Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia da Electrónica das Telecomunicações e dos Computadores

Infraestruturas Computacionais Distribuídas

Introspecção no Java

Versão 1.2

Analisar objectos através da introspecção

- Introspecção: Mecanismo para a descoberta de informação sobre objectos e classes em tempo de execução *(runtime)*
- Utilidade:
  - Mostrar informação sobre um objecto
  - Criar instâncias de classes cujo nome só é conhecido em tempo de execução
  - Invocação de métodos apenas sabendo o seu nome

Introspecção em Java

- Através da classe *java.lang.Class*
- Qualquer tipo em java é representado por uma instancia da classe `java.lang.Class`
- Todos os Objectos em Java herdam de `java.lang.Object` obtendo o método `getClass()`
- Com o objecto da classe *Class* podemos:
  - Perguntar informação acerca da classe
  - Descobrir os seus campos e métodos
  - Criar novas instancias (objectos) dessa classe
  - Descobrir a super classe, subclasse e os interfaces dessa classe

Métodos úteis da classe *Class (1)*

- Obter o objecto da classe *Class* que representa a classe especificada pelo seu nome
  - `public static Class forName(String className)`
- Retornar o nome completo do objecto da classe *Class*
  - *`Public String getName()`*
    - Ex: “java.util.Vector”
- Obter um conjunto de *flags* com a informação acerca da classe, como por ex. se é abstracta se é um interface, etc...
  - `public int getModifiers()`
- Retornar uma nova instancia do tipo representado pelo objecto desta classe.
  - `Public Object newInstance()`
    - Assume um construtor sem argumentos

Métodos úteis da classe *Class (2)*

- Obter um *array* com todas as classes internas da classe
  - `public Class[] getClasses()`
- Saber quais os construtores, campos e métodos que existem na classe
  - `public Constructor getConstructor(Class[] params)`
  - `public Constructor[] getConstructors()`
  - `public Field getField(String name)`
  - `public Field[] getFields()`
  - `public Method getMethod(String Name,Class[] params)`
  - `public Method[] getMethods()`
- Descobrir qual o *package* e a super classe
  - `public Package getPackage()`
  - `public Class getSuperClass()`

Programar com a classe *Class*

- No método `toString`:

```
public String toString(){
   return "O meu tipo é: " + getClass().getName();
}
```

- Imprimir os nomes de todos os métodos de uma classe:

```
public void imprimeMetodos(){
   Class c = this.getClass();
   Method[] m = c.getMethods();
   for(int i=0; i<m.length; i++)
     System.out.println("Metodo[" + i + "]: " + m[i]);
}
```

Programar com a classe *Class*

- Output gerado:

```
Metodo[0]: public java.lang.String exintrospeccao.MyClass.toString()
Metodo[1]: public void exintrospeccao.MyClass.imprimeMetodos()
Metodo[2]: public native int java.lang.Object.hashCode()
Metodo[3]: public final native java.lang.Class
   java.lang.Object.getClass()
Metodo[4]: public final void java.lang.Object.wait(long,int) throws
   java.lang.InterruptedException
Metodo[5]: public final void java.lang.Object.wait() throws
   java.lang.InterruptedException
Metodo[6]: public final native void java.lang.Object.wait(long) throws
   java.lang.InterruptedException
Metodo[7]: public boolean java.lang.Object.equals(java.lang.Object)
Metodo[8]: public final native void java.lang.Object.notify()
Metodo[9]: public final native void java.lang.Object.notifyAll()
```


Outras classes relacionadas

- Residem no package *java.lang.reflect*
  - Field
    - `public Object get(Object obj)`
    - `public void set(Object obj, Object value)`
  - Constructor
    - `public Object newInstance(Object[] args)`
  - Method
    - `public Object invoke(Object obj, Object[] args)`

Obter o objecto *Class*

- Todas as classes tem um objecto *Class* correspondente que se pode obter:
  - Escrevendo o nome da classe seguido por "`.class`"
    - Ex: `Vector.class`
  - Invocando o *`getClass()`* de uma instancia que herde de *Object* (todas)
    - Ex: `Vector.getClass()`
  - Invocando o `Class.forName(className)` passando a String com o nome do tipo
    - Ex: `Class.forName("java.util.Vector")`
  - Carregar uma classe a partir de um dado ficheiro `.class`, usando o objecto *ClassLoader*
    - Ex:
```
ClassLoader cl = ClassLoader.getSystemClassLoader();
cl.loadClass("myPackage.myClass");
```
- Também se pode saber a classe de um objecto recorrendo à keyword ***instanceof***
  - *Ex:*
```
if(x instanceof Circle)
((Circle)x).setRadius(5);
```


Exemplo

```
public class ExIntrospeccao {
  public static void main(String[] args) {
    try{
      Class cl = Class.forName("java.awt.Rectangle");
      Class[] paramTypes = new Class[] {Integer.TYPE, Integer.TYPE};
      Constructor ct = cl.getConstructor(paramTypes);

      Object[] constArgs = new Object[] {new Integer(12), new Integer(21)};
      Object rectang = ct.newInstance(constArgs);

      Method m = cl.getMethod("getWidth", null);
      Object width = m.invoke(rectang, null);

      System.out.println("O objecto é: " + rectang);
      System.out.println("A largura é: " + width);
    }catch(Exception e){e.printStackTrace();}
  }
}
```

Resultado:
O objecto é: java.awt.Rectangle[x=0,y=0,width=12,height=21]
A largura é: 12.0

Utilizações da Introspecção

- Poder carregar e usar classes recebidas através da rede
- IDE's (Integrated development Environments)
  - Ex: JBuilder, para mostrar os métodos de um objecto
- JavaBeans (componentes GUI dinâmicos)
- JDBC databases
- JavaMail
- Jini
- ...

Benefícios

- A introspecção ajuda a manter o *software* robusto
- Pode ajudar a tornar as aplicações
  - Flexíveis
  - Expansíveis
- Pode reduzir o tamanho da imagem em memória *(footprint)* de uma aplicação
- Melhora a performance da aplicação pois, inicialmente, carrega menos classes
- Carregamento das classes apenas quando são necessárias
- Melhorar a reutilização de código
- Permite reduzir código condicional

"*Bibliografia*" utilizada

- Java tutorial da Introspecção:
  - http://java.sun.com/docs/books/tutorial/reflect/
- O'Reilly Tutorial:
  - http://www.oreillynet.com/pub/d/860
- The Java™ Programming Language, 3rd Edition
  - Arnold, Gosling, Holmes
  - Sun® Microsystems