MC302 – Programação Orientada a Objetos 22/03/2016

Instituto de Computação – UNICAMP

Prof.: Cecília Rubira

LAB04: Agregação e *Composite*

INSTRUÇÕES INICIAIS:

Nota: Os exercícios devem ser executados na ordem apresentada, pois o nível de dificuldade é crescente.

1. Obtenha o arquivo **lab04.zip** (disponível no ensino aberto);
2. Descompacte esse arquivo no seu diretório de trabalho (workspace do Eclipse);
3. Crie um novo projeto Java no Eclipse. Clicar em File > New > Java Project, manter o lugar padrão e escrever como nome do projeto exatamente o nome da pasta descompactada.
4. Ao final do lab, compacte a pasta raiz do projeto, em um arquivo com extensão .**zip**, e suba no ensino aberto no seu portfólio, disponibilizando-o para os formadores. Suba apenas um arquivo. **Outros formatos não serão avaliados.**
5. Suba um arquivo de texto com a resposta para a questão 1 **dentro** da pasta compactada. Serão aceitos os formatos .**txt** e **pdf**. Coloque seu nome e RA neste arquivo.

Pacote *br.unicamp.ic.mc302.fila*:

1. Abra os arquivos `Lista.java`, `Fila.java` e `ExemploFila.java`. Execute a classe ExemploFila. O resultado mostrado não respeita o comportamento padrão de uma fila (primeiro que entra/primeiro que sai). Por quê? Introduza uma relação de agregação entre as classes Fila e Lista de modo que a política primeiro que entra/primeiro que sai passe a valer. Reexecute ExemploFila. O que acontece?

Pacote *br.unicamp.ic.mc302.veiculos*:

2. Abra o arquivo `Carro.java`. Defina classes correspondendo ao motor do carro (Motor) e ao seu tanque de combustível (TanqueCombustivel) e agregue esses elementos à classe Carro. A classe Motor deve guardar informação indicando se o motor está ou não ligado e a classe TanqueCombustivel deve guardar a quantidade de combustível disponível e a sua capacidade máxima, sendo que a quantidade disponível nunca pode ser maior que a capacidade máxima. Defina uma operação ligar( ) na classe Carro que ativa o motor e gasta parte do combustível do tanque. Defina também operações na classe Carro para verificar se o motor está ou não ligado e qual a quantidade de combustível disponível no tanque.

3. Defina uma classe Caminhao com dois atributos: número de eixos (numEixos) e carga suportada (capacidade). Os valores de numEixos e capacidade devem ser passados para objetos do tipo Caminhao através do construtor da classe. Implemente operações para obter os valores desses dois atributos. Defina uma classe ExemploCaminhao que cria um objeto do tipo caminhão e imprime seu número de eixos e sua capacidade.

4. Caminhões e carros são veículos bastante parecidos, embora não seja completamente preciso afirmar que um caminhão é um subtipo de carro. O que você faria para evitar que o mesmo conjunto de operações esteja repetido tanto na classe Carro quanto na classe Caminhão, tendo em vista que carros e caminhões têm características em comum? Implemente sua solução.

5**.** Como visto em sala de aula, o Padrão de Projeto ***Composite*** permite que uma hierarquia de classes seja tratada como uma única classe. Isto é, ***Composite*** permite que um objeto de uma classe seja constituído de outros objetos semelhantes a ele formando uma hierarquia (semelhantes, significa aqui, classes que implementam um contrato comum).

Seguindo este padrão podemos construir um objeto que seja construído de outros objetos tal que, um ou mais objetos desses podem ser do mesmo tipo do objeto construído. O objeto é construído por objetos que contém uma coleção de outros objetos, os quais contêm uma coleção de outros objetos, e assim sucessivamente. Contudo esses objetos não são quaisquer, suas classes compatilham um contrato comum. Por exemplo, um objeto do tipo Figura:

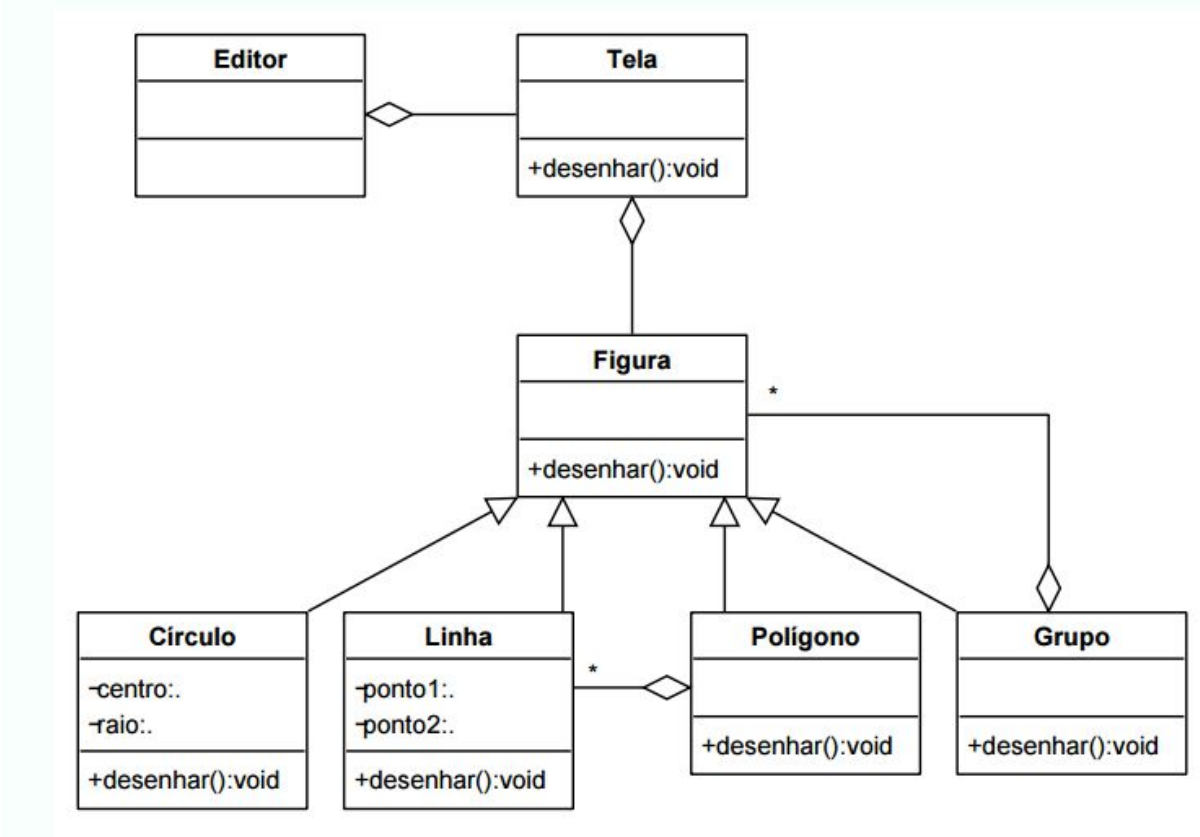
Existem várias figuras, mas todas elas podem ser construídas pela composição de outras figuras. Uma ou mais figuras são primitivas, ou seja, não são construídas a partir de nenhuma outra figura. As outras são apenas composições. O exemplo mais simples é a figura Linha que representa uma semi-reta. Um Triangulo é composto por três objetos da classe Linha. Um quadrado por quatro , etc... Um circulo não é representável por um conjunto finito de linhas, logo precisamos de uma figura primitiva Circulo.

Este padrão pode ser usado sempre que desejarmos construir objetos pela composição recursiva de objetos do mesmo tipo ou queiramos construir uma hierarquia de objetos do mesmo tipo. Este é o padrão por detrás da estrutura de árvore, muito utilizada em computação. A estrutura de arvore tem como primitivas as folhas. Os ramos são os objetos compostos. A árvore é uma composição de ramos, por sua vez compostos de ramos menores, por sua vez compostos de ramos menores e folhas. Sempre que estiver perante uma estrutura deste tipo, ou semelhante, poderá usar o padrão ***Composite***.

Exercício: Implemente o exercício de editor de desenhos geométricos (transparências 62-64 do material de estudos contido no arquivo “cap04-2-v6.pdf” – Herança Simples, Agregação, Associação, Herança Múltipla), reproduzido abaixo, segundo o padrão de projeto ***Composite***. Defina uma classe ExemploEditor que cria um objeto do tipo editor com uma tela com um polígono, uma reta, um círculo e um grupo (contendo ao menos dois elementos). Faça com que o objeto do tipo editor envie para a tela a operação de desenhar.

Descrição do Exercício (transparências 62-64):

Um editor de desenhos geométricos possui uma tela para desenhar. Essa tela pode ser constituída de muitas figuras. Figuras podem ser círculos, linhas, polígonos e grupos. Um grupo consiste de muitas figuras e um polígono é composto por um conjunto de linhas. Quando um cliente pede para que uma tela se desenhe, a tela, por sua vez, pede para cada uma das figuras associadas que se desenhe. Da mesma forma, um grupo pede que todos os seus componentes se desenhem. Crie uma hierarquia de generalização/especialização que classifique essas diferentes figuras geométricas e identifique o comportamento de cada tipo abstrato de dados que você criar, bem como os seus respectivos atributos.



Item Extra:

Uma empresa de ônibus contratou você para criar um programa Java para controlar reservas de lugares em um ônibus. Cada ônibus possui 40 lugares, divididos em 4 fileiras.

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 * 0 0 0 0
0 0 0 0 0 0 0 0 0 0

A poltrona é referida pela fileira e pela coluna, Por exemplo, a poltrona marcada com * é a poltrona da fileira 6 assento B. As fileiras são 1, 2. 3 .. 10, e os assentos são A,B,C,D.

O objetivo do seu programa é ver se a poltrona desejada está ou não ocupada.
Se estiver ocupada, imprima a mensagem ``Poltrona ocupada! Tente outra.'' na tela, mostre a alocação atual do ônibus e peça outra poltrona. Se a poltrona estiver vaga imprima a mensagem ``Reserva efetuada!'' na tela, mostre a alocação atual do ônibus e peça outra poltrona. O programa deve ser executado enquanto existir lugar não reservado ou o índice da poltrona for não negativo.