The Java Reflection API

Java Reflection – API

- The Reflection API allows to:
  - Handle classes (`java.lang.Class`)
  - Handle constructors (`java.lang.reflect.Constructor`)
  - Handle methods, their return type and their parameters (`java.lang.reflect.Method`)
  - Handle class attributes (`java.lang.reflect.Field`)
  - Handle types (`java.lang.reflect.Type`)
- Fields, methods and constructors are `Member` of a class

Java Reflection – API

- Working with classes:
  - Each class has a `class` attribute to get the corresponding `Class` instance
  - Each object has a `getClass()` method to get its `Class` instance
  - The `Class` class has several methods to search for classes, for example:
    - `Class.forName(String name)` searches a class based on its package and on its name

Java Reflection – API

- Working with constructors/instantiating classes:
  - `Class.newInstance()` allows instantiating classes with a default contructor
  - `Class.getConstructors()` returns an array of public constructors
  - `Class.getDeclaredConstructors()` returns an array of all the constructors (public, package, protected, private)
  - Constructors are invoked using `Constructor.newInstance(Object[] args)`

Java Reflection – API

- Working with methods:
  - `Class.getMethods()` returns an array of public methods
  - `Class.getDeclaredMethods()` returns an array of all methods, whatever their modifier is
  - Methods are invoked using `Method.invoke(Object obj, Object[] args)`

Java Reflection – API

- Working with fields:
  - `Class.getFields()` returns an array of public attributes
  - `Class.getDeclaredFields()` returns an array of all attributes, whatever their modifier is
  - Various `getXxx()` methods are available (`getBoolean(Object obj)`, `getInt(Obj obj)`, etc.)
  - Corresponding `setXxx()` methods are available (e.g. `setBoolean(Object obj, boolean b)`, etc.)

Upward compatibility example (1/2)

```
VariableResolver variableResolver = null;
try {
  // this plugin is built against CC plugin 1.0
  VariableResolver =
    new BuildVariableResolver(build, launcher);
}
catch(NoSuchMethodError nsme) {
  // cf. next slide
}
```

Upward compatibility example (2/2)

```
// ...
catch(NoSuchMethodError nsme) {
  // upward compatibility with CC plugin 1.1
  try {
    variableResolver = (VariableResolver)
BuildVariableResolver.class.getConstructors()
[0].newInstance(build,
Computer.currentComputer());
  } catch(Exception e) {
    // ...
  }
}
```

Reflection vs. Introspection

- Introspection is tightly linked to the JavaBeans API:
  - `java.beans.BeanInfo`: Interface providing methods to describe a Java Bean
  - `java.beans.Introspector.getBeanInfo()`: Method used to get instances of `BeanInfo`
- Introspection uses the Reflection API
- All in all: Reflection and introspection are often used as synonyms