Relatório de Projeto LP2

Design Geral:

Nosso design foi pensado de modo a ter um melhor relacionamento entre as partes do sistema. Basicamente foram criadas duas camadas que facilitam a organização do sistema, sendo essas, uma camada que coordena o funcionamento do sistema em geral e outra camada que faz o funcionamento do sistema de empréstimos.
Para a atribuição de cartões fidelidade foi usado o padrão strategy, usando interfaces que definem o comportamento do usuário de acordo com a data de devolução do empréstimo.
Além disso nosso sistema foi dividido em pacotes diferentes para cada parte do sistema.
Para os itens usamos herança para definir os diversos tipos de item que existe.
A maioria das exceções foi lançada no controlador do sistema, e eram lançadas em casos como: dados em branco ou dados que não correspondem ao que o método pede.

Caso 1:

No primeiro caso é pedido para criar o usuário do sistema, para isso criamos a classe Usuario, que pode se relacionar entre si realizando empréstimos e mantendo ou emprestando itens. Os métodos do CRUD do usuário ficaram em Controller, alguns deles usando delegação para seu total funcionamento. Para armazenar o Usuario no Controller foi usado um Mapa onde a chave é um Objeto do tipo UsuarioKey (uma classe criada com o objetivo de tornar mais rápida a localização de usuário nesse mapa), e como valor um objeto do tipo Usuario.
Cada usuário possui uma lista de itens que ele possui, itens que ele emprestou, que está emprestando no momento e de itens que está emprestado de outro usuário.

Caso 2:

No segundo caso foi pedido a criação e o CRUD de itens, para isso usamos polimorfismo com herança, onde primariamente existia uma classe abstrata Item e dela herdavam outras classes que representavam os itens, cada item contendo suas respectivas características. Há também outra classe abstrata, a BluRay, da qual derivam todos os tipos de BluRay. O CRUD dos itens está no Controller, e também há uso de delegação.

Caso 3:

Para o caso 3 foram implementados os métodos necessários no Controller e na Facade para realizar o que era pedido. Primeiramente o método de listar os itens por ordem alfabética foi criado, para isso foi necessário a criação de uma classe NomeComparator que implementa Comparator, para que seja possível ordenar os itens pelo nome. Para ordenar pelo valor do item, foi feita a mesma coisa: criamos uma classe ValorComparator que implementava Comparator e assim implementamos o método responsável por lista por valor.
Para pesquisar detalhes de itens foi criado um método no Controller que recebe os parâmetros, constrói o objeto Item necessário, busca na lista de itens do Controller o item mais semelhante e o retorna.

Caso 4:

No caso 4 foi criada a classe Emprestimo, que é a classe responsável por guardar informações acerca dos empréstimos feitos entre os usuários, tais como os itens que foram emprestados, os usuários, a data do empréstimo e a data de devolução. Para coordenar os empréstimos feitos foi criada a classe ControllerEmprestimo na qual tem os métodos necessários para realizar um empréstimo e para encerrar empréstimos.
Para iniciar um novo empréstimo é criado um objeto do tipo empréstimo, e para a construção desse objeto será passado como parâmetro o usuário dono, o usuário que receberá o item emprestado, o item a ser emprestado, a data do empréstimo e a quantidade de dias que o item pode ser emprestado. Para o uso de datas foi utilizado o pacote java.time.
Ao realizar o empréstimo, o item do usuário que emprestará será passado a lista de itens pegos do usuário que receberá.
Para devolver o item, foi criado um método responsável por fechar o empréstimo, esse método irá receber a data de devolução e irá checar se há dias em atraso e em seguida calculará os pontos de reputação, que será visto no caso 6, após isso irá retornar o item para o dono e encerrar o empréstimo.

Caso 5:

Para realizar as funcionalidades relacionadas aos empréstimos foram criados métodos no Controller.
Para listar os empréstimos com uma certa ordem, foi criada a classe EmprestimoComparator, implementa Comparator. Essa classe irá auxiliar os métodos que irão ordenar os empréstimos de usuários que tem itens de outros usuários ou que está emprestando para outro usuário.
Para listar o histórico de empréstimos de um item foi atribuído um ArrayList de empréstimo onde estão todos os empréstimos que esse item já foi submetido, assim fica possível a listagem.
Para listar os itens emprestados no momento apenas é checado a lista de empréstimos que existe em Controller e listar os itens de cada empréstimo.
Os top 10 itens é feito a partir da checagem da quantidade de empréstimo que cada item possui em sua lista de empréstimos, os 10 itens com mais empréstimos são selecionados.
Para listar os itens emprestados no atual momento é apenas checado os empréstimos que estão presentes na lista de empréstimos em Controller.

Caso 6:

O cálculo da reputação foi feito em Emprestimo, sempre que um usuário devolvia um item sem atraso, sua reputação aumentava, e ao devolver o atraso sua reputação diminuía. No método responsável por fechar um empréstimo na classe Empréstimo, era comparada a data pré-estabelecida para a devolução do item a data em que o item foi devolvido, o próprio java.time comparada essas datas, em seguida era chamado um outro método que fazia todos os cálculos com base em dias e o valor do item emprestado, o resultado desse cálculo era atribuído ao Usuario em seu atributo reputacao.

Caso 7:

Para criar os cartões fidelidade foi utilizado o padrão strategy. Foi criada uma interface Cartao, e nessa interface existe um método que retorna a quantidade de dias que um Usuario que possui esse cartão pode passar com um item. Sendo assim foram criadas

classes para cada tipo de cartão fidelidade, onde essas classes implementam a classe Cartao e seus métodos.
No método de devolver empréstimo em Emprestimo, foi implementado uma nova função: ao fechar o empréstimo é calculado a reputação, então após isso é checado a reputação total do Usuario e então é atribuído a ele um cartão fidelidade, em um novo atributo cartao em Usuario.

Caso 8:

No caso 8 foi criada a classe ReputacaoComparator e ReputacaoInversoComparator que implementam Comparator, pois era necessário listar Usuarios de acordo com a sua reputação.
Para listar os Usuarios com reputacao menor que 0, todos os Usuarios da lista em Controller eram checados, os que possuem a reputação menor que 0 eram retornados.
Para listar por ordem de reputação, a lista de Usuarios de Controller era ordenado usando as classes criadas.

Caso 9:

Para o caso 9 foi pedido a criação de métodos que salvem o estado do sistema, para isso são criados os arquivos usuarios.txt e empréstimos.txt para armazenar os dados de usuários e empréstimos do sistema, em todas as classes foi implementada a interface Serializable para que tornasse isso possível.
Isso tudo foi implementado nos métodos “iniciarSistema” e “fecharSistema”, presentes e Controller e delegado na Facade. O “iniciarSistema” vai ler (se houver os arquivos) todos os dados de Usuario, então irá pegar o nome e o telefone de cada pra criar o UsuárioKey, pra poder colocar no HashMap de usuários do controller em tempo de execução, e assim então o “fecharSistema” irá salvar apenas usuarios.values(), utilizando o .writeObject(). O mesmo é feito com a lista de empréstimos em Controller.

Considerações:

Todos os métodos do Controller foram delegados na classe Facade, logo serão bem-sucedidos os testes de aceitação.

Link para repositório no GitHub:

https://github.com/ramonfragoso/ProjetoLP2