**PARADIGMAS DE PROGRAMAÇÃO**

EXERCÍCIOS LISTA 09: **PARADIGMAS ORIENTADOS A OBJETOS**

Prof.ª Lucília Ribeiro

Aluno: **Thiago Santos Oliveira**  Matrícula:20141002803464

**RESPOSTA 01:** Uma linguagem orientada a objetos deve fornecer suporte para três recursos chave de linguagem:tipos de dados abstratos, herança e vinculação dinâmica de chamadas a métodos;

*ABSTRAÇÃO DE DADOS: Uma abstração é uma visão ou representação de uma entidade que inclui apenas os atributos mais significativos, agrupando assim, as entidades cujos atributos comuns não precisam ser considerados. Sintaticamente, um tipo de dados abstrato é um invólucro que inclui apenas a representação de dados de um tipo de dados específico e os subprogramas que fornecem as operações para esse tipo. Unidades de programa que usam um tipo de dados abstrato podem declarar variáveis de tal tipo, mesmo que a representação real seja dele ocultada. Um exemplar de um tipo de dados abstratos é chamado de um objeto.*

*HERANÇA: Quando um novo tipo abstrato de dados pode herdar os dados e funcionalidades de algum tipo existente, e também é permitido que ele modifique algumas das entidades e adicione novas, o reuso é amplamente facilitado sem requerer mudanças ao tipo abstrato de dados reutilizado. Os programadores podem começar com um tipo abstrato de dados existente e projetar um descendente modificado dele para atender a um novo requisito do problema.*

*VINCULAÇÃO DINÂMICA: Um propósito para essa vinculação é permitir que os sistemas de software sejam melhor estendidos durante o desenvolvimento e a manutenção. É um tipo de polimorfismo fornecido pela vinculação dinâmica de mensagens às definições de métodos.*

**RESPOSTA 02:** Cada objeto de uma classe tem seu próprio conjunto de variáveis de instância, que armazenam o estado do objeto. A única diferença entre dois objetos da mesma classe é o estado de suas variáveis de instância. Já variáveis de classe pertencem à classe, em vez de ao seu objeto, então existe apenas uma cópia para a classe.

**RESPOSTA 03:**

1. **HERANÇA MÚLTIPLA:** Quando uma classe tem mais de uma classe pai. Os relacionamentos de classes em uma herança múltipla podem ser mostrados em um grafo de derivação.
2. **VARIÁVEL POLIMÓRFICA:** É o tipo da classe base, e esta define ao menos o protocolo de um método que é sobrescrito pelas classes derivadas. Pode referenciar objetos da classe base e de classes descendentes, então a classe do objeto para o qual ela aponta nem sempre pode ser determinada estaticamente.
3. **MÉTODO SOBRESCREVEDOR:** Método que sobrescreve um método da classe pai.

**RESPOSTA 04:**

1. **A exclusividade dos objetos**, onde há a preocupação em projetar um sistema de objetos que absorve todos os outros conceitos de tipo. Todos os tipos são classes.
2. **As subclasses são subtipos?** Uma classe derivada e chamada de um subtipo se tem um relacionamento é-um com sua classe pai. As características de uma subclasse para garantir que ela é um subtipo são: os métodos da subclasse que sobrescrevem métodos da classe pai devem ser compatíveis em relação ao tipo com seus métodos sobrescritos correspondentes.
3. **Verificação de tipos e polimorfismo**. A vinculação de mensagens a métodos enviados por variáveis polimórficas deve ser dinâmica e existem duas formas de verificação de tipos ente uma mensagem e um método de uma linguagem fortemente tipada onde os tipos dos parâmetros e do retorno devem ser verificados rem relação ao tipo esperado da mensagem.
4. **Herança simples e múltipla -** O propósito da herança múltipla é permitir que uma nova classe herde de duas ou mais classes. Logo a razão para a não inclusão de herança múltiplas está entre duas categorias: complexidade e eficiência.
5. **Alocação e liberação de objetos –** Existem duas questões de projeto relacionados à alocação e à liberação de objetos. A primeira é o local onde os objetos são alocados. Se eles se comportam como tipos de dados abstratos, talvez possam ser alocados de qualquer lugar. Isso significa que podem ser alocados da pilha de tempo de execução ou explicitamente criados no monte com um operador ou função, como new. A segunda questão se preocupa com casos nos quais os objetos são alocados do monte. A questão é se a liberação é implícita, explícita ou de ambos os tipos.
6. **Vinculação estática e dinâmica –** Sabendo que a vinculação dinâmica de mensagens a métodos em uma hierarquia de herança é parte essencial da programação orientada a objetos. Também sabe-se que vinculações estáticas são mais rápidas. Logo a questão está na alternativa ao usuário de especificar se uma vinculação deve ser estática ou dinâmica, em determinadas situações.
7. **Classes aninhadas –** Uma das motivações primárias para classes aninhadas é o ocultamento de informação. A classe na qual a nova está aninhada é chamada de classe aninhadora. As questões de projeto mais óbvias associadas com aninhamento de classes são relacionadas à visibilidade.
8. **Inicialização de objetos –** Diz respeito a se e como os objetos são inicializados para valores ao serem criados. Se são inicializados manualmente ou por meio de algum mecanismo implícito.

**RESPOSTA 05:** Em C++, um método deve ser definido como virtual para permitir vinculação dinâmica. Em Java, todas as chamadas a métodos são dinamicamente vinculadas, a menos que o método chamado tenha sido declarado como **final**, de modo que não pode ser sobrescrito e todas as vinculações são estáticas.

K