Linguagem de Programação I

Prof. Edson Kaneshima

Controle de Acesso e Polimorfismo

Aula 05

Controle de Acesso

- Os modificadores de acesso public, private, protected e default determinam quais atributos e métodos de uma classe são visíveis a outras classes.
- Acesso default – atributos e métodos declarados sem o modificador de acesso. O atributo pode ser lido ou alterado por qualquer outra classe do mesmo pacote. O método pode ser chamado por qualquer outra classe do mesmo pacote.

  Exemplos:

  ```
  String nome;

  String getNome( ){

          return nome;

  }
  ```

Controle de Acesso

- Acesso privado – evita que um atributo ou método seja usado por outra classe.

  Exemplos:

  ```
  private String nome;

  private double calculaSaldo( ){
          return saldo + limite;
  }
  ```

Controle de Acesso

- Acesso público – atributo ou método disponível para qualquer outra classe.

  Exemplos:

  ```
  public String nome;

  public String getNome( ){
        return nome;
  }
  ```

Controle de Acesso

- Acesso protegido – atributo ou método disponível para subclasse de uma classe e também para outras classes do mesmo pacote.

  Exemplos:

  ```
  protected String nome;

  protected double calculaSaldo( ){
          return saldo + limite;
  }
  ```

Controle de Acesso

Visibilidade	public	protected	default	private
Da mesma classe	Sim	Sim	Sim	Sim
De qualquer classe no mesmo pacote	Sim	Sim	Sim	Não
De qualquer classe fora do pacote	Sim	Não	Não	Não
De uma subclasse no mesmo pacote	Sim	Sim	Sim	Não
De uma subclasse fora do mesmo pacote	Sim	Sim	Não	Não

Polimorfismo

- Uma regra simples permite reconhecer se a herança é ou não o código adequado para seus dados. A regra “é-um” diz que todo objeto da subclasse é um objeto da superclasse.

- Ex.: todo gerente é um empregado. Assim, faz sentido que a classe Gerente seja uma subclasse de Empregado. Naturalmente, o contrário não é verdadeiro – nem todo empregado é um gerente.

- Uma outra maneira de formular a regra “é-um” é o princípio da substituição. Esse princípio determina que você pode usar um objeto de uma subclasse sempre que o programa espera um objeto da superclasse.

Polimorfismo

- Exemplo: pode-se atribuir um objeto de subclasse a uma variável de superclasse.

  Empregado e;

  e = new Empregado ( ); //espera-se um objeto Empregado

  e = new Gerente ( ); //OK, Gerente também pode ser usado

- Em Java as variáveis de objetos são **polimórficas.** Uma variável do tipo Empregado pode se referir a um objeto do tipo Empregado ou a um objeto de qualquer subclasse da classe Empregado, como por exemplo: Gerente, Executivo, Secretária.

Polimorfismo

```
Gerente chefe = new Gerente( );

Empregado [ ] equipe = new Empregado[3];

equipe[0] = chefe;
```

- Neste caso, as variáveis equipe[0] e chefe referem-se ao mesmo objeto. No entanto, equipe[0] somente é considerada como um objeto Empregado pelo compilador.
- Isso significa que pode-se chamar:

```
chefe.setBonus(5000);    //OK
```

mas não

```
equipe[0].setBonus(5000);    //Erro
```

Polimorfismo

- O tipo declarado de equipe[0]é Empregado, e o método setBonus não é um método da classe Empregado.

- Entretanto, não se pode atribuir uma referência de superclasse a uma variável de subclasse. Por exemplo, não é permitido executar a atribuição:

  Gerente g = equipe[i]; //Erro

- A razão é clara. Nem todos os empregados são gerentes. Se essa atribuição tivesse sucesso e passasse a se referir a um objeto Empregado que não é um gerente, então seria possível posteriormente invocar g.setBonus(100) e aconteceria um erro de execução.

Polimorfismo

```
public class TestaGerente {
    public static void main (String args[]) {
        Gerente chefe = new Gerente ("Luiz", 8000, 500);
        Empregado[ ] equipe = new Empregado[3];
        equipe[0] = chefe;
        equipe[1] = new Empregado("Fatima", 5000);
        equipe[2] = new Empregado("Geraldo", 4000);
        for (int i = 0; i < equipe.length; i++) {
            System.out.println("nome = " + equipe[i].getNome( ) +
                               ", salario = " + equipe[i].getSalario( ) +
                               " - "  + equipe[i].getClass().toString( ));
        }
    }
}
```