Exercícios sobre o Paradigma de Análise Orientada a Objetos

Aula 2: 30 mar 2021 - Prof. Me. Argemiro Bevilacqua

Disciplina de Análise Orientada a Objetos 3º período

Nome do Aluno: Caio Augusto Barretta

Estes exercícios são para fixação da teoria apresentada na aula do dia 30 de março. Todas as respostas podem ser encontradas facilmente no texto. Então, primeiro dê uma revisada na teoria que encontrará suas respostas. Responda com suas palavras, depois de entender a teoria. Não é pra fazer *copy/paste*! Estes exercícios vão contar como presença para as aulas dos dias **30 março e 6 de abril**, com a nova metodologia de ensino a distância. O professor vai avaliar, corrigir, comentar e devolver aos alunos. Podem fazer *upload* de arquivos no *Moodle*, contendo as suas respostas. A devolutiva do professor será pelo *Moodle* também.

1. O que é um objeto? Explique com suas próprias palavras e dê exemplos. Considere apenas objetos do mundo real, conforme explicado pelo dicionário Aurélio.

**R: Um objeto é um exemplo concreto de uma classe. Por exemplo: *Classe Cachorro*, um objeto poderia ser meu cachorro, que é da raça *Shih-tzu*, tem *5 anos de idade*, seu nome dela é *Kiara*, e tem outros atributos e comportamentos que se assemelham a classe que deu origem, no entanto tem as especifidades (concretude) do objeto exemplificado.**

**Em outras palavras uma classe pode ser comparada com uma forma, e cada objeto com o material que está sendo moldado pela forma.**

1. Idem, da mesma forma, agora explique, o que é um **objeto computacional**? Imagine que você já criou um no seu programa, como por exemplo, uma conta bancária.

**R: Um objeto computacional é uma representação semelhante ao mundo real, dentro do domínio computacional, ou seja, é um exemplo concreto de uma classe implementada que tem como objetivo resolver algum problema real atrás de computação. Logo o domínio é executado por inputs, processamento e outputs que descrevem algum processo do mundo real, como por exemplo uma conta bancária.**

**Encontraremos, certamente, dados do corretor, sua identificação como por exemplo: Nome, Sobrenome, Endereço, Certidão de pessoa física e afins, dados da sua conta, como por exemplo: Saldo, Tipo de cota, histórico de transações e afins.**

**Nessa perspectiva, um objeto computacional tenta representar o mundo real dentro de um domínio (dados e estruturas de dados) para tentar solucionar algum problema por meio de entradas, processamento e saídas.**

1. Explique com suas palavras, em detalhes, o que significa o conceito de **encapsulamento** de um objeto computacional? Você pode usar uma classe programada por outro programador, mesmo sem conhecer o seu código?

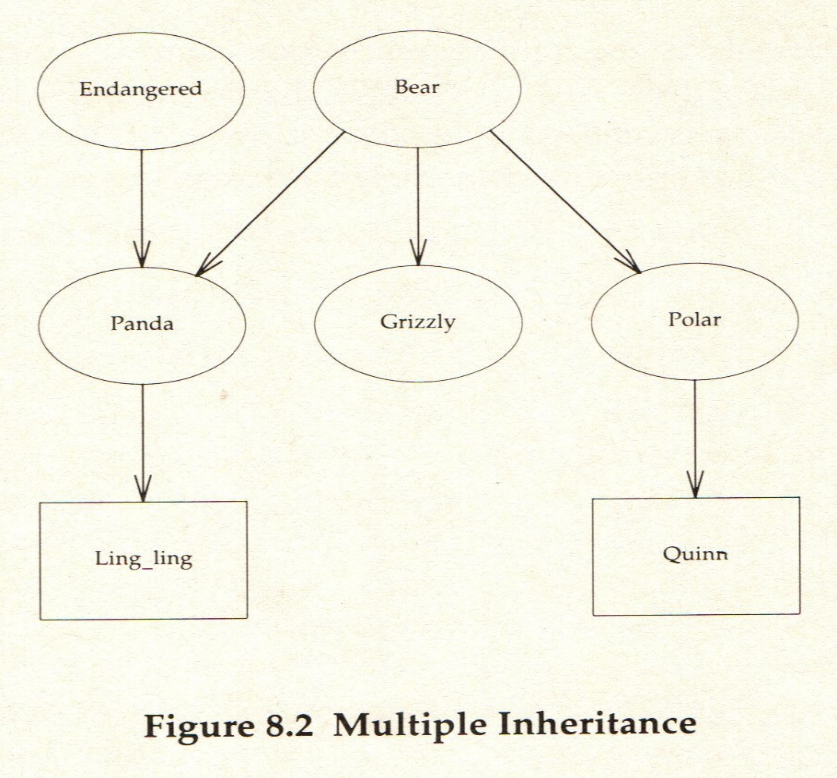
**R: O encapsulamento, como o próprio nome sugere, é o fato de ter uma interface computacional que abstraí a complexidade da implementação do usuário que vai consumi lá.**

**Por exemplo: Alguma API de um SGBD, nós podemos abrir uma transação no banco de dados, executar comandos, efetuar o *commit* ou *rollback* da transação e fechar a conexão, No entanto, ficamos abstraídos do processo que ocorre dentro da API, dos tipos de estruturas de dados e afins, pois essa API está encapsulada, por exemplo, dentro da biblioteca utilizada.**

1. Ainda falando de objetos computacionais genericamente. O que faz um objeto computacional ter comportamentos e ações, para que simule comportamentos e ações de objetos do mundo real? Como se faz isto sendo você o programador?
2. Relacione a primeira coluna com a terceira coluna, pondo um número na coluna do meio:

|  |  |  |
| --- | --- | --- |
| **Tabela de Conceitos em Programação Orientada a Objetos** | | |
| **Conceito** | **número** | **Descrição** |
| 1. Abstração (pensamento de criança) |  | Casa, carro e ser humano: Casa tem porta, carro tem rodas, ser humano tem cabeça, tronco, braços e pernas |
| 1. Classificação (pensamento de criança) |  | As abstrações definem grupos de objetos: tudo que for casa, tudo que for carro, tudo que for ser humano. O que são estes grupos? |
| 1. Instanciação (pensamento de criança |  | Um carro de cor vermelha |
| 1. Atributos ou propriedades de uma classe |  | Características de um objeto, que variam de uma classe para outra, por exemplo, a altura de uma pessoa ou a cor de um objeto carro |
| 1. Método operações ou comportamentos |  | Similar à uma função em C, a maior parte de um código em C++; JAVA, ou qualquer linguagem de programação OO. |
| 1. Visibilidade |  | Define a acessibilidade a um método ou variável, pode ser: Privada, protegida, publica ou pacote |

1. Observe a figura.



**Exercício sobre como desenhar um diagrama de classes.**

Aqui temos um exemplo de herança múltipla, considerando a representação computacional de objetos do mundo real. Temos 3 tipos de ursos: panda chinês, urso pardo norte americano e urso polar. Todos são ursos, então foi criada uma super classe “Bear”. Assim, todas as suas sub classes herdam as características de urso. Temos também representada uma herança múltipla. “Endangered” quer dizer perigoso. As instanciações de objetos são representadas por retângulos e dão os nomes aos animais.  
  
 Esse diagrama foi pego de um livro antigo, de 1991! Então, redesenhe tudo, porque não está no padrão atual do diagrama de classes que foi passado em aula. Ponha alguns atributos na superclasse e defina a visibilidade desses atributos. Lembre-se de que temos de usar a visibilidade “protected”. Use os símbolos “+”, “-“ e “#”, divida os retângulos que representam as classes, olhe no texto da aula.

1. **Conceito de polimorfismo** (estude primeiro o slide da aula)**.** O conceito de polimorfismo é um dos recursos mais poderosos em programação orientada a objetos. Pode um método herdado de uma super classe comportar-se de maneira diferente do que foi herdado? O que acontecerá se você como programador redefinir este método, usando o mesmo nome que foi herdado da superclasse? O que é o conceito de polimorfismo?
2. **Exercício sobre herança, superclasse e subclasse** (estude primeiro no slide da aula). Imagine um ambiente de programação onde temos uma superclasse e uma subclasse. Isto significa que a subclasse vai herdar todos os atributos de sua superclasse. E se você não quiser que a sua subclasse use alguns atributos herdados, o que faria com a proteção dos atributos da superclasse. Dica, use “+”, “-“, ou “#”.
3. Explique o conceito de **Generalização.** Dica: tem a ver com **Herança.**
4. **Desenhe um diagrama de classes.**   
    Quando trabalhamos com M.E.R. temos o conceito de **especialização.** Um exemplo bem intuitivo e muito usado é o exemplo de uma pessoa (superclasse) que se especializa em um funcionário (subclasse) ou em um cliente (outra subclasse). Dica: funcionário tem o atributo salário além de todos os atributos de pessoa. Cliente tem o atributo limite de crédito, além de todos atributos herdades da superclasse pessoa. Então, desenhe tudo isto num diagrama de classes. Indique os atributos.