Programação de Computadores II - Prof. Senne
Aula 4: Hierarquia de Classes - Perguntas & Respostas

1. Além da herança, em uma subclasse é possível mudar a assinatura de um método de uma superclasse?

Métodos da superclasse podem ser redefinidos nas subclasses, mas para isto é necessário manter a assinatura do método. Se a assinatura do método for alterada, não se trata de redefinição de método herdado e sim a definição (na subclasse) de um novo método.

2. Na formação dos métodos da subclasse posso utilizar um objeto (this.__) da superclasse?

Sim, um método da subclasse pode fazer referência a um objeto da superclasse. Mas a palavra "this" refere-se a um objeto da própria classe em que a palavra é usada. Para fazer referência a um objeto da superclasse deve-se usar a palavra "super" (super.__).

3. A classe principal de um programa pode ter subclasses? Na redefinição de métodos, quais são os tipos de métodos que podem ser redefinidos? Uma subclasse pode herdar apenas métodos públicos?

*Sim, a classe principal pode ter subclasses. Mas métodos estáticos, como o método **main()**, não não herdados. Qualquer método herdado pode ser redefinido. Uma subclasse irá herdar apenas os métodos públicos ("public") e os métodos protegidos ("protected"). Métodos privados ("private") não são herdados.*

4. Se a superclasse tiver dois construtores, um com parâmetros e outro sem, como ficaria a implementação do construtor da subclasse?

No construtor da subclasse deve-se chamar o construtor da superclasse como super() (caso se queira chamar o construtor sem parâmetros) ou como super(p1, p2, ..., pn) (caso se queira chamar o construtor com parâmetros, onde p1, p2, ..., pn são valores dos parâmetros).

5. Quando vai fazer o construtor da subclasse precisa fazer a chamada da superclasse (implícito ou explícito)? Quando for explícito, como fazer com os parâmetros?

Sim, todo construtor de subclasse precisa chamar (implícita ou explicitamente) o construtor da superclasse. A chamada implícita é sempre para o construtor sem parâmetros (caso exista na superclasse). Qualquer chamada ao construtor com parâmetros deve ser feita explicitamente. Para isso, usar super(p1, p2, ..., pn), onde p1, p2, ..., pn são os valores dos parâmetros.

6. Pode haver uma subclasse de outra subclasse?

Sim. Somente classes definidas como "final" não podem ter subclasses.

7. É possível mudar o modificador de um campo na superclasse (deixar de ser privado) para que ele possa ser herdado pelas subclasses?

Sim. Pode-se usar o modificador "protected". Campos protegidos são herdados pelas subclasses.

8. Uma classe pode ter infinitas subclasses? Uma subclasse pode ter outra subclasse?

Uma classe pode ter tantas subclasses quantas forem necessárias. Sim, uma subclasse pode ter subclasses. Somente classes definidas como "final" não podem ter subclasses.

9. Posso alterar um valor de "Automovel" herdado de "Veiculo", sem alterar seu valor na classe original (Veículo)?

Não sei se entendi a pergunta. Numa hierarquia de classes apenas os métodos são herdados, pois, em geral, os métodos são públicos e os campos são privados. Os métodos herdados podem ser redefinidos na subclasse, mas isto não altera o método da superclasse.

10. Uma subclasse, cuja mãe é uma subclasse, herda o método redefinido ou o método original da mãe de sua superclasse?

Toda subclasse herda os métodos (públicos) de sua superclasse imediata. No exemplo que você apresenta será herdado o método já redefinido e não o método original.

11. Como fazer a leitura simultânea de dois dados inseridos pelo usuário? É possível fazê-lo? Há um limite de subclasses? Se eu criei o objeto Veiculo v = new Veiculo(x,y), passei para v = new Automovel() e quero, por alguma razão, voltar para v = new Veiculo(x,y), como fazer?

Não é possível a leitura simultânea de dois dados. Cada leitura refere-se a apenas um dado. Não há limite para o número de subclasses. Ao fazer v = new Automovel() perde-se a referência para o objeto criado em Veiculo v = new Veiculo(x,y). Uma possibilidade é definir uma outra variável Veiculo w = v; para fazer referência a este objeto. Dessa forma, depois de fazer v = new Automovel(), se quiser voltar ao objeto original, basta fazer: v = w.

12. É possível ter mais de uma classe principal, ou seja, mais de uma classe com o método main()?

Sim, um projeto pode ter várias classes executáveis. Neste caso, será necessário indicar qual dessas classes deverá ser executada.

13. Como ocorre a transformação de um double, como preco, em uma string?

O método toString(), disponível em todas as classes pois é herdado da classe Object, realiza a transformação para string.

14. Os métodos protegidos da superclasse são herdados pelas subclasses? É possível uma subclasse ser superclasse de outra? Serão herdados todos os métodos? Inclusive da primeira para a última?

Sim. Os métodos herdados pelas subclasses são os métodos públicos ("public") e os métodos protegidos ("protected"). Sim, numa hierarquia de classes, uma subclasse pode ter subclasses e assim uma subclasse pode ser superclasse de uma outra subclasse. Os métodos herdados são sempre os métodos (públicos ou protegidos) da superclasse imediata.

15. No slide da aula diz que podemos usar campos como "protected". Quando usaríamos os campos dessa forma, visto que podemos usar os "gets" para saber o valor dos campos?

Sim, a linguagem Java permite definir os campos de uma classe como protegidos ("protected") para serem herdados pelas suas subclasses. Mas isso é desnecessário, como você aponta, pois pode-se usar os getters para acessar o valor dos campos da superclasse.

16. Uma subclasse pode ter mais de uma superclasse?

Não. Em Java, uma subclasse tem apenas uma superclasse.

17. Não entendemos muito bem a ideia do polimorfismo. O que é a assinatura do método?

O polimorfismo existe como decorrência da hierarquia de classes porque uma variável de referência para um objeto de uma superclasse pode ser usado também como referência para um objeto da subclasse. Por exemplo: Automovel é subclasse de Veiculo. Então podemos fazer (a) Veiculo v = new Veiculo() e depois fazer (b) v = new Automovel(). Note que em (a), a variável v faz referência a um objeto da classe Veiculo e em (b) a mesma variável v faz referência a um objeto da classe Automovel. Assim, um método chamado pela variável v pode

ter mais de uma "forma de comportamento": pode se comportar como um método da classe Veiculo ou como um método da classe Automovel. Se este método foi redefinido na subclasse, essas duas "formas de comportamento" serão diferentes. O nome "polimorfismo" vem desta possibilidade de apresentar diferentes "formas de comportamento". A assinatura de um método é sua identificação, formada pelo seu nome e pelos tipos de sua lista de parâmetros. Por exemplo, um método definido como:

public double calcular(double x, int y)

tem a assinatura: ***calcular_double_int****.*
Um método definido como:

public double calcular(int a, double b),

embora tenha o mesmo nome, tem uma assinatura diferente: ***calcular_int_double*** *e, portanto, trata-se de um outro método.*

18. Como funciona o operador **instanceof**?

O operador ***instanceof*** *é usado para testar se uma variável faz referência a uma classe ou a uma de sua subclasses (ver resposta da* ***Questão 17****). Por exemplo:*

if (v ***instanceof*** *Automovel) ...*

19. O operador **instanceof** permite o acesso aos campos de uma classe?

Não. Veja resposta da ***Questão 18****. Em geral, os campos de uma classe são privados e, portanto, são acessíveis somente dentro da classe.*

20. É possível uma superclasse ser uma subclasse de uma das suas subclasses? Um ciclo hierárquico.

Não. Uma hierarquia de classes na linguagem Java não pode ter ciclos.

21. Qual a necessidade de utilizar a palavra "final" para declarar uma classe?

Numa hierarquia de classes, a palavra "final" é usada para indicar que uma classe não pode ter subclasses. A palavra "final" também pode ser usada em outros contextos. Por exemplo, pode ser usada para indicar que um método não pode ser redefinido, ou que uma variável não pode ser redefinida.

22. Se eu tenho uma subclasse de uma subclasse, como posso chamar o construtor da primeira subclasse?

Não. Usando a palavra "super" pode-se acessar os métodos públicos da superclasse. Por exemplo: super() refere-se ao construtor da superclasse e super.calcular() refere-se ao método calcular() da superclasse. ***Não é possível*** *"subir" mais de um nível na hierarquia. Por exemplo: super.super() ou super.super.calcular()* ***não são aceitos*** *em Java.*