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

### 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.*

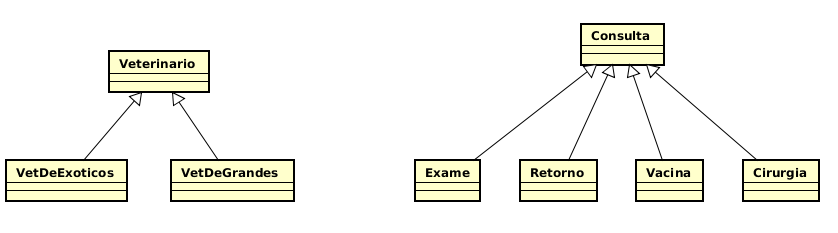
*Já a classes menos coesa é Consulta, que depende de Pet, Veterinario, Exame e Cliente.*

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 – está no Eclipse*

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:

SistemaDePagamento.emitirBoleto();

Cliente.zerarConsultasAcumuladas();

Cliente.obterNumConsultas();

Atributos que podem ser estáticos:

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.

*Feito no Eclipse.*

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 dependencia, 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 depedências não estruturais.*

*FALTA COLOCAR AS CLASSES DE FRONTEIRA NO DIAGRAMA E DEPOIS TERMINAR ESSE EXERCICIO*

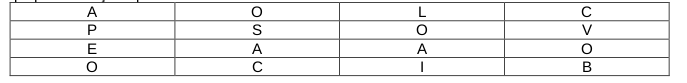
32- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências estruturais.

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

### 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.



Saco

Boia

SAPO

BOI

VACA

LEAO

CASACO

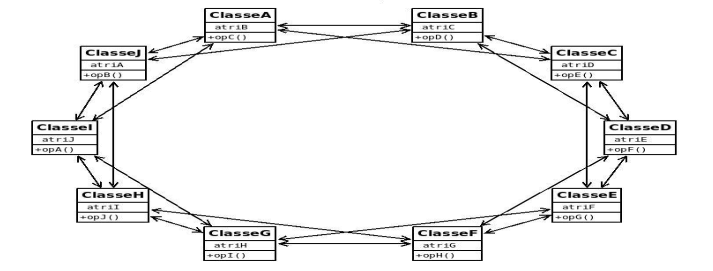
SOCO

BOLO

ESPACO

nao entendi esse exercicio

### 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.



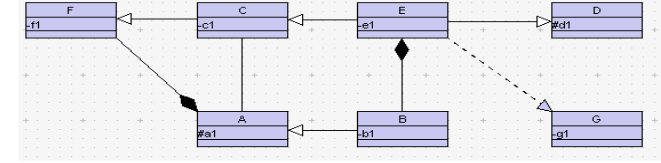
Todas as classes desse diagrama apresentam alto acoplamento e baixa coesão. Isso significa dizer que as classes possuem múltiplas dependências umas das outras, então se uma delas for modificada, todo o ciclo vai ser quebrado, pois uma classe sofreu alteração. Para um projeto profissional, de negócios, 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.

O diagrama remodelado seria, mais ou menos, dessa forma:

Precisamos inserir moderadores de acesso (public, private, protected e default)

FALTA REMODELAR ESSE DIAGRAMA

### 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.