Classes anônimas

Introdução

Classes anônimas são classes internas sem nome. Como eles não têm nome, não podemos usá-los para criar instâncias de classes anônimas. Como resultado, temos que declarar e instanciar classes anônimas em uma única expressão no ponto de uso.

Classes internas anônimas são úteis para escrever classes de implementação para interfaces de ouvinte na programação gráfica.

Podemos estender uma classe existente ou implementar uma interface.

Estender uma classe

Quando instanciamos uma classe anônima de uma existente, usamos a seguinte sintaxe:

Entre parênteses, especificamos os parâmetros exigidos pelo construtor da classe que estamos estendendo. Considere a classe Livro:

```
class Livro {
  public Livro(Text descricao) { /* alguma coisa */ }
  public String getDescricao() {
    return descricao;
  }
}
```

Podemos simplesmente instanciar a classe Livro e utiliza-la:

```
Livro l = new Livro("Java para iniciantes");
System.out.println(l.getDescricao());
// Resultado: Java para iniciantes
```

Ou, podemos instanciar uma classe anônima a partir da classe Livro. Dessa forma poderiamos implementar novos métodos ou sobrescrever os existentes.

```
Livro l = new Livro("Java para iniciantes") {
  @Override
  public String getDescricao() {
    return descricao.toUpperCase();
  }
}
System.out.println(l.getDescricao());
// Resultado: JAVA PARA INICIANTES
```

Implementar uma interface

Também podemos instanciar uma classe anônima a partir de uma interface:

É claro, no Java as interfaces do Java não possuem construtores, portanto os parênteses sempre permanecem vazios. Esta é a única maneira de fazê-lo para implementar os métodos da interface:

```
interface ClickListener {
  public void click();
}
```

Depois de instanciar uma classe anônima, podemos atribuir essa instância a uma variável para poder referenciá-la em algum lugar mais tarde.

Podemos fazer isso usando a sintaxe padrão para expressões Java:

```
ClickListener listener = new ClickListener() {
  @Override
  public void click() { /* alguma tarefa aqui */ }
};
```

Como já mencionamos, uma declaração de classe anônima é uma expressão; portanto, ela deve fazer parte de uma declaração. Isso explica por que colocamos um ponto-e-vírgula no final da declaração.

Obviamente, podemos evitar atribuir a instância a uma variável se criarmos essa instância em linha:

```
Button button = new Button();
button.setClickListener(new ClickListener() {
  @Override
  public void click() { /* alguma tarefa aqui */ }
});
```

Devemos usar essa sintaxe com muito cuidado, pois ela pode facilmente reduzir a legibilidade do código, especialmente quando a implementação do método click() ocupa muito espaço.

Referências

1. Anonymous Classes - Oracle <https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html >
2. Classes Anônimas e Aninhadas em Java - Devmedia <https://www.devmedia.com.br/classes-anonimas-e-aninhadas-em-java/31167 >
3. Anonymous Inner Class in Java <https://www.geeksforgeeks.org/anonymous-inner-class-java/>
4. Anonymous Classes in Java <https://www.baeldung.com/java-anonymous-classes>
5. Anonymous Class Declarations - Oracle <https://docs.oracle.com/javase/specs/jls/se7/html/jls-15.html#jls-15.9.5 >
6. DEITEL, Paul J.; DEITEL, Harvey M.. Java: Como programar. 6. ed. São Paulo: Pearson, 2005. Disponível em <https://plataforma.bvirtual.com.br/Acervo/Publicacao/325>