Projeto de Sistema de Software de Clínica Veterinária

Lista 2

- Os exercícios devem ser apresentados na mesma ordem dos enunciados e devem conter uma sequência lógica. Os exercícios são referentes ao mesmo sistema, portanto deve haver coerência entre eles. Os exercícios que não estiverem numa sequência lógica serão devidamente descontados.

- A Lista deve ser realizada em grupo de 3 a 4 integrantes, mas apenas um aluno precisa entregar a Lista por e-mail. Favor copiar os outros integrantes do grupo em cada e-mail enviado. Exceções devem ser tratadas com o próprio professor com antecedência.

- Os diagramas devem ser construídos em alguma ferramenta CASE, mas a Lista deve ser entregue no formato digital no e-mail pwvendramel@gmail.com em um único arquivo PDF até às 23h00 de 22/04/2018.

- Os slides dos capítulos 5, 7 e 8 podem apoiar a realização da maioria dos exercícios desta Lista.

- Para cada exercício em branco, incompleto ou que não atenda o enunciado, será subtraído 1 ponto da Nota de Listas conforme explicado no primeiro dia de aula. Em determinados casos, o desconto pode ser de 0,5 ponto.

- Listas com respostas suspeitas de plágio serão devidamente anuladas e "zeradas". Os exercícios com respostas iguais entre grupos diferentes serão anulados e descontados. O aluno poderá ser convidado para resolver questões durante a aula com o objetivo de validar os exercícios da Lista.

Parte A: Exercícios sobre o Projeto

1- Construa uma nova versão refinada do diagrama de classes de projeto a partir dos exercícios 14, 17, 19, 21, 23, 25 e 27. Troque os <<estereótipos>> da categorização BCE pelos <<estereótipos>> do padrão de projeto MVC. A notação de classe deve continuar sendo mantida e as multiplicidades dos relacionamentos devem ser exibidas.

2- As relações de gen/espec modeladas apresentam classificação dinâmica? Justifique a tua resposta.

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

4- Modele concomitantemente as dependências não estruturais por parâmetro e por variável local entre as classes de controle e modelo. Para cada relação entre as classes de controle e modelo, justifique o motivo de ter escolhido o tipo de dependência não estrutural.

5- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências não estruturais por parâmetro e por variável local

6- Modele concomitantemente as classes parametrizadas com a estrutura <List> e <Set> para resolver o lado muitos dos relacionamentos. Para cada classe parametrizada modelada, justifique o motivo de ter escolhido o tipo de estrutura de dados.

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

8- Modele a estrutura <TreeSet>. Justifique a razão dessa estrutura no seu diagrama. Represente graficamente como essa árvore trabalharia em tempo de execução.

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

10- Modele três relações de interface com nomes adjetiváveis, estabelecendo o devido contrato de comportamento entre as classes consumidoras e fornecedoras e declarando as operações nas interfaces a

serem implementadas pelas classes fornecedoras. Justifique a razão de existência de cada uma das relações de interface.

11- Apresente a estrutura básica de código em JAVA ou C# para implementar as relações de interface.

12- Modele duas relações de delegação com relacionamento de composição, utilizando classes diferentes para cada uma. Justifique a razão de existência de cada uma das relações de delegação.

13- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de delegação com relacionamento de composição.

14- Transforme as duas relações de delegação com relacionamento de composição para relações de delegação com dependência estrutural. Justifique a tua resposta.

15- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de delegação com relacionamento de dependência estrutural.

16- Independentemente das relações de gen/espec já existentes no diagrama, abstraia o domínio e modele uma relação de herança múltipla. Justifique a razão de existência dessa herança múltipla.

17- Apresente a estrutura básica de código em C++ para implementar a relação de herança múltipla.

18- Transforme a relação de herança múltipla para uma relação de realização. Justifique a tua resposta.

19- Apresente a estrutura básica de código em JAVA ou C# para implementar a relação de realização.

20- Transforme a relação de herança múltipla para uma relação de delegação com relacionamento de composição. Justifique a tua resposta.

21- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar a relação de delegação com relacionamento de composição.

22- Faça um quadro comparativo entre reuso por generalização, realização e delegação, apresentando no mínimo duas vantagens e duas desvantagens para cada um desses conceitos.

23- Modele o padrão de projeto Composite. Qual o propósito desse padrão no diagrama de classes de projeto?

24- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Composite.

25- Construa um diagrama de sequência para o CSU05 utilizando notação MVC.

Parte B: Atividades de Abstração

1- 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. Só vale para esta atividade os objetos encontrados na horizontal, vertical ou diagonal. Explique a solução apresentada.

A	O	L	C
P	S	O	V
E	A	A	O
O	C	I	B

2- Construa um diagrama de classes para representar a classe Amphibia. O diagrama deve apresentar as heranças e pelo menos uma operação polimórfica para contextualizar o domínio. Explique a razão de existência de cada classe e do polimorfismo. É importante ler um pouco sobre o domínio para construir esse diagrama.