UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
EQUIPE: BRUNO RODRIGO PINHEIRO DE MENESES; SAMMARA BESERRA NUNES; VINÍCIUS JORGE SILVA
DISCIPLINA: LABORATÓRIO DE PROGRAMAÇÃO II
TURMA: 03

Relatório do Projeto Tracking Things

O projeto consiste na elaboração de um sistema que cadastra usuários, itens que pertencem a esses usuários e registra o empréstimos dos mesmos. Além disso, registra a reputação de cada pessoa que acessa o sistema baseado na devolução e empréstimo de itens. Tal atividade usou a aplicação de interface, Strategy, Herança e padrões GRASP, como será melhor detalhado a seguir. É interessante ressaltar que todas a entidades que geravam algum tipo de objeto tiveram acesso a um controller e consequentemente, acesso ao SystemController que repassava seus dados a Facede.

Casos de uso

- **Caso 1:**

No caso de uso 1, é pedido que o sistema realize o CRUD dos usuários dos sistema, cadastrando um nome, número de telefone e um e-mail. Em casos posteriores, essa entidade poderá cadastrar itens, removê-los e registrar empréstimos relacionado a esses itens. São identificados por um ID composto por nome e telefone nomeado IDUsuario. também podem ter uma certa reputação que irá definir seu futuro tipo de cartão fidelidade, no caso 7.

- **Caso 2:**

No caso de uso 2, é pedido que o sistema possa fazer o CRUD dos itens emprestáveis, seja capaz de adicionar, remover, pesquisar e atualizar o item, sempre mantendo seu nome, seu valor e seu status de empréstimo. Desse modo, criamos uma classe abstrata Item, onde fizemos extensões de acordo com o tipo do item pedido, podendo ele ser Blu-rays, que ainda podia ser estendido para os tipos Filme, Show ou Série, Jogos Eletrônicos e Jogos de Tabuleiros, e com isso, adicionamos um conjunto de itens (Set) no usuário, que ficaria responsável pela adição e remoção do mesmo, além disso implementamos os métodos responsáveis pela pesquisa dos itens, como o método *getInfoItem*, que retorna as informações do item, além de um método responsável pela atualização dos dados. Por fim, também foram adicionados as novas funcionalidades na fachada do sistema.

- **Caso 3:**

Neste caso, haverá a ordenação dos itens em dois aspectos: por nome e pelo valor. Essa ordenação será realizada pelo implementação da interface Comparator, apenas adaptando para quando o critério de ordenação for um número *double* ou uma *String*. Além da ordenação, é pedido que o usuário resgate informações de um certo item dado o nome do mesmo, o telefone do dono e o seu nome. Nisso, o nome e telefone do dono já constitui o IDUsuário ao qual o dono teste item pertence. Depois é só fazer uma busca no HashSet de itens que esse usuário possui e recuperar os dados buscados.

- **Caso 4:**

Este caso ficou responsável por registrar empréstimos e suas informações como item emprestado, dono do item, requerente do item, data de empréstimo, dias requeridos para o empréstimo e data de devolução do item. Além de registrar empréstimos, esse caso os controla, alterando dados.

- **Caso 5:**

No caso 5, trabalha com listagem de itens, assim como a ordenação dos mesmo relacionados a empréstimos. Semelhante ao caso de uso 3, aqui haverá a instância de métodos que ordenarão empréstimos por nome do item emprestado, enquanto usuários pegam emprestado ou empresta um item. Outra finalidade requerida é listar o histórico de empréstimos de um item na ordem que os empréstimos aconteceram, listar itens não emprestados por ordem alfabética, itens emprestados no momento e o top 10 de itens mais emprestados. Aqui, nos casos de ordenação, será implementado a interface Comparator e converter os HashSets em Lists para que a ordenação seja permitida.

- **Caso 6:**

Para o caso 6, foi pedido que o sistema possa ser qualificado para manter uma reputação para cada usuário, dessa forma, criamos uma nova entidade denominada Reputação, onde ela ficaria responsável por guardar e manipular seu valor (double), e a instanciamos no Usuário, sendo assim, fizemos as devidas alterações no controle de usuário, adicionando um novo método responsável por elevar ou reduzir seu valor, e utilizamos em cada método cujo a reputação sofreria alteração, desse modo, respeitando as diretrizes do caso 6.

- **Caso 7:**

Neste caso 7, entra em prática um importante uso da programação orientada a objetos: Strategy. Seria a mudança de comportamento no tipo que cada usuário tem dentro do sistema. No caso do TT, Os usuários podem “migrar” entre quatro tipos de comportamento diferentes que são atualizadas de acordo com suas reputações, mantidas no caso 6: Noob, FreeRider, Caloteiro e BomAmigo. Nessas características, durante a execução do sistema, cada um pode mudar certos atributos como permissão para pedir itens emprestado e máximo de dias que pode ficar com um item.

- **Caso 8:**

O caso de uso 8 pede que o sistema seja apto em poder listar os usuários baseados em sua reputação. Para tal, criamos um comparator que receberia a reputação dos usuários e utilizamos a classe Collections para realizar a ordenação, podendo ser em ordem crescente ou decrescente, a depender do método chamado, posteriormente listamos os usuários tendo como base o toString do mesmo, além de inserir as novas funcionalidades na fachada, no Sistema e no Controle de Usuários, no qual este último ficaria de fato responsável por tal funcionalidade.

- **Caso 9:**

Para o último caso, foi solicitado que o sistema pudesse armazenar em disco todos os dados cadastrados para manter seu estado mesmo que o programa seja fechado, ou seja, persistência. Para isso, nós implementamos os métodos responsáveis por tais ações, no Iniciar Sistema, realizamos uma leitura dos arquivos onde foram armazenados os dados que consolidam o sistema, por meio do Object InputStream, ao finalizar o programa, separamos as informações em dois arquivos, onde um ficou responsável por guardar a referência de todos os usuários cadastrados no sistema, e mais um outro arquivo contendo as informações relacionadas aos empréstimos dos itens, e realizamos a escrita por meio do Object Output Steam, valendo salientar que para a escrita foi necessário implementar o Serializable em algumas classes, deixando elas serializáveis, para que o armazenamento pudesse ser realizado com sucesso. Por fim, colocamos os novos métodos no Sistema, onde ficou responsável por tais ações, e posteriormente na Fachada.