Projeto da disciplina Laboratório de Programação II

31 de Julho de 2018

O que foi realizado

Caso 1:
Foi criada a classe Item, uma classe abstrata que representa todos os itens do sistema. E então, criada mais três classes herdando da classe Item: uma que representa os itens com quantidade fixa, uma que representa os itens por quilo e uma que representa os itens por unidade. Foi escolhido esse design, pois os itens possuem atributos e métodos em comum, por isso achamos melhor abstrair a classe Item. Criamos uma classe ControllerItem que é responsável por armazenar, os itens cadastrados no sistema os métodos para criar, atualizar, exibir e deletar itens do sistema. Os itens são armazenados em um TreeMap, pois cada item tem tem um id para ser identificado. Além disso, foram criados os testes de unidades para as classes com lógicas testáveis.

Caso 2:
Foram criadas classes que representam controladores para fazer ordenação dos itens cadastrados. Foi criado a classe OrdenaItensPorNome para ordenar itens pelo nome, foi criada a classe OrdenaPorCategoria para ordenar os itens pela categoria, foi criada a classe OrdenaItensPorMenorPreco para ordenar itens pelo menor preço e foram criados os métodos no controlador de itens para realizar essas funcionalidades. E também foi feito o método para buscar um item a partir do seu nome(fazendo a busca nos itens cadastrados) e a resposta da busca será em ordem alfabética. Além disso, foram criados os testes de unidades para esses métodos.

Caso 3:
Foi criada a classe Lista De Compras que representa uma lista de compras, que tem um TreeMap que associa um Item a uma classe Compra. Essa classe Compra guarda o Item e a quantidade que pretende ser comprada naquela lista de compras que ele está contido e essa classe agora também é responsável por atualizar a quantidade de um item em uma lista de compras. Foi criado um controlador para lista de compras que possui um TreeMap que associa o descritor da lista à ListaDeCompras, e é responsável por criar uma lista, adicionar itens a essa lista, atualizar a quantidade desse item em uma data lista, deletar um item de uma lista e finalizar uma lista de compras. Na exibição dos itens que estão em uma lista de compras são ordenados em duas camadas, pela categoria e em caso de empate pelo nome, para resolver o problema fizermos a ordenação pelo nome e então pela categoria já que o Java não anula a ordenação já feita antes. A classe ControllerDeListasDeCompras tem acesso a todos os itens cadastrados, pois recebe como parâmetro no construtor o controlador de itens.

Caso 4:
Foram criados métodos para pesquisar listas de compras pelo descritor, pela data e por Item, o método pesquisaListaDeCompras, pesquisa e retorna o descritor da lista de compras com o descritor passado como parâmetro, já o método pesquisaListasDeComprasPorData pesquisa e retorna os descritores de todas as listas de compras criadas na data passada como parâmetro e o método pesquisaListasDeComprasPorItem que pesquisa e retorna a data de criação e o descritor de todas as listas de compras que possuem o item passado como parâmetro.

Caso 5:
Foram implementados os métodos no ControllerListaDeCompras para a geração automática das listas de compra. Sendo divididos por estratégias, sendo elas: geraAutomaticaUltimaLista, que apenas copia a última lista adicionada ao sistema, geraAutomaticaItem, que copia a última lista em que o item passado como parâmetro está presente, nessa estratégia percorremos todas as listas de compras cadastradas verificando se possuía tal item, se sim guardamos o descritor daquela lista para caso ela seja a última, conseguiremos gerar uma lista a partir desta que foi encontrada. E geraAutomaticaItensMaisPresentes, que gera uma nova lista contendo os itens mais cadastrados nas listas de compra do sistema, para essa estratégia, percorremos os itens cadastrados nas listas e íamos vendo a quantidade dele na lista de compras e em quantas listas ele estava presente, caso fosse em mais da metade, retornamos a quantidade média que ele apareceu nas listas de compras(arredondando para cima).

Caso 6:
Foi criada a classe Supermercado, para o melhor gerenciamento dos estabelecimentos de compra que guardar o nome do supermercado e um ArrayList de itens que foram cadastrados nesse supermercado. No método sugereMelhorEstabelecimento percorremos os itens de uma dada lista e verificamos os preços nos supermercados onde este item existe para saber qual é o mais barato. E então ordenamos os Supermercados de acordo com o valor total. E a partir da posição passada como parâmetro é retornado o toString do respectivo Supermecado. A partir do mesmo método também é possível imprimir o toString de um item que está dentro de um Supermercado.

Caso 7:
Foram criados os métodos de salvarDados e carregarDados nos respectivos Controllers para que as informações dos itens e das listas de compra presentes no software permaneçam ativos mesmo após o mesmo ser finalizado e reiniciado. Para salvar os dados(TreeMap que associa um id ao item e o id que é usado para saber a posição do item a ser cadastrado), resolvemos usar a classe ObjectOutputStream, na qual recebe um objeto do tipo FileOutputStream(que recebe o caminho para o arquivo onde será salvo) e então usamos o método writeObject para escrever determinado objeto para o arquivo desejado. De forma análoga irá ser feita no meu controllerListaDeCompras para salvar as listas cadastradas no meu sistema. Isso será feito toda vez que o usuário chamar o método fechaSistema. E o para carregar dados usamos a classe ObjectInputStream, na qual recebe um objeto do tipo FileInputStream(que recebe o caminho para o arquivo onde está os dados

a serem carregados) e então usamos o método readObject para ler o objeto que está no arquivo. Isso será feito ao chamar o método iniciaSistema.

Links úteis

Diagrama:
https://drive.google.com/file/d/1nEb6keHoAH5x4pthC9IGggCncBpPKWAk/view?usp=sharing
GitHub:
https://github.com/IgorSilveira7/Projeto_LP2/