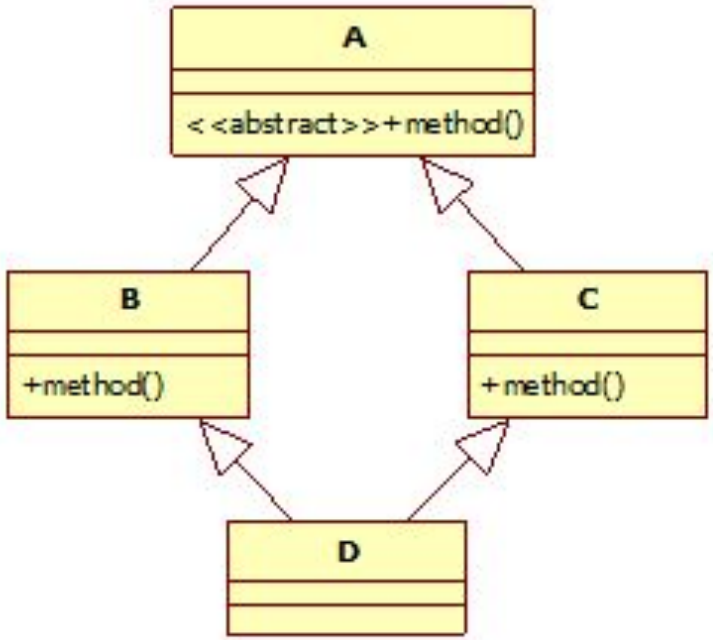
Slides 13: Interfaces (herança múltipla)

Baseado nos slides oficiais do livro Java – Como programar – Deitel e Deitel – 10ª edição

Introdução

- Conceito
  - Herança múltipla = quando uma classe herda simultaneamente de mais de uma classe
- Restrições
  - Em Java (e outras linguagens) não há herança múltipla, isto é, cada classe só pode ter uma superclasse direta
  - Evitar problemas como a herança diamante



Interfaces

- Conceito
  - Representam a definição de um conjunto de métodos que devem ser implementados por outras classes.
  - São semelhantes a classes abstratas onde todos os métodos são abstratos.
  - Definem o que deve ser feito e não como fazer
  - Usando o design por contrato a interface representa o mecanismo de interação entre o contratado (responsável por implementar) e o contratante (que vai usar)
  - Numa interface pode haver:
    - Atributos, que são implicitamente public, static e final
    - Métodos, que são implicitamente public e abstract
  - Não há construtores porque interfaces não são instanciáveis

Interfaces e herança múltipla

- Conceito
  - As interfaces tornam possível implementar o conceito de herança múltipla em Java
  - Uma classe só pode herdar de uma superclasse mas podem implementar (ou realizar) quantas interfaces quiser
    - Assim como é feito com os métodos abstratos na herança, as classes que implementarem uma interface devem definir os seus métodos obrigatoriamente
    - Os nomes das interfaces geralmente terminam em "avel" (em português) ou "able" (em inglês) e representam a característica que as classes devem implementar
  - Ex:Comparable, Serializable, Runnable, ActionListener

Criando interfaces

- As interfaces são criadas de forma semelhante às classes, mas usando a palavra reservada interface ao invés de class
- Dentro da interfaces são permitidas apenas declarações de atributos e métodos vazios
  - public interface Pagavel{
    double getSalario( );
    }

- Há a possibilidade inclusive de fazer herança entre as interfaces, usando extends
  - public interface Previdenciavel extends Pagavel{
    double getDescontoPrevidencia( );

Usando interfaces

- Para especificar que uma classe implementa os métodos de uma interface, usamos a palavra reservada implements em sua declaração
- Caso a classe implemente mais de uma interface, essas devem ser separadas por vírgula
  - public class Aluno **implements** Serializable { … }
  - public class Janela extends JFrame **implements** ActionListener, MouseListener, KeyListener { … }
  - public class Monitor extends Aluno implements Pagavel { … }

Exemplo 1

Refatorar o exemplo das classes de TipoResultado do EAFC, criando duas interfaces: Pontuavel que tem métodos para obter os pontos de cada lutador, e Interrompível que indica o motivo da interrupção da luta. Crie uma subinterface Golpeavel que herda de Interrompivel para lutas que foram interrompidas por um golpe.

Interface Comparable

- Popular na linguagem Java, é usada para indicar que objetos de uma classe são comparáveis entre si
- Obriga a implementar o método int compareTo (Object), em que o resultado deve retornar um inteiro:
  - >0 se o objeto é maior que o parâmetro;
  - =0 se são iguais
  - <0 se o objeto é menor que o parâmetro

Exemplo 2

Criar as interfaces Pagavel e Previdenciavel conforme mostrado. A partir disso, refatorar o exemplo com Aluno e Funcionario onde Pessoa (superclasse de Aluno) implementa a interface Comparable que compara os nomes. O Funcionario implementará a interface Previdenciavel e implementará seus métodos. Deve-se criar uma classe Monitor que implementará a interface Pagavel. Criar um main com todas essas classes e mostrá-las ordenadas..

Métodos default

- A partir da versão 8 do Java passou a ser possível implementar métodos nas interfaces
- Para isso, usa-se a palavra default antes do tipo de retorno, e então coloca-se o corpo da função
- Nesse caso o método default funciona como um método não abstrato de uma classe abstrata
- Dessa forma não é necessário sobrescreve-lo

```
public interface Previdenciavel extends Pagavel
    double getSalario ( );
    double getDescontoPrevidenca( );
    default double getDescontoImpostoDeRenda ( ){ ... }
```

Interfaces e polimorfismo

- Também é possível realizar a vinculação dinâmica de tipos com interfaces
  - Ou seja, é possível declarar um objeto do tipo da interface, e atribuir a uma instancia de um objeto que a implementa
  - public class Aluno implements Comparable { … }

  - Comparable c = new Aluno (...)

Downcasting

- Chamamos de downcasting, quando queremos fazer o contrário da conversrão do polimorfismo
  - Uma classe que foi atribuída a uma referência da superclasse ser convertida par subclasse
- Esse típo de conversão é forçada e caso não seja possível irá ger um erro de ClassCastException
  - Vitoria v; …
  - VitoriaPorNocaute n = (VitoriaPorNocaute) v;

Operador instanceof

- O operador instanceof retorna um resultado verdadeiro ou falso, indicando se um objeto é de uma classe que
  - Ou herda de uma determinada classe
  - Implementa uma interface específica

Aluno a; …

Comparable sala [ ] ; …

if (sala[0] instanceof Aluno{...}

Exercícios

Refatorar as classes do exemplo das lutas para definir as classes e métodos abstratos plausíveis. Criar um cartel de lutas usando polimorfismo.