

# Aula Prática 12

## Objetivos

Java Reflection

### Problema 12.1

Utilizando as facilidades de *Reflection* do Java escreva um programa que leia o nome de uma classe (pode usar as que construiu até agora) e que apresente as características dessa classe (interfaces, superclasses, construtores, métodos, atributos).

- Acrescente a possibilidade de criar objetos dessa classe, identificando os construtores, deixando o utilizador decidir qual usar e aceitando os argumentos a partir na consola.
- Guarde os elementos que forem sendo criados numa lista e invoque um método comum a todos (por exemplo `toString()`).
- Inclua a possibilidade de invocar outros métodos sobre esses objetos.

*Nota: A construção e invocação deve ser sempre decidida pelo utilizador. Tire partido das classes `java.lang.Class` e `java.lang.reflect.*`*

### Problema 12.2

Tome como referência o seguinte código (`IPlugin.java` e `Plugin.java`). Construa um conjunto de classes que implementem `IPlugin` e que permitam mostrar as funcionalidades do programa principal.

```
// IPlugin.java
package reflection;
public interface IPlugin {
    public void fazQualQuerCoisa ();
}

// Plugin.java
package reflection;

import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;

abstract class PluginManager {
    public static IPlugin load(String name) throws Exception {
        Class<?> c = Class.forName(name);
        return (IPlugin) c.newInstance();
    }
}

public class Plugin {
    public static void main(String[] args) throws Exception {

        File proxyList = new File("reflection/plugins");
        ArrayList<IPlugin> plgs = new ArrayList<IPlugin>();
        for (String f: proxyList.list()) {
```

```

        try {
plgs.add(PluginManager.load("reflection."+f.substring(0,f.lastIndexOf('.'))));
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
    Iterator<IPlugin> it = plgs.iterator();
    while (it.hasNext()) {
        it.next().fazQualQuerCoisa();
    }
}
}

```

### Problema 12.3

Refça o exercício 5.3 utilizando Java Reflection para implementar um sistema de plugins que permitam operar com diferentes formatos de ficheiro de agenda de contactos.

- Comece por especificar a interface dos plugins (loadFile, saveFile, etc);
- Prepare o programa principal para carregar automaticamente os plugins. Defina um diretório de defeito para colocação dos plugins.
- Desenvolva um plugin para cada um dos formatos identificados. Coloque os “.class” no diretório de plugins.
- Teste a aplicação.