sistema de maneira a possibilitar o retrocesso ao estado imediatamente anterior quando o comando é solicitado.

Antes de qualquer ação capaz de mudar o estado do sistema, ele é salvo na pilha do Undo, e toda vez que o método Undo é chamado, o estado desfeito é armazenado na pilha do Redo para ser refeito.

Agenda de Pagamentos

Para permitir mais flexibilidade na gestão de pagamentos, agora é possível armazenar e manipular diferentes agendas de pagamentos, podendo alterar a periodicidade em que é realizada a entrega do pagamento ao empregado. Para implementar tal funcionalidade, foram necessários alguns ajustes:

- Agora o Empregado armazena a agendaPagamento, atributo da classe Empregado, representado pela classe AgendaDePagamento
- A classe AgendaDePagamento mantém o regime de pagamento seguido pelo Empregado e realiza o tratamento de sua entrada
- Como é possível ao usuário criar agendas de pagamento personalizadas para diferentes empregados, foi criado também uma classe para armazenar as Agendas de pagamentos, que é mantida através da persistência utilizando-se de serialização com arquivos .xml.
- Agora que o pagamento é realizado de maneira personalizada para todos os empregados, de acordo com sua agenda de pagamentos, é criada a classe ProcessaPagamento, que tem a especialidade de realizar o cálculo do pagamento individual de um empregado, independente de seu tipo.

Refatorando o projeto

O código também foi refatorado, entre a milestone 1 e a milestone 2, de forma a realizar uma limpeza, numa tentativa de eliminar o máximo de code smells e otimizar sua legibilidade. Agora a lógica do sistema está mais isolada, os métodos e processos de conversão estão encapsulados dentro das classes utilitárias, permitindo a manipulação limpa de suas funcionalidades.