

```

1 // Example 8.3 foi/examples/ReflectionDemo.java
2 //
3 //
4 // Copyright 2003 Bill Campbell and Ethan Bolker
5 import java.lang.reflect.*;
6
7 // A short program to illustrate how Java uses
8 // class information dynamically.
9 //
10 //
11 // This file declares class Greeting as well as class
12 // ReflectionDemo. Java requires that a public class be
13 // declared in a file that matches its name, but Greeting
14 // is not marked public.
15 //
16 // %> java ReflectionDemo
17 // Greeting@93dee9 is an instance of class Greeting
18 // classOfG.toString(): class Greeting
19 // classOfG.getName(): Greeting
20 // fields in class Greeting (not inherited):
21 // name: message, type: class java.lang.String
22 // methods in class Greeting (not inherited):
23 // invoking hello
24 // hello, world!
25 // Creating an object when you know the name of its class:
26 // g = (Greeting)Class.forName("Greeting").newInstance();
27 // g.toString(): Greeting@6f0472
28 // Try to create an instance of nonexistent class Foo:
29 // java.lang.ClassNotFoundException: Foo
30
31 public class ReflectionDemo
32 {
33     public static void main( String[] args )
34     {
35         Greeting g = new Greeting();
36         Class classOfG = g.getClass();
37         out(g.toString() + " is an instance of " +
38             classOfG.toString());
39         out("classOfG.toString(): " + classOfG.toString());
40         out("classOfG.getName(): " + classOfG.getName());
41
42         out("fields in class Greeting (not inherited):");
43
44         Field[] greetingFields = classOfG.getFields();
45         for (int i=0; i < greetingFields.length; i++) {
46             Field f = greetingFields[i];
47             if (f.getDeclaringClass() == classOfG) {
48                 out("name: " + f.getName() + ", type: " + f.getType());
49             }
50
51             out("methods in class Greeting (not inherited):");
52
53             Method[] greetingMethods = classOfG.getMethods();
54             for (int i=0; i < greetingMethods.length; i++) {
55                 Method m = greetingMethods[i];
56

```

```

57         if (m.getDeclaringClass() == classOfG) {
58             out("invoking " + m.getName());
59             try {
60                 m.invoke(g, null);
61             }
62             catch( Exception e) {
63                 out(e.toString());
64             }
65         }
66     }
67
68     out("Creating an object when you know the name of its class:");
69     out("g = (Greeting)Class.forName(\"Greeting\").newInstance()");
70     try {
71         g = (Greeting)Class.forName("Greeting").newInstance();
72         out("g.toString(): " + g.toString());
73     }
74     catch (Exception e) { // couldn't find class
75         out(e.toString());
76     }
77
78     out("Try to create an instance of nonexistent class Foo:");
79     Object o;
80     try {
81         o = Class.forName("Foo").newInstance();
82     }
83     catch (Exception e) { // couldn't find class
84         out(e.toString());
85     }
86
87     // too lazy to type "System.out.println()"
88     // public static void out( String s )
89     {
90         System.out.println(s);
91     }
92
93     }
94
95     class Greeting
96     {
97         public String message = "hello, world";
98
99         public void hello()
100         {
101             System.out.println(message + "!");
102         }
103     }

```