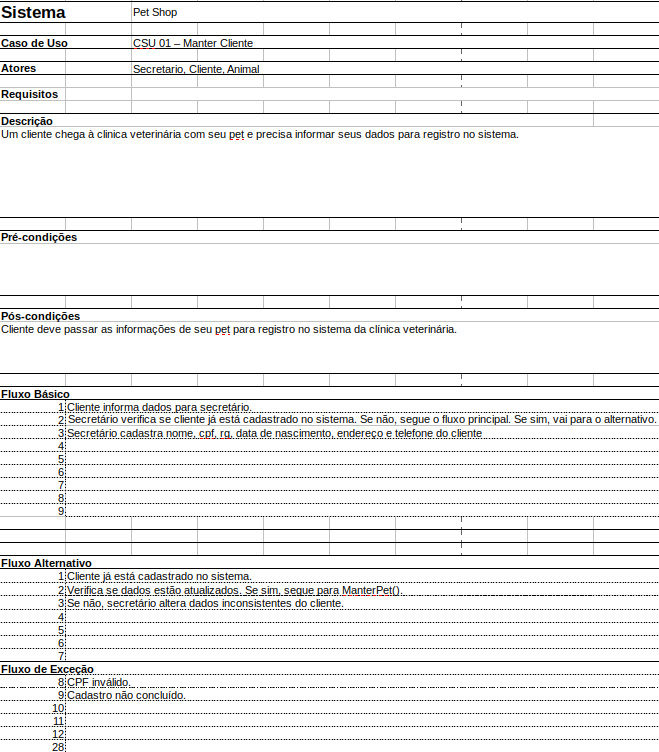
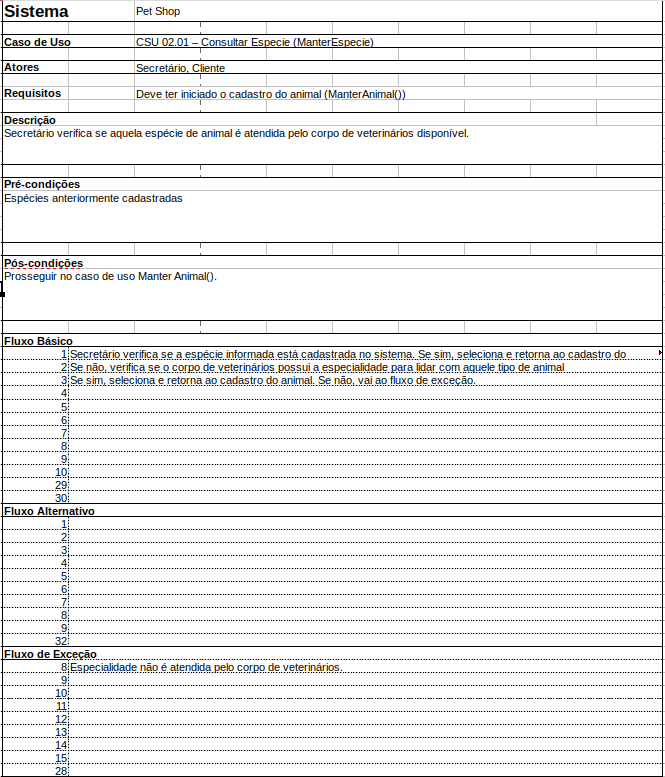
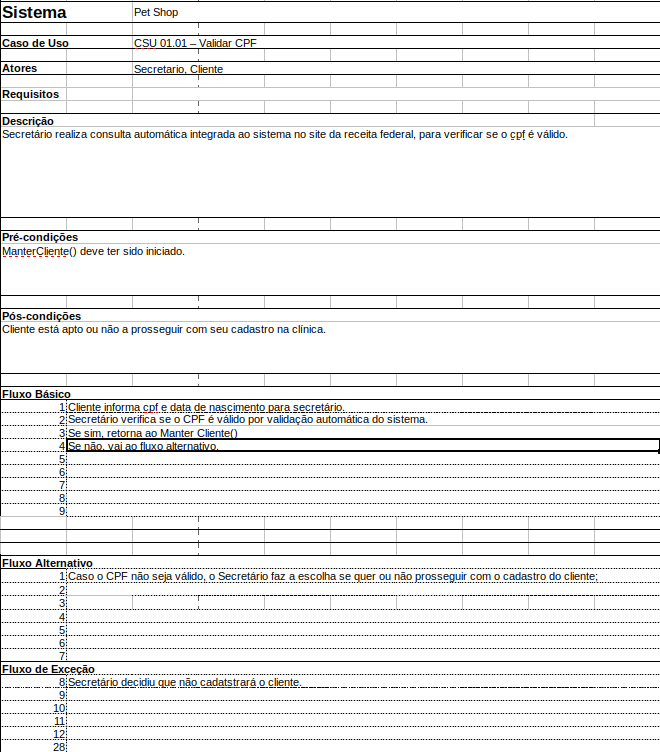
# Lista de Exercícios para a N1 - Engenharia de Software III

1- Especifique textualmente o caso de uso Manter Cliente (CSU01), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado.





2- Especifique textualmente o caso de uso Manter Animal (CSU02), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado. O caso de uso Manter Espécie deve ser adequado e especificado como caso de uso incluído (<<include>>) do caso de uso principal, isto é, o CSU02.

3- Especifique textualmente o caso de uso Manter Veterinário (CSU03), apresentando os fluxos (cenários)

principal, alternativo e de exceção, de acordo com o template disponibilizado.

4- Especifique textualmente o caso de uso Marcar Consulta (CSU04), apresentando os fluxos (cenários)

principal, alternativo e de exceção, de acordo com o template disponibilizado.

5- Especifique textualmente o caso de uso Realizar Consulta (CSU05), apresentando os fluxos (cenários)

principal, alternativo e de exceção, de acordo com o template disponibilizado. Os casos de uso Manter

Histórico e Gerar Conta devem ser especificados como casos de uso incluídos (<<include>>) do caso de uso

principal, isto é, o CSU05. Deve surgir um novo caso de uso denominado Solicitar Exame como caso de uso

estendido do caso de uso principal, isto é, o CSU05.

6- Especifique textualmente o caso de uso Marcar Exame (CSU06), apresentando os fluxos (cenários) principal,

alternativo e de exceção, de acordo com o template disponibilizado. Este caso de uso deve ser adaptado e

especificado como caso de uso principal, retirando o mesmo como caso de uso estendido (<<extend>>) do

caso de uso Realizar Consulta (CSU05).

7- Modele um novo Diagrama de Casos de Uso (DCU) com base nas especificações textuais.

8- Modele uma VCP para o CSU01, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU01.

9- Modele uma VCP para o CSU02, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU02.

10- Modele uma VCP para o CSU03, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU03.

11- Modele uma VCP para o CSU04, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU04.12- Modele uma VCP para o CSU05, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU05.

13- Modele uma VCP para o CSU06, utilizando a categorização BCE. A classe de controle deve apresentar dois

métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos.

Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU006.

14- Modele um diagrama de sequência para o CSU01. Se a interação for complexa, definam os quadros de

interação.

15- Modele um diagrama de sequência para o CSU02. Se a interação for complexa, definam os quadros de

interação.

16- Modele um diagrama de sequência para o CSU03. Se a interação for complexa, definam os quadros de

interação.

17- Modele um diagrama de sequência para o CSU04. Se a interação for complexa, definam os quadros de

interação.

18- Modele um diagrama de sequência para o CSU05. Se a interação for complexa, definam os quadros de

interação.

19- Modele um diagrama de sequência para o CSU06. Se a interação for complexa, definam os quadros de

interação.

20- Modele um Diagrama de Classes de Projeto a partir das VCPs modeladas e dos diagramas de sequência, e mantenha a utilização da categorização BCE. Os devidos atributos e métodos devem continuar sendo exibidos. As multiplicidades dos relacionamentos devem ser apresentadas.

### 21- Qual é a classe de entidade mais coesa e a menos coesa do diagrama de classes de projeto? Justifique a tua resposta.

*As classes mais coesas são aquelas que possuem pouca ou nenhuma dependência de outras classes. Logo, a mais coesa do nosso diagrama é Veterinario, que não depende de nenhuma informação de outras classes para persistir.*

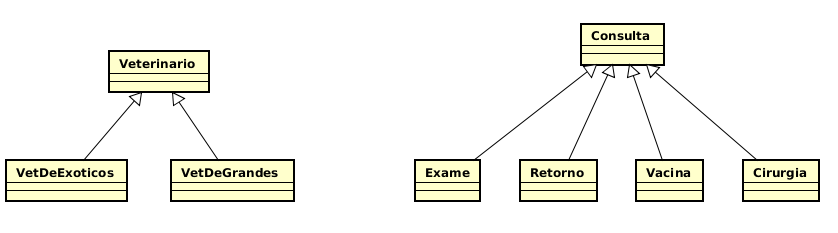
*Já a classes menos coesa é Consulta, que depende de Pet, Veterinario, Exame e Cliente.*

22- Qual é a classe de entidade mais acoplada e a menos acoplada do diagrama de classes de projeto? Justifique a tua resposta.

*A classe mais acoplada é Pet, pois é excluída caso o Cliente também seja excluído, ou seja, é totalmente dependente.*

*A classe menos acoplada é Veterinário, que não depende de nenhuma outra parte do sistema para ser implementada. É totalmente independente em sua criação.*

23- Modele duas relações de gen/espec e ative o princípio de polimorfismo universal de inclusão em cada uma delas. Justifique a razão de existência de cada gen/espec e das operações polimórficas.

*Esse modelo, no qual há o princípio de polimorfismo universal, permite que um objeto pertença a várias classes ao mesmo tempo, ou seja, a classe pai aponta para um objeto da classe filha e permite que essa seja instanciada como uma ou como outra.*

*class Consulta{}*

*class Exame extends Consulta{}*

*class Retorno extends Consulta{}*

*Consulta c1 = new Exame();*

*Consulta c2 = new Retorno();*

*24 – Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de gen/espec e as operações polimórficas.*

25- As relações de gen/espec modeladas violam o Princípio de Liskov? Justifique a tua resposta.

*Não. Todas as classes descritas podem ser substituídas pela sua superclasse.*

26- Quais restrições {OCL} sobre gen/espec são aplicáveis nas relações modeladas? Justifique a tua resposta.

???

29- Modele seis membros estáticos, sendo três atributos e três métodos. Justifique a criação de existência de cada um dos membros estáticos modelados.

Os métodos e atributos estáticos são aqueles que não têm seu conteúdo alterado durante a execução do programa. Se ele tem uma função específica, como zerar o valor de uma variável, vai permanecer eternamente com essa atribuição.

Métodos que podem ser estáticos, dentro do escopo do Sistema de Clínica Veterinária:

*SistemaDePagamento.emitirBoleto();*

*Cliente.zerarConsultasAcumuladas();*

*Cliente.obterNumConsultas();*

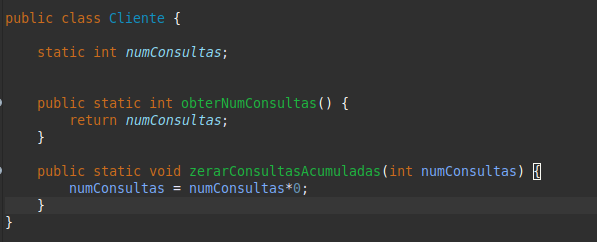
Atributos que podem ser estáticos, dentro do escopo do Sistema de Clínica Veterinária:

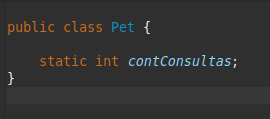
*Cliente = static int numConsultas;*

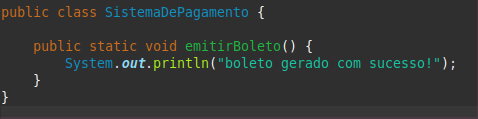
*Veterinario = static String crmv;*

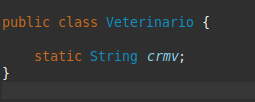
*Pet = static int contConsultas;*

30- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar os seis membros estáticos.









31- Transforme todos os relacionamentos de associação ou agregação entre as classes de entidade e todos os relacionamentos entre as classes de fronteira e controle para dependências estruturais. Explique a vantagem e desvantagem desse tipo de dependência.

*A dependência estrutural é um tipo de dependência mais fácil de ser implementado e pensado. Nessa dependência, a classe dependente possui um atribuo que referencia outra classe.*

*Isso acaba ocasionando um alto acoplamento (baixa coesão). Por isso, precisamos observar se as dependências das classes não podem ser modificadas para dependências não estruturais.*

32- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências estruturais.

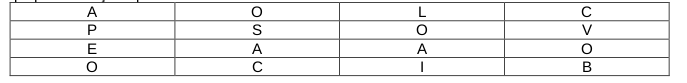
*Já foi feito nos exercícios anteriores… acredito que tenhamos que focar apenas em melhorar nosso diagrama…*

*36- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as classes parametrizadas com a estrutura <List> e <Set>.*

*38- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar a estrutura de dados <TreeSet>.*



### 39- A partir do caça objetos abaixo, encontre no mínimo cinco objetos e modele uma relação todo-parte, abstraindo o nome de uma ave na classe todo e os cinco objetos encontrados como partes deste todo. Explique a solução apresentada.



A classe Ave possui 5 atributos essenciais para ser construída. Foi instanciada como uma galinha.

Class Ave {

String bico;

String asa;

String pena;

String ovo;

Boolean voa;

public Ave (String bico, String asa, String pena, String ovo, boolean voa){

this.bico = bico;

this.asa = asa;

this.pena = pena;

this.ovo = ovo;

this.voa = voa;

}

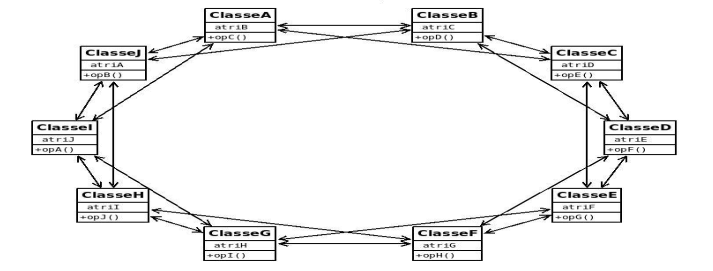
Class Natureza {

public static void main (String [] args) {

Ave galinha = new Ave(“curto”, “pequenas”, “finas”, “pequeno”, false);

}

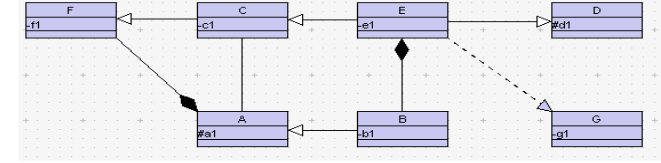
### 40- Analise o diagrama de classes abaixo. Descreva a coesão e acoplamento desse diagrama. Modele um diagrama de classes que seja melhor em termos de coesão e acoplamento e explique o porquê do novo diagrama de classes ser mais vantajoso do que o abaixo.



Todas as classes desse diagrama apresentam alto acoplamento e baixa coesão. Isso significa dizer que as classes possuem múltiplas dependências umas das outras, então se uma delas for modificada, todo o ciclo vai ser quebrado, pois uma classe sofreu alteração. Para um projeto profissional, de negócios, é uma situação inapropriada. Imagine que está trabalhando em um projeto com mais de 100 classes interdependentes (alto acoplamento) e precisa executar alteração em uma delas. Todo o projeto necessitará de refatoração e isso, numa empresa, envolve tempo e desperdício de tempo é desperdício de dinheiro. Devemos, então, aumentar a coesão e diminuir o acoplamento para que as classes possam ser modificadas sem maiores influências em um grande número de outras classes.

Para remodelar o diagrama acima, precisaríamos conhecer minimamente o domínio, pois no diagrama não está nem indicado qual é a relação de atributos entre uma classe e outra. A única coisa que se pode concluir é que precisaríamos inserir moderadores de acesso (public, private, protected e default) e, ao menos, uma ou mais classes abstratas ou interfaces.

### 41 - Identifique cada um dos relacionamentos utilizados no diagrama de classes abaixo, inclusive se são simétricos ou assimétricos. Quais atributos a instância de cada classe pode visualizar? Justifique a tua resposta.



Relacionamentos:

Dependência: Uma classe depende da outra para que toda a sua funcionalidade seja implementada corretamente.

E - G → dependência. E necessita de G.

Composição: Quando o container é destruído, o conteúdo também é.

Então, no caso, as relações de composição presentes entre as classes:

A - F → A possui F. Se A é destruída, F também é

E - B → E possui B. Se B é destruída, E também é.

Generalização/Especialização: representa a herança entre classes.

B - A → Herança - B herda de A.

F - C → Herança - C herda de F.

E - C → Herança - E herda de C.

E - D → Herança. E herda de D.

A classe E possui herança múltipla.

Visibilidade de atributos:

B - A → como B herda de A, B tem permissão para visualizar os atributos protegidos de A.

E - D → como E herda de D, E tem permissão para visualizar os atributos protegidos de D.

Todos os outros atributos de outras classes são visíveis apenas para a própria classe, visto que são atributos privados.

43- Descreva as características de cada um dos programadores cegos.

Programador 1 – descreve o animal por cima, toca o animal e explica aos outros. Analisa profundamente, coloca o ouvido perto do bicho, análise mais material das coisas.

Programador 2 – descreve o animal mais minuciosamente, observa com mais atenção os detalhes sonoros.

Programador 3 – metódico, calculista, presta muita atenção nos detalhes, nas texturas, abre caixas

Programador 4 – sente cheiros e percebe que tem uma base embaixo do bicho, todas bagunçadas, percebe q a criatura anda em bando

Programador 5 – fina arte, historiadora, de vez em quando artista e psicóloga. Carrega uma escada. Olhar as coisas de cima. O conjunto da obra. Uma mulher de visão. Vê luz e escuridão e formas, de modo meio abstrato.

44- Qual é a relação da estória apresentada com o projeto de desenvolvimento de software?

A conexão diz que é importante ter diferentes pessoas trabalhando em um mesmo projeto, pois cada um tem um ponto de vista e pode identificar melhorias que o outro pode não ter visto.

45- Produza uma charge, a mão, abordando a estória do elefante e dos cinco programadores. Explique o significado da sua charge.

Os participantes da charge, ou do projeto de construção do elefante, não conversaram muito antes de decidir pela melhor opção de como seria o animal e, por isso, criaram algo fantasioso e que não atende tão bem às funcionalidades que deveria.

