Programação de Computadores II - Prof. Senne
Aula 5: Classes Abstratas e Interfaces - Perguntas & Respostas

1. Uma superclasse é abstrata. Quando vou declarar uma subclasse sua, preciso declarar a subclasse também como abstrata?

Sim, se a subclasse também for abstrata, ou seja, se a subclasse também contém algum método abstrato.

2. Para utilizar uma interface utiliza-se a palavra **implements** e para as demais classes abstratas a palavra **extends**?

*Sim, para utilizar uma interface, uma classe deve implementar os métodos (abstratos) dessa interface. Por isso utiliza-se a palavra **implements**. A palavra **extends** é usada para definir uma hierarquia. Pode-se ter hierarquia de classes concretas, hierarquia de classes concretas e abstratas, hierarquia de classes abstratas e mesmo, hierarquia de interfaces. Consulte na Java API a interface **Collection** e verifique que essa interface faz parte de uma hierarquia de interfaces.*

3. Por que criar uma classe abstrata e obrigar a criar uma subclasse, sendo que poderia ser feito tudo na primeira classe? Existe uma facilidade do Java com classes abstratas?
4. Qual a vantagem de usar uma classe abstrata junto com uma subclasse ao invés de usar apenas uma classe concreta?

*As questões 3 e 4 são semelhantes. O uso de hierarquia de classes (concretas ou abstratas) tem o propósito de facilitar a construção de classes, por meio do reuso de código. Se você quer criar apenas uma subclasse concreta a partir de uma classe abstrata, realmente não tem sentido criar a classe abstrata. Uma hierarquia só tem sentido se você quiser ter a possibilidade de criar várias subclasses a partir de uma superclasse (concreta ou abstrata). Numa hierarquia de classes, uma subclasse pode ou não redefinir um método da superclasse. Mas se a superclasse for abstrata, a subclasse tem que **necessariamente** redefinir os métodos abstratos da superclasse, pois do contrário, a subclasse também será abstrata. Portanto, a criação de classes abstratas é um recurso adicional que a linguagem Java oferece para **obrigar** a redefinição de métodos nas subclasses.*

5. Para que serve o construtor de uma classe abstrata?
6. Por que existe um construtor em uma classe abstrata?

As questões 5 e 6 são semelhantes. Embora não seja possível criar objetos de uma classe abstrata, em uma hierarquia de classes o construtor de uma subclasse chama (explícita ou implicitamente) o construtor da superclasse. Como uma classe abstrata deve, necessariamente, ter subclasses, a classe abstrata precisa ter um construtor.

7. Como a interface não necessita de construtor, ela não é subclasse de Object? Pode-se criar um objeto de uma classe abstrata?

*Correto, interfaces não fazem parte da hierarquia de classes e, portanto, não são subclasses da classe Object. Mas é possível haver hierarquia de interfaces. Veja, por exemplo, na **Java API** a interface **Collection**: public interface Collection extends Iterable. Portanto, a interface Collection é uma subinterface de Iterable ou, em outras palavras, Iterable é **"uma superinterface"** de Collection (note que classes têm apenas **uma** superclasse, mas interfaces podem ter **várias** superinterfaces. Portanto na frase acima, tem sentido escrever "uma superinterface" e não "a superinterface". Não é possível criar objetos de uma classe abstrata (e nem de interfaces, é claro).*

8. Por que ao se criar uma classe que implementa a interface ActionListener aparece a anotação @Override, dentre outras sugestões?

*A anotação **@Override** é exigida pelo compilador para tornar explícita a intenção do programador em redefinir um método herdado de uma superclasse ou em implementar um método exigido por uma intrface.*

9. Considerando a superclasse abstrata A e sua subclasse B, ainda pode-se criar uma subclasse C herdeira da classe B? Caso seja possível, o método abstrato da classe A é herdado pela C ou C herda o método implementado na classe B, como deveria acontecer caso todas as classes fossem concretas?

Sim, classes abstratas também podem ser organizadas numa hierarquia. Uma classe da hierarquia sempre herda métodos públicos de sua superclasse. Assim, a classe C vai herdar métodos (abstratos ou concretos) da classe B.

10. Por que as interfaces só apresentam métodos abstratos?

Uma interface é uma espécie de "contrato": quem assina se responsabiliza por implementar alguns métodos (cumprir o contrato). Uma interface pode definir vários métodos, mas nunca conter implementação desses métodos. A interface só diz "o que" o método deve fazer, mas não "como" o método faz. Os detalhes de "como" o método vai fazer "o que" deve ser feito será definido em "uma" implementação da interface. Observar a frase: "uma" implementação da interface. Isso tem sentido, pois uma interface pode ter diversas implementações diferentes.

11. Um método abstrato em uma subclasse torna a subclasse abstrata, mesmo que a superclasse não tenha nenhum método abstrato?

Sim, a presença de um método abstrato em qualquer classe (seja subclasse ou superclasse) torna essa classe uma classe abstrata.

12. Para programas diferentes, pode-se utilizar classes abstratas de um programa em outro programa?

Sim, desde que as classes dos programas estejam todas no mesmo pacote.

13. Como transformar um programa escrito em Java em um aplicativo executável?

Um aplicativo Java executável é um arquivo com extensão ".jar", que é criado por meio da opção "Build Main Project" do menu "Run" da ferramenta Netbeans.

14. Pode-se transformar uma classe abstrata em concreta ou vice-versa?

Para transformar uma classe concreta em classe abstrata basta incluir nesta classe um método abstrato. Uma classe é abstrata porque contém um ou mais métodos abstratos. Assim, uma classe abstrata se transforma em classe concreta se seus métodos abstratos forem excluídos da classe. Mas, normalmente, constrói-se uma classe concreta a partir de uma classe abstrata por meio de uma hierarquia de classes, em que a classe abstrata é a superclasse e a classe concreta é uma subclasse, na qual os métodos abstratos são implementados.

15. Como funcionam as definições de constantes em uma interface e como podem ser usadas?

*Além de definir métodos abstratos, uma interface pode também definir constantes, ou seja, campos declarados como "public final". Por exemplo: **public final int MAX = 1000;** Neste caso, qualquer classe que "implementa" essa interface pode usar a constante MAX.*