Lista 2

Instruções Gerais:

- Os exercícios devem ser apresentados na mesma ordem dos enunciados e devem conter uma sequência lógica. Todos 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 pode ser realizada individualmente, em dupla ou em trio, 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 slides dos capítulos 5 e 8 podem apoiar a realização da maioria dos exercícios desta Lista.

- 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 17/09/2016.

- 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 alguma questão durante a aula com o objetivo de validar os exercícios da Lista.

Parte A: Os exercícios dessa parte devem ser feitos com base no exercício 19 da Lista 1.

1- Apresente a modelagem das relações de gen/espec do exercício 19 da Lista 1.

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

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

4- As relações de gen/espec apresentam problema de Classificação Dinâmica? Justifique a tua resposta.

Parte B: Os exercícios dessa parte devem ser feitos com base no exercício 18 da Lista 1.

5- Apresente o diagrama de classes de projeto do exercício 18 da Lista 1.

6- Transforme todos os relacionamentos de associação ou agregação entre classes de modelo para dependências estruturais. Explique a vantagem e desvantagem desse tipo de dependência.

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

8- Transforme todos os relacionamentos de associação entre as classes de controle e modelo para dependências não estruturais por parâmetro. Explique a vantagem e desvantagem desse tipo de dependência.

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

10- Transforme todos os relacionamentos de associação entre as classes de controle e modelo para dependências não estruturais por variável local. Explique a vantagem e desvantagem desse tipo de dependência.

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

12- Modele as classes parametrizadas com a estrutura <List> para resolver o lado muitos dos relacionamentos. Por que tais classes foram modeladas?

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

14- Modele as classes parametrizadas com a estrutura <Set> para resolver o lado muitos dos relacionamentos. Por que tais classes foram modeladas?

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

16- Qual a diferença entre classe parametrizada, multiobjetos e collection?

17- Modele três interfaces 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 interfaces.

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

19- Modele duas relações de delegação, utilizando classes diferentes para cada uma. Justifique a razão de existência de cada uma das relações de delegação.

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

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