Caso 1:

O caso 1 pede que seja criado um CRUD dos usuários de TT. A entidade que representa o usuário contém as informações de: nome, e-mail e número do celular do usuário, além de uma coleção de itens que o usuário possui que, neste caso, implementamos um mapa de itens, já que a existência de um identificador do item facilitaria na hora de recuperar informações e evitaria iterações desnecessárias. Inicialmente, foi cobrado apenas que informações sejam adicionadas, removidas, pesquisadas e atualizadas na entidade que representa o usuário e todas essas operações foram implementadas no controlador do usuário .Outra condição que foi estabelecida foi que dois usuários seriam iguais se tivessem o mesmo nome e o mesmo número de celular, para isso, fizemos um *hashcode* que envolve estes dois atributos do usuário. Além disso, criamos um *controller* de usuários, para que este gerenciasse o mapa de usuários, que este teria como referência um *token* para cada usuário, sendo resultado da concatenação do nome e do telefone do usuário, não sendo possível o cadastro de usuários iguais.

Caso 2:

Já o segundo caso trata de um **CRUD** de itens emprestados. Todo item deve ter um nome, valor e se está emprestado ou não. Por não se tratar de uma classe mais genérica e que não seria utilizada em sua forma proposta, tornamos o item uma classe abstrata que estende características às suas subclasses, que são mais específicas e retratam os objetos que serão emprestados na prática. Além disso, criamos as subclasses do item: **BluRay**, **Jogos Eletrônicos** e **Jogos de Tabuleiro**. Na entidade *BluRay*, percebemos que essa se dividia em outras classes menores e mais específicas referente à shows, séries e filmes. Cada um desses *BluRays* tem características específicas, mas todos são subclasses do *BluRay* e herdam atributos e comportamentos do *BluRay*. Além disso, implementamos as condições de igualdade para os diferentes tipos de *BluRays*. Criamos também *Enums* para nos referirmos aos tipos específicos de Jogos Eletrônicos, um *controller* para itens, que é responsável pelo armazenamento de todos os itens dos usuários, assim como, criamos um mapa dos itens dos usuários, utilizando seu nome como identificador dos itens, o que facilitaria na recuperação dos dados do item e evitaria iterações desnecessárias, não sendo possível cadastrar dois itens iguais na coleção de itens do usuário.

Caso 3:

No caso 3 é requisitado que implementemos 3 métodos nos quais 2 deles pedem que listemos todos os itens já cadastrados usando um atributo do item como comparador e para isso criamos uma classe chamada ItemController que ele terá as informações de todos os itens já cadastrados no sistema e assim podemos pegar essa estrutura de dados e ordena-lo da forma que quisermos, no primeiro caso precisamos ordenar os itens em

ordem alfabética usando o nome do item como comparador assim usamos o método sort para ordenar a lista usando esse atributo como comparador e após isso simplesmente concatenamos as representações textuais de cada item e retornamos essa informação para a facade, no segundo método o mesmo ocorre só que nesse caso usamos o atributo preço do item como atributo a ser comparado,já no terceiro método a ideia e diferente dos outros dois que no caso o método ele serve pra que o usuário que esta usando o sistema possa pesquisar mais detalhes sobre um item,usando como parâmetros o nome do usuário ,seu telefone e o nome do item,assim o controle de usuarios irá acessar a chave no mapa de usuários que contem o usuário que tem o nome e telefone requisitados assim o usuário ira repassar a informação do item que esta sendo pedida retornando-a em forma de String.

Caso 4:

No caso 4 o sistema permite o empréstimo entre os usuários em consequência uma nova classe foi criada: Empréstimo e ela serve para guardar as informações do empréstimo tendo como atributos: dono do item emprestado, o usuário que pegou o item emprestado, o item emprestado,data inicial do empréstimo, número de dias combinados para empréstimo, data real de devolução do item e se está em atraso o empréstimo e a quantidade de dias em atraso ,sendo que foi necessário fazer métodos para poder determinar a data de vencimento do empréstimo com isso pegamos a informação em String da data do empréstimo e a convertemos para Date no formato dd/MM/yyyy para que depois possamos usar o método set date para poder adicionamos o período padrão para a devolução do item que era 7 dias assim poderíamos construir o atributo o numero de dias combinados para o empréstimo do item e quando for requisitado para verificar se o item foi entregue em atraso esta classe tem o método que recebe a data de entregue e caso seja valida o método a transforma em Date e usa o método after para saber se o dia de devolução ultrapassou o vencimento pre-estabelecido na criação do caso esteja atrasado a devolução o atributo que diz se ta atrasado fica igual a verdadeiro e o método calcula os dias em atraso.

Caso 5:

No caso 5 agora é necessario implementar 6 métodos novos no qual o primeiro metodo coonsiste em fazer o controlador de usuarios usar o metodo do usuario para retornar uma listagem dos itens que o usuario pegou emprestado ordenados em ordem alfabetica,no caso o codigo pega o array de itens que foram emprestados os ordena e o retorna o mesmo e feito no segundo metodo a diferença é o array utilizado que agora e dos itens que o usuario emprestou , no terceiro metodo dessa vez sera acessado o controlador de itens para que ele possa pegar o item cadastrado no array de itens gerais e assim listar os emprestimos associados a este item,simplesmente o controlador de item vai invocar o metodo que retorna o historico dos emprestimos em que esse item participou,no quarto metodo e necessario a listagem de itens que não foram emprestados,neste caso o controlador de itens vai diferenciar os itens que foram emprestador dos que não forame simplesmente via concatenar os toString de cada item e depois vai retornar para que esse retorno chegue na facade,no quinto metodo e necessario agora que o controlador de itens ordene os itens pelo o numero de emprestimos que este item participou que no caso esse numero e um atributo de item que incrementa 1 no atributo toda vez que ele e emprestado após ter os itens ordenados o controlador simplesmente concatena os toString() desses objetos com quebra de linha para repassar para a classe que invocou esse metodo já o sexto metodo ira fazer o mesmo que o quarto metodo só que em vez de pegar os itens não em-

prestados ele ira pegar os itens emprestados e ira listados e retornar para a classe que invocou esse metodo.

Caso 6:

Neste caso de uso foi necessario implementar um sistema de reputações de cada usuario ,no qual cada usuario agora tem um atributo em numero decimal que define a sua reputação inicialmente a reputação do usuario e de acordo com 5 por cento do valor de todos os itens que este usuario possui,ou seja,toda vez que o usuario for cadastrar um item em seu nome sua reputação aumentarar em 5 pro cento mas caso o usuario não tenha itens sua reputação e zero,quando o dono do item empresta um de seus itens sua reputação aumenta em 10% do valor do item emprestado,ja para o usuario que pediu o item emprestado pode ocorrer 2 casos o primeiro caso é se ele entregar o item ate a data limite sua reputação aumenta em 5% do valor do item , já a outra situação caso ele atrase o sistema deve diminuir sua reputação em uma proporção de acordo com os dias atrasados, por exemplo caso usuario atrase 2 dias deve ser descontado de sua reputação 2% do valor do item.

Caso 7:

Neste caso de uso foi necessario criar uma classificação de dos usuarios de acordo com a sua reputação (Noob,Caloteiro,Bom Amigo,Free Ryder) cada classificação desse tem um periodo maximo de dias que o usuario pode pegar um item emprestado,foi criado um enum para representar isso e quando o sistema vai fazer açoes que utilizam emprestimos ele tem que alterar seu tipo de cartao que no caso e essa classificação respeitando um intervalo logo o controlador de usuarios pega a reputação do usuario e verifica sua reputação e muda seu tipo do cartao ,quando ocorre um emprestimo o alem de acessar o controlador de emprestimos para criar o emprestimo o mesmo verifica se o emprestimo e valido e quando vai emprestar o item ele faz as mudanças ditas no caso 6
sobre emprestar e devolver o item e para cada ação dessa a reputaçao de ambos usuarios é mudada em consequencia seu tipo de cartão pode ser mudado de acordo com a sua reputação fazendo a mudança em tempo de execução e verificado a reputação atraves de outro metodo do controlador e é mudado o atributo mas há casos que o usuario não pode pegar itens emprestados logo o sistema joga uma exception anunciando que o usuario esta inelegivel para pegar emprestado os itens.