```
// This example is from the book "Java in a Nutshell, Second Edition".
// Written by David Flanagan.  Copyright (c) 1997 O'Reilly & Associates.

import java.lang.reflect.*;

/** A program that displays a class synopsis for the named class */
public class ShowClass {
  /** The main method.  Print info about the named class */
  public static void main(String[] args) throws ClassNotFoundException {
    Class c = Class.forName(args[0]);
    print_class(c);
  }

  /** Display the modifiers, name, superclass and interfaces of a class
   *  or interface. Then go and list all constructors, fields, and methods. */
  public static void print_class(Class c)
  {
    // Print modifiers, type (class or interface), name and superclass.
    if (c.isInterface()) {
      // The modifiers will include the "interface" keyword here...
      System.out.print(Modifier.toString(c.getModifiers()) +  c.getName());
    }
    else
      System.out.print(Modifier.toString(c.getModifiers()) + " class " +
                       c.getName() +
                       " extends " + c.getSuperclass().getName());

    // Print interfaces or super-interfaces of the class or interface.
    Class[] interfaces = c.getInterfaces();
    if ((interfaces != null) && (interfaces.length > 0)) {
```



```
      if (c.isInterface()) System.out.println(" extends ");
      else System.out.print(" implements ");
      for(int i = 0; i < interfaces.length; i++) {
        if (i > 0) System.out.print(", ");
        System.out.print(interfaces[i].getName());
      }
    }
    System.out.println(" {");           // Begin class member listing.

    // Now look up and display the members of the class.
    System.out.println(" // Constructors");
    Constructor[] constructors = c.getDeclaredConstructors();
    for(int i = 0; i < constructors.length; i++)      // Display constructors.
      print_method_or_constructor(constructors[i]);

    System.out.println(" // Fields");
    Field[] fields = c.getDeclaredFields();           // Look up fields.
    for(int i = 0; i < fields.length; i++)            // Display them.
      print_field(fields[i]);

    System.out.println(" // Methods");
    Method[] methods = c.getDeclaredMethods();        // Look up methods.
    for(int i = 0; i < methods.length; i++)           // Display them.
      print_method_or_constructor(methods[i]);

    System.out.println("}");            // End class member listing.
  }

  /** Return the name of an interface or primitive type, handling arrays. */
  public static String typename(Class t) {
```

```
  String brackets = "";
  while(t.isArray()) {
    brackets += "[]";
    t = t.getComponentType();
  }
  return t.getName() + brackets;
}

/** Return a string version of modifiers, handling spaces nicely. */
public static String modifiers(int m) {
  if (m == 0) return "";
  else return Modifier.toString(m) + " ";
}

/** Print the modifiers, type, and name of a field */
public static void print_field(Field f) {
  System.out.println("  " +
                     modifiers(f.getModifiers()) +
                     typename(f.getType()) + " " + f.getName() + ";");
}

/** Print the modifiers, return type, name, parameter types and exception
 *  type of a method or constructor.  Note the use of the Member interface
  * to allow this method to work with both Method and Constructor objects */

public static void print_method_or_constructor(Member member) {
  Class returntype=null, parameters[], exceptions[];
  if (member instanceof Method) {
    Method m = (Method) member;
    returntype = m.getReturnType();
```



```
    parameters = m.getParameterTypes();
    exceptions = m.getExceptionTypes();
  } else {
    Constructor c = (Constructor) member;
    parameters = c.getParameterTypes();
    exceptions = c.getExceptionTypes();
  }

  System.out.print("  " + modifiers(member.getModifiers()) +
                   ((returntype!=null)? typename(returntype)+" " : "") +
                   member.getName() + "(");
  for(int i = 0; i < parameters.length; i++) {
    if (i > 0) System.out.print(", ");
    System.out.print(typename(parameters[i]));
  }
  System.out.print(")");
  if (exceptions.length > 0) System.out.print(" throws ");
  for(int i = 0; i < exceptions.length; i++) {
    if (i > 0) System.out.print(", ");
    System.out.print(typename(exceptions[i]));
  }
  System.out.println(";");
 }
}
```