Relatório de projeto E-Câmara Organizada(E-CO)

O design do projeto foi decidido inteiramente pelo grupo tendo que sofrer alterações em alguns casos para que se adequasse ao que era pedido na especificação. Com isso, o nosso projeto implementou herança entre as propostas e entre pessoa e deputado, e foi decidido também que seria necessário 3 controladores, um para pessoa e deputado, um para as propostas de leis e outra para as comissês, e todos eles eram controlados por um controlador geral.
Com relação a exceções e tratamento de erros, nós utilizamos uma classe validador que verificava os erros mais comuns e previsíveis, e para outras mais específicas ela é capturada no próprio método que recebe os parâmetros que serão verificados.

Caso 1:
Para o caso 1 foi criado uma classe pessoa que armazenava seus atributos e métodos específicos que possui dois tipos de construtores diferentes dependendo da pessoa possuir ou não um partido político. Outro atributo de Pessoa é o seu dni que é o mais importante da classe por servir como um identificador dessa pessoa.

Caso 2:
Para o caso 2 criamos uma classe deputado que herda da classe pessoa e além disso possui alguns atributos e métodos específicos que são usados somente pela classe deputado como o setLeisAprovadas() que toda vez em que é chamado ele adiciona em 1 o número de leis daquele deputado que foi aprovado. Outro método importante também que é sobrescrito através do override é o exibir de deputado que possui algumas alterações sutis em relação ao exibir da classe pessoa.

Caso 3:
Para o caso 3 como foi brevemente descrito anteriormente, foi feito um tipo de exibição na classe pessoa que levava o seu nome, dni, estado, se ele pertencesse a um partido seria exibido também e por fim os interesses da pessoa. Já para deputado seria necessário incluir o partido, a data de início de seu mandato e a quantidade de leis aprovadas desse deputado.

Caso 4:
O caso 4 no geral foi o mais simples de ser implementado, era necessário somente criar uma lista de Strings, que toda vez que um partido da base do governo era cadastrado, ele era exibido no sistema e para a sua exibição utilizamos um sort padrão para ordenar em ordem alfabética no momento de sua impressão.

Caso 5:
Para o caso 5 criamos uma classe comissão que armazena uma lista dos deputados que fazem parte dessa comissão e também armazena o tema da comissão e como

métodos ele possui o método que retorna todos os integrantes da comissão para ser usado na hora da votação de um projeto, que será melhor explicado no caso 7, e também um cadastraIntegrante() que é usado na hora do cadastro da comissão para cadastrar todos os deputados na comissão apropriamente.
Caso 6:
Para o caso 6 foi criado uma classe chamada PropostaLegislativa que armazena todos os atributos e métodos comuns entre os diferentes tipos de propostas diferentes, como o autor do projeto, o código, ano de criação, ementa, interesses da proposta, situação, URL, o local atual da proposta e uma lista de Strings que armazena toda a tramitação da proposta no sistema. Para a PL que herda da classe PropostaLegislativa ela possui de específico o atributo booleano conclusivo que responde se a PL é conclusiva ou não, e possui também um método toString() específico de PL. Para PLP e PEC elas possuem como atributo específico os artigos da constituição que serão afetados com a aprovação dessa proposta mas as duas possuem um método toString diferente para ser diferenciado além de próprio código como identificador único de cada proposta.

Caso 7:
Para o caso 7 foi criado uma classe Votacao que executa uma votação no sistema e também executa todas as alterações necessárias no sistema dependendo do resultado da votação e recebe como parâmetro todos os atributos necessário para executar uma determinada votação, caso seja feita em uma comissão, o método votaComissão() é chamado e possui como parâmetro a comissão onde será realizada a votação, a proposta que será votada, o próximo local de votação e por fim o autor da proposta que será votada. Caso a votação seja realizada em plenário o método votaPlenário() será chamado e possui como parâmetro a proposta que será votada, uma lista com os deputados presentes na votação, o número total de cadeiras no plenário, a posição política da proposta e o autor da proposta que será votada.
Uma coisa interessante desse caso é que os métodos de votação possuem muitas condições dependendo do resultado da votação e dependendo do tipo de proposta.

Caso 8:
Para o caso 8 não houve muita dificuldade para a sua implementação a única particularidade era que para ele ser implementado é necessário uma implementação do caso 7 totalmente correta para que a impressão da tramitação que está na classe PropostaLegislativa que é a chamada getTramitação() ser retornada corretamente. No geral foi um caso muito simples de ser implementado.

Caso 9:
Para a implementação do caso foi preciso criar um atributo novo em Pessoa que armazena o tipo de estratégia para essa pessoa executar o desempate entre as

propostas. Foi criado uma interface Estrategia e 3 classes de estratégias diferentes que implementam essa interface e cada uma dessas 3 classes possui um comparador diferente que recebe uma lista e retorna o código da proposta mais adequada ao tipo da estratégia.

Caso 10:
Para a implementação do caso 10 se fez necessário a criação de um atributo novo em todas as classes do sistema que é o identificador serial da classe e com isso também se fez necessário a implementação da interface Serializable em cada classe do sistema que é necessário para que o sistema possa ser armazenado e salvo em um arquivo. Para o método de salvar é passado o controlador geral e ele é serializado e colocado em um arquivo. Para carregar o código executa a leitura do arquivo e retorna o objeto que está armazenado no arquivo. Para limpar o sistema é executado um reset do controlador geral que está sendo executado e o que está armazenado é resetado também.

Integrantes:
Natan Vinícius da Silva Lucena: Responsável pela implementação parcial dos casos 1 a 3, implementação dos casos 4 a 10 e documentação parcial do código.

Cayo Vinicius Viegas: Responsável pela implementação parcial do caso 1, implementação parcial da classe ValidadorGeral, implementação e correção dos testes de unidade e documentação parcial do código.

Jackson Mateus: Responsável pela implementação parcial dos casos 1 a 4,implementação e correção dos testes de unidade e documentação parcial do código.

Lucas Leal de Lucena: Responsável pela implementação do caso 3, implementação parcial do caso 9, correção do código dos casos 1 a 6 e implementação parcial dos testes de unidade das classes entidades e documentação parcial do código.