Relatório Projeto LP2

eDoe.com

Caso 1: O caso um pede para que seja criado um sistema que permita o CRUD de usuarios receptores e doadores. Optamos por criar um classe abstrata chamada Usuário que irá conter os atributos nome, email, celular, id e classe todos em String. Depois criamos duas classes que herdam da classe Usuario, a primeira denominada Doador e a outra Receptor. Essas classes irão possuir os mesmo atributos de Usuario se diferenciando no método toString(). Por fim criamos uma classe chamada ControllerUsuarios que possui uma HashMap de usuários como atributo e tem como responsabilidade receber um arquivo de usuários receptores que é lido fazendo com que estes usuários sejam automaticamente criados. Após isso a classe irá ter métodos com o adicionaDoador() que cadastrar um usuário doador no mapa de Usuário entre outros métodos que tem a função de atualizar Usuário, pesquisar usuário(pelo nome ou id) e remover usuário.

Caso 2: O caso dois pede para que usuários doadores possam inserir itens a serem doados. E para isso resolvemos criar uma classe chamada Item que possui como atributos o id do item, uma lista de tags que irá conter as características de um item, a quantidade, uma String contendo a descrição do item e um atributo Doador já que todo item tem um doador(Composição). Após isso resolvemos criar a classe ControllerItens que vai ter como responsabilidade adicionar, atualizar, remover e pesquisar itens a partir da agregação do controller de usuario e do uso dos metodos getDoador() e getReceptor() para alocar cada um dos itens ao usuário correto. A classe também irá conter um HashSet de descritores de cada item que não irá permitir cadastrar um descritor de Item já existente.

Caso 3: Para o caso três criamos 3 métodos adicionais na classe ControllerItens, o primeiro chamado listaDescritorDeItensParaDoacao() que permite uma listagem de todos os descritores de itens cadastrado no sistema, ordenado em ordem alfabética

pela descrição do item, para isso criamos a classe DescricaoItemOrdemAlfabetica() que implementa a interface Comparator. O segundo método chamamos de listaItensParaDoacao() que permite a listagem de todos os itens inseridos no sistema, ordenada pela quantidade do item no sistema, para isso criamos outra classe que implementa a interface Comparator e a chamamos de OrdemQuantidadeDeItens(). Por fim, o último método pesquisa ItemParaDoacaoPorDescricao(String descricao) tem como função listar todos os itens relacionados a uma dada string de pesquisa utilizando a classe ItemOrdemAlfabetica() que também implementa uma interface comparator.

Caso 4: O caso quatro pedisse para que os usuários do sistema que são receptores indiquem os itens que estão precisando receber. Para isso criamos uma classe chamada ItemNecessario que possui quase os mesmos atributos e métodos da classe Item mais se diferencia ao estar associada a um Usuário do tipo receptor. Modificamos a classe ControllerItens para receber os métodos de itens necessários, com isso criamos cruds como adicionaItemNecessario(), listaItensNecessarios(), atualizaItemNecessario() e removeItemNecessario() que se comunicam com o ControllerUsuario e usam o método getReceptor() para realizar a adição, listagem, atualização ou remoção de itens necessários no mapa de itens necessários localizados na classe Receptor.

Caso 5: O caso cinco pede para implementarmos a funcionalidade mais importante do sistema: encontrar casamentos (matches) entre itens a serem doados e itens necessários. Para isso criamos atributos para acumular pontos de match nas classes Item e ItemNecessário, depois criamos 3 métodos adicionais na classe ControllerItens o primeiro chamado pecorreItensNecessariosReceptor() que é responsável por olha o descritor de item e as tags dos itens a serem doados para fazer o casamento com o item necessário e assim incrementar pontos de match. O segundo método se chama comparaTagsESoma() que é responsável por comparar tags, se elas forem iguais e estiverem na mesma posição somam 10 pontos, mas se forem iguais em posições diferentes somam 5 pontos. O último é método mais importante do projeto ele se chama match() e será ele quem irá realizar o

casamento entre itens, primeiro adicionando os matches numa lista e depois irá ordená-los de acordo com sua pontuação de match, por fim irá retornar uma String contendo todos os itens casados pela função.

Caso 6: Conhecendo os possíveis casamentos entre itens necessários e para doação, um usuário receptor pode solicitar uma doação. Com isso criamos três classes adicionais no projeto a primeira chamada Doação que irá representar uma doação entre um usuário doador e um usuário receptor através dos seus itens necessários, ela irá conter a data que uma doação foi realizada, um item doado, um item necessário, getters, setters e um toString() que irá retornar as informações de uma doação. A segunda classe se chama ControllerDoacao que controla as doações entre usuários, possui o método RealizaDoacao() em que o usuário deve indicar o identificador do item necessário e do item a ser doado. E o sistema deve validar o pedido de doação olhando se os descritores de itens são mesmo iguais. Caso o pedido seja validado, o sistema deve atualizar a quantidade de itens a serem doados e de itens necessários dos itens envolvidos nesta doação. Se uma dessas quantidades cair para zero, o item específico (para doação ou necessário) é removido do sistema. O segundo método presente na classe é o listarDoacoes() que lista o histórico de doações pela ordem em que as mesmas foram realizadas (da mais antiga para a mais nova). E caso as datas sejam iguais ele lista pela ordem alfabética das descrições dos itens doados. Para isso criamos a terceira e última classe do caso, chamada OrdenarDoacaoPelaData que implementa uma interface comparator que ordena a lista de doações pela data no método listaDoacoes().

Caso 7: O ultimo caso pedisse para que o edoe.com armazene em disco todos os dados necessários para manter o seu estado mesmo que o programa seja fechado. Para isso, em todas as classes presentes no package controllers criamos os métodos salvar() e carregarDados() na qual esse primeiro irá usar o ObjectOutputStream e o writeObject para salvar, e o segundo metodo irá usar o ObjectInputStream e o readObject para carregar os dados salvos. Nas classes Usuario, Item, ItemNecessário e Doação implementamos a interface Serializable. E por fim na facade os métodos finalizaSistema() e iniciaSistema() responsáveis,

respectivamente, por fazer o salvamento dos dados no arquivo e o carregamento dos dados salvos no arquivo. Com isso ao se usar o programa e adicionar usuários, itens a serem doados, itens necessários, realizar doações, etc. tudo isso poderá ser visto caso o programa seja fechado e aberto novamente.

Grupo:
Almir Crispiniano
Caroliny Silva
Davidson Guedes
Matheus Gusmão