

Tópicos Abordados

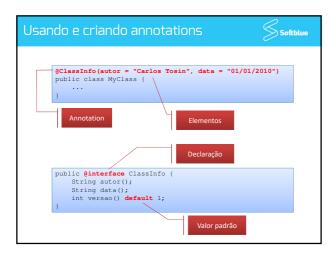


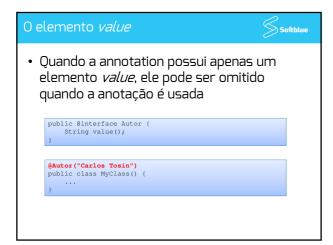
- Annotations
 - O que são
 - Como usar e criar
 - Elemento *Value*, @Retention, @Target
 - @Override e @SuppressWarnings
- Reflection API
 - Objeto *Class*
 - Instanciando objetos
 - Invocando métodos

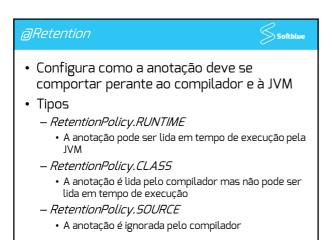
O que são annotations

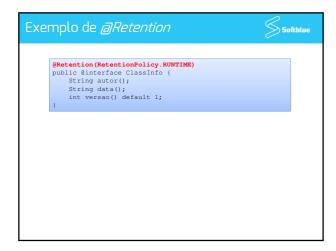


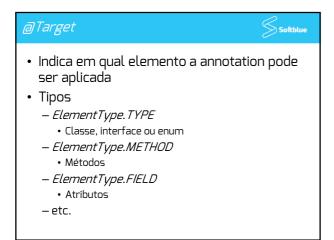
- Mecanismo criado a partir do Java 5
- São declarações no código que podem ser usadas por ferramentas ou programas externos
- Não influenciam diretamente na execução do código











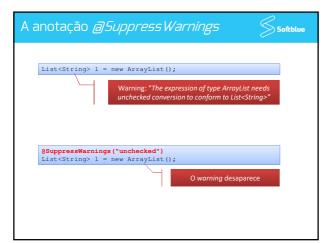


A anotação @Override Indica que um método sobrescreve outro É opcional, mas quando utilizada gera erro de compilação se o método anotado não estiver sobrescrevendo um método da superclasse [@Override public String toString() { return "..."; } Se o método toString() não existir na superclasse, gera erro de compilação

A anotação @SuppressWarnings



- Utilizada para remover mensagens de warning do código
- O seu uso mais comum é para remover mensagens de conversão de tipos quando o generics é utilizado
- Pode anotar classes, métodos e código



A Reflection API do Java permite que as classes conheçam sobre suas estruturas internas O ponto de entrada é um objeto Class, que guarda informações sobre uma classe Class c = String.class; Class c = Class.forName("java.lang.String"); Class Cl

Class<String> c = (Class<String>) Class.forName("java.lang.String");

O objeto Class pode representar também outros elementos que não sejam classes, como interfaces e enums A partir do objeto Class, é possível descobrir quais são os atributos, construtores e métodos Método Descrição getField() Retorna um array de atributos getField() Retorna um atributo específico getConstructors() Retorna um construtores getConstructor() Retorna um construtor específico getMethods() Retorna um array de métodos getMethod() Retorna um método específico

