Java Reflection

Conceitos

O **conceito Reflexão** pode ser definido como o processo no qual um programa de computador pode observar e modificar sua própria estrutura e comportamento.

A utilização de reflexão também é conhecida como **metaprogramação**, pois com sua utilização um programa pode realizar computações a respeito dele próprio.

A **introspecção** é um subconjunto da reflexão que permite um programa a obter informações a respeito de si próprio.

Dessa forma, é possível criar um código que lida com uma classe cuja estrutura ele não conhece.

Esse é um recurso poderoso, mas que precisa ser utilizado com bastante responsabilidade.

Reflexão em JAVA:

Em Java, as classes que implementam funcionalidades relacionadas a reflexão ficam no pacote java.lang.reflect e são conhecidas como **APIReflection**. Apesar do nome, grande parte dessas classes na verdade implementam apenas funções de introspecção. Em outras palavras, é possível obter informações sobre as classes de um software, mas não é possível modificá-las.

Problemas ao usar Reflexão:

Um dos principais preços que se paga ao se utilizar reflexão é o desempenho. Invocar um método a partir de reflexão é mais demorado do que invocar um método diretamente no objeto.

JAVA Reflection API

Obtendo informações sobre as classes:

A classe principal que é o ponto de partida para obtermos informações sobre os elementos de um programa é a classe **java.lang.Class**, por isso devemos instancia-la. Essa instância possui as metainformações, não do objeto, mas da classe. Como quais são seus atributos, quais são os seus métodos e qual é sua superclasse e suas interfaces.

Utilizando referências estáticas:

A forma mais direta de obtermos uma instância de **Class** é através de uma referência estática da classe. Isso é utilizado quando sabemos em tempo de compilação qual classe precisa ser referenciada. Para que isso seja feito, utilizamos o nome da classe seguido por **.class**, o que retorna a instância de **Class** respectiva.

Pode não parecer muito útil utilizar referências estáticas da classe, pois se já sei com qual classe vou trabalhar, para que preciso utilizar reflexão? Porém, essas referências estáticas são muito utilizadas quando precisamos passar uma instância de **Class** como parâmetro para um método.

Um exemplo é o método **imprimeNomeClass()** que recebe como parâmetro o tipo **Class<?>**, significando que qualquer tipo pode ser passado como parâmetro. O uso de tipos genéricos também possibilita que seja feita uma certa restrição no tipo de classe que pode ser passada como parâmetro. Por exemplo, se definirmos o tipo do parâmetro como **Class<? extends Serializable>**, aceitamos como parâmetro somente instâncias de **Class** de tipos que implementam a interface **Serializable**.

Apesar de tipos primitivos em Java não serem considerados classes, é possível obter uma representação de **Class** para eles. Para isso, a referência estática desses tipos pode ser utilizada, como por exemplo **int.class** e **char.class**.

Recuperando a classe de um objeto:

Uma outra forma muito comum de se obter uma instância de **Class** é através de um objeto dessa classe. A classe **Object** define o método **getClass()** que retorna a representação da classe daquele objeto.

Uma String com o nome da classe:

Pode ser obtido uma instância de uma classe através de uma String com o nome da classe. Para fazer isso é necessário chamar o método **forName()** da classe **Class** passando o nome da classe como parâmetro. Ao fazer isso, a classe será procurada pela máquina virtual, carregada caso isso ainda não tenha sido feito e retornada. A chamada desse método pode lançar um **ClassNotFoundException** caso a classe não seja encontrada.