Reflexão em Java: Explorando Classes e Métodos em Tempo de Execução

Digamos que você tenha a seguinte classe abaixo, mas você por algum motivo não tem informação sobre os modificadores de acesso dos atributos e dos métodos dessa classe:

```
public class DispositivoEletronico {

    String tipo;
    String marca;
    double potencia; // em watts

    void configurarDispositivo(String tipo, String marca, double potencia) {
        this.tipo = tipo;
        this.marca = marca;
        this.potencia = potencia;
    }

    double calcularConsumo(int horasDeUso) {
        return this.potencia * horasDeUso; // Consumo em watts-hora
    }

    void exibirDetalhes() {
        System.out.println("Tipo: " + this.tipo);
        System.out.println("Marca: " + this.marca);
        System.out.println("Potência: " + this.potencia + " watts");
    }
}
```

A **Reflexão** em Java é uma técnica que permite inspecionar ou modificar o comportamento de classes, métodos e atributos **em tempo de execução**.

Isso significa que você pode descobrir detalhes sobre uma classe, como seus métodos, atributos e modificadores de acesso, sem saber exatamente como a classe foi implementada durante a compilação.

Agora, vamos destrinchar as informações da classe DispositivoEletronico usando o seguinte código:

```
import java.lang.reflect.Field;
import java.lang.reflect.Method;

public class ExemploReflexao {
    public static void main(String[] args) {
        // Obtendo a classe DispositivoEletronico
        Class<DispositivoEletronico> dispositivoEletronicoClass = DispositivoEletronico.class;

        // Imprimindo os modificadores de acesso dos atributos
        System.out.println("Atributos:");
        Field[] fields = dispositivoEletronicoClass.getDeclaredFields();
        for (Field field : fields) {
            int modifiers = field.getModifiers();
            String accessModifier = java.lang.reflect.Modifier.toString(modifiers);
            System.out.println(accessModifier + " " + field.getName());
```

```
        }

        // Imprimindo os modificadores de acesso dos métodos
        System.out.println("\nMétodos:");
        Method[] methods = dispositivoEletronicoClass.getDeclaredMethods();
        for (Method method : methods) {
            int modifiers = method.getModifiers();
            String accessModifier = java.lang.reflect.Modifier.toString(modifiers);
            System.out.println(accessModifier + " " + method.getName() + "()");
        }
    }
}
```

Vamos ver o que este código faz, um passo de cada vez:

1. **Importações**:

   ```
   import java.lang.reflect.Field;
   import java.lang.reflect.Method;
   ```

   Aqui, estamos importando as classes da biblioteca `java.lang.reflect`, que nos dão as ferramentas para usar reflexão. `Field` representa os atributos de uma classe e `Method` representa seus métodos.

2. **Definição da Classe**:

   ```
   public class ExemploReflexao {
   public static void main(String[] args) {
   ```

   Essa é a classe principal onde o método `main` será executado. Tudo o que vai ser impresso ou manipulado acontece dentro deste método.

3. **Obtendo a classe `DispositivoEletronico` usando reflexão**:

   ```
   Class<DispositivoEletronico> dispositivoEletronicoClass =
   DispositivoEletronico.class;
   ```

   Neste ponto, o código está obtendo uma referência da classe `DispositivoEletronico` usando o método `.class`. Essa referência é do tipo `Class<DispositivoEletronico>`. Ou seja, o Java agora tem uma "descrição" da classe `DispositivoEletronico`, e podemos usar essa referência para explorar seus atributos e métodos.

4. **Imprimindo os modificadores de acesso dos atributos**:

   ```
   System.out.println("Atributos:");
   ```

```
Field[] fields =
dispositivoEletronicoClass.getDeclaredFields();
for (Field field : fields) {
    int modifiers = field.getModifiers();
    String accessModifier =
java.lang.reflect.Modifier.toString(modifiers);
    System.out.println(accessModifier + " " +
field.getName());
}
```

Vamos ver o que acontece aqui:

- `dispositivoEletronicoClass.getDeclaredFields()` retorna um array de `Field` contendo todos os atributos (campos) declarados na classe `DispositivoEletronico`, sejam eles privados, protegidos ou públicos.
- O loop `for` percorre esse array de atributos.
- Para cada atributo, o código usa `field.getModifiers()` para obter os **modificadores de acesso** (como `public`, `private`, `protected`, etc.).
- O método `java.lang.reflect.Modifier.toString(modifiers)` converte esses modificadores para uma string legível. Usamos aqui uma importação completa (`java.lang.reflect.`) porque não importamos explicitamente a classe Modifier, se quisessemos apenas fazer referência a classe Modifier teriamos que adicionar `import java.lang.reflect.Modifier;`
- Finalmente, ele imprime o modificador de acesso seguido pelo nome do atributo usando `field.getName()`.

Isso significa que o código está mostrando os modificadores de acesso e o nome de todos os atributos da classe `DispositivoEletronico`.

5. **Imprimindo os modificadores de acesso dos métodos**:

```
System.out.println("\nMétodos:");
Method[] methods =
dispositivoEletronicoClass.getDeclaredMethods();
for (Method method : methods) {
    int modifiers = method.getModifiers();
    String accessModifier =
java.lang.reflect.Modifier.toString(modifiers);
    System.out.println(accessModifier + " " +
method.getName() + "()");
}
```

Este bloco é similar ao dos atributos, mas agora estamos lidando com os **métodos**:

- `dispositivoEletronicoClass.getDeclaredMethods()` retorna um array de `Method` contendo todos os métodos declarados na classe `DispositivoEletronico`.
- O loop percorre esses métodos, pega os modificadores de acesso de cada método usando `method.getModifiers()` e os converte para string.
- Finalmente, imprime o modificador e o nome do método.

Assim, o código mostra quais métodos existem na classe `DispositivoEletronico`, juntamente com seus modificadores de acesso (como `public`, `private`, etc.).

Executando o código :

```
$ java -cp .  ExemploReflexao
Atributos:
private tipo
private marca
private potencia

Métodos:
public exibirDetalhes()
public calcularConsumo()
public configurarDispositivo()
```