Interfaces e Herança

Prof. Rafael Guterres Jeffman
rafael.jeffman@gmail.com

Interfaces

- Em diversos momentos do desenvolvimento de software, é muito importante que a iteração entre os componentes seja clara e bem definida.

- Em diversos algoritmos, não é o tipo de dado, ou a classe de um objeto, que interessa ao algoritmo, mas o seu comportamento.

- Interfaces são utilizadas para definir o comportamento desejado de objetos, independente a que classe eles pertencem.

Interfaces em Java

- Definem o comportamento que uma classe deve possuir, mas não como deve ser implementado.

- Interfaces podem ser implementadas em classes, ou estendidas por novas interfaces.

- Uma interface só pode conter métodos abstratos e declaração de constantes.

- Todos os elementos de uma interface são públicos.

Implementando Interfaces

- Uma classe que implementa uma interface irá gerar objetos que podem ser vistos como sendo do tipo da própria interface.

- Uma classe que implementa uma interface deve implementar todos os métodos não-estáticos definidos por essa interface.

Exemplo de Interface

```
public interface ContatoView {

    public void printContato();

    public String readNome();

    public String readEndereco();

    public String readTelefone

}
```

Herança

- Herança é uma relação entre classes do tipo “**é um**”
- É a relação mais forte existente entre duas classes.
- Utilizamos herança quando queremos especializar o comportamento de uma classe, mas os objetos desta classe ainda podem ser vistos como objetos da classe base.
- Em POO, existe herança simples e herança múltipla.

Conceitos em Herança

- Classe Base: a classe da qual se “herda” os atributos e comportamentos.

- Classe Derivada: a classe que é implementada a partir de uma classe base.

- Sobrescrita de Métodos: métodos de instância que são implementados na classe derivada e existem na classe base.

- Ocultação de Métodos: métodos de classe que são implementados na classe base e na derivada.

Visibilidade de Membros

- public: acesso ao membro por qualquer classe do sistema.

- private: acesso ao membro apenas pela própria classe.

- protected: acesso ao membro por qualquer classe pertencente a mesma hierarquia.

Polimorfismo

- É definido como a variação de **tipos** para uma mesma interface.
- Existem três tipos de polimorfismo:
  - Polimorfismo *ad hoc*: utiliza sobrecarga de métodos.
  - Polimorfismo Paramétrico: também conhecido com “programação genérica”.
  - Polimorfismo por Sub-tipo: polimorfismo definido por herança, normalmente chamado apensa de “polimorfismo”.
- Java implementa os três tipos de polimorfismo, com algumas limitações no caso do Polimorfismo Paramétrico.

Tipo de Dados
Estático e Dinâmico

- Tipo de Dado Estático
    - é o tipo de dado que representa o objeto do ponto de vista do código. Define as mensagens e atributos acessíveis.
- Tipo de Dado Dinâmico
    - é o tipo de dado que representa o objeto em tempo de execução, e define o comportamento do objeto.

Herança em Java

- Java implementa apenas herança simples, evitando o problema de *múltipla herança de estado.*

- Para definir a herança de uma classe, utiliza-se a palavra reservada **extends**.

- Todos os membros públicos e protegidos de uma classe são herdados pela classe derivada.

- Construtores não são herdados.

- Toda classe em Java deriva de uma classe. Se nenhuma classe é explicitamente utilizada, a classe base será a classe **Object**, logo, todo objeto em Java também é um Object.

Super

- Para acessar membros de uma classe base, na classe derivada, utiliza-se a palavra reservada **super**.

- Sempre que se utiliza “super”, uma mensagem é enviada à classe base.

- Deve-se utilizar “super” **sempre** em um construtor não-padrão, mas deve ser o primeiro comando do método.

Evitando Herança

- Em Java, é possível evitar herança utilizando a palavra reservada **final**.

- Um método **final** não pode ser sobrescrito nas classes derivadas.

- Uma classe **final** não pode ser derivada (Ex.: String)

- Uma classe pode ter um método **final** e ainda assim ser derivada.

Classes Abstratas

- Métodos abstratos são métodos que não possuem implementação. São utilizados para definir uma interface de um comportamento, mas não o comportamento em si.

- Uma classe que possua um método abstrato será uma classe abstrata.

- Uma classe abstrata não pode ser instanciada.

instanceof

- O operador **instanceof** testa um objeto em tempo de execução para verificar se o objeto é de uma classe específica.

- O resultado de **instanceof** é um valor booleano.

- Se a classe utilizada para o teste for uma classe base do objeto, o resultado será verdadeiro.

Java 8

- Lançado no dia 18/03/2014.

- Inclui novas alterações na linguagem Java (métodos default e static em interfaces)

- Seu uso ainda **não** é recomendado!

Interfaces

- Uma interface pode conter métodos abstratos, métodos *default*, métodos estáticos, e atributos constantes.

Métodos *Default*

- São métodos não-abstratos de uma interface.
- Permitem modificar uma interface existente sem prejudicar código já existente.
- Exemplo:

```
default int getRandomInteger (int min, max) {
    return (int)(Math.random() * (max - min)) + min;
}
```