Programação Orientada a Objetos

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

Sumário

1. Operador static;
2. Operador final;
3. Agregação de classes;
4. Composição de classes;
5. Classe Object;
6. Tratamento de exceção;
7. Classe JOptionPane.

Operador static

- Quando declaramos um método estático (**static**) em uma classe, ele pode ser acessado em outra classe sem a necessidade de um objeto;
- Uma classe não pode ser configurada com **static**;
- Nós já trabalhamos com métodos estáticos:
  - double r1 = Math.sqrt(9);
  - double r2 = Math.pow(x, 2);
- A mesma regra se aplica aos atributos também:
  - Math.PI;

Operador static

Exemplo:

```
public class Util {
  public static void separador(int n,
String s) {
    String sep = "";
    for(int i = 0; i < n ; i++) {
      sep += s;
    }
    System.out.prinln( sep );
  }
}
```

```
public class Programa {
  public static void main(String args[]) {
    System.out.print("Maria");
    Util.separador(10, "#");
    System.out.println("João");
    Util.separador(10, "-");
  }
}
```

Operador final

- Uma classe configurada com **final** não pode ser herdada:

```
public final class Animal {
   ...
}
```

```
public class Cachorro extends Animal {
   ...
}
```

Gera erro aqui:
"Cannot inherit from final Animal".
"Não pode herdar de Animal final".

Operador final

- Um método **final** não pode ser sobrescrito na subclasse:

```
public class Animal {
   public final void correr() {
      ...
   }
}
```

```
public class Cachorro extends Animal {
   @Override
   public void correr() {
      ...
   }
}
```

Gera erro aqui:
"Overriden method is final".
"O método sobrescrito é final".

Operador final

- Um atributo **final** não pode ter seu valor modificado, ou seja, é uma constante.

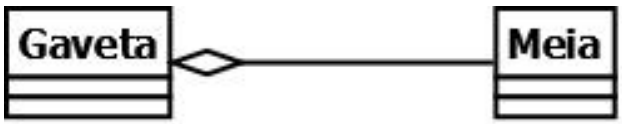
```
public class Animal {
   public static final int codigo = 100;
}
```

```
public class Cachorro extends Animal {
  public Cachorro() {
    Animal.codigo = 200;
  }
}
```

Gera erro aqui:
"Cannot assign a value to final variable".
"Não pode atribuir um valor a uma variável final".

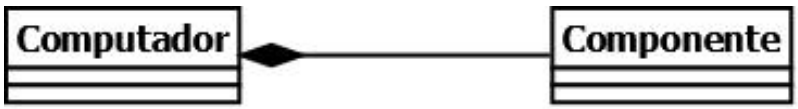
Agregação de classes

- Ocorre quando uma classe não depende de outra(s) para existir;
- Exemplo: Gaveta e Meia
- Uma Gaveta pode conter Meias, mas a Gaveta não é feita de Meias. Ou seja, mesmo sem Meias a Gaveta ainda existirá



Composição de classes

- Ocorre quando uma classe depende de outra(s) para existir;
- Exemplo: Computador e Componente
- Um Computador é formado por seus componentes, como por exemplo placa-mãe, gabinete, hd, memória, placa de vídeo, etc. Sem todas essas peças não existe o Computador.



Classe Object

- Object é a raiz da hierarquia de classes do Java, a superclasse de todas as classes, direta ou indiretamente.
- Possuem os métodos:
  - equals: comparar objetos;
  - toString: representar o objeto como texto;
  - hashCode: número que identifica suas posições em coleções baseadas em hash;
  - getClass: a classe que o objeto faz parte.
  - E outros...

Classe Object

Exemplo:

```
public class Cliente {
  private String nome;
  public String getNome() {
    return this.nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
}
```

```
public class Programa {
  public static void main(String args[]) {
    Cliente c = new Cliente();
    System.out.println( c.getClass() );
    System.out.println( c.hashCode() );
    c.setNome("Goku");
    System.out.println( c.getNome() );
  }
}
```

Tratamento de exceção

- É uma forma eficiente de tratar erros;
- Tenta fazer algumas operações (**try**).
- Se der erro em pelo menos uma delas, automaticamente, esse erro será interceptado e tratado, sem mostrar mensagens enigmáticas para o usuário (**catch**);
- Opcionalmente, depois do processamento, pode executar um código específico independente se deu certo ou errado (**finally**).

Tratamento de exceção

- **Modelo do try-catch-finally:**

```
try {
  // operações
} catch( Exception e ) {
  // tratamento do erro
} finally {
  // executa independente se deu certo ou deu erro
}
```

Classe JOptionPane

- Serve para criar uma mensagem personalizada dentro de uma pequena janela;
- Pode ser para mostrar um texto ou fazer entrada de dados;
- Pode mostrar janela de confirmação também, com as opções de Sim, Não, Cancelar.

Classe JOptionPane

Exemplos:

ENTRADA

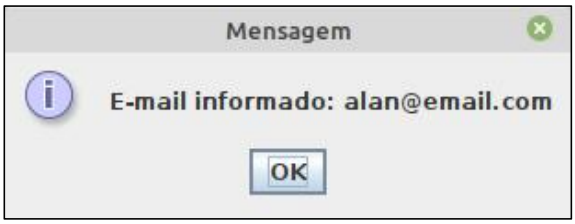
String email = JOptionPane.showInputDialog("Digite seu e-mail");

Classe JOptionPane

Exemplos:

SAÍDA

JOptionPane.showMessageDialog(null, "E-mail informado: " + email);



Classe JOptionPane

Exemplos:

CONFIRMAÇÃO

int resp = JOptionPane.showConfirmDialog(null, email+" está correto?");

resp = 0, se “SIM” / resp = 1, se “NÃO” / resp = 2, se “Cancelar”

Exercícios

1) Cite quatro exemplos diferentes que podem ser considerados como agregação ou composição (dois de cada).

2) O que acontece se uma classe for programada como final? E se um atributo for “setado” como final?

3) Qual a vantagem de programar um método estático?

4) Pesquise e cite mais dois métodos que fazem parte da classe Object.

5) Desenvolva um programa que exemplifica o controle de exceção com try-catch-finally e utilize janelas da classe JOptionPane.

Referências

Stackoverflow. Disponível em <https://pt.stackoverflow.com/questions/25619/composi%C3%A7%C3%A3o-e-agrega%C3%A7%C3%A3o-quais-as-diferen%C3%A7as-e-como-usar>. Último acesso maio/2020. Publicado em setembro/2015.

Devmedia. Disponível em <https://www.devmedia.com.br/java-object-class-entendendo-a-classe-object/30513>. Último acesso maio/2020. Publicado em 2014.