Worksheet 02: Further exercises on Java Reflection

The Reflection API allows a Java program to inspect and manipulate itself; it comprises the `java.lang.Class` class and the `java.lang.reflect` package, which represents the members of a class with `Method`, `Constructor`, and `Field` objects.

1. Write a Java program that reads the name of a class from the command line and emits the interface of the class in Java syntax (interface or class, modifiers, constructors, methods, fields; no method bodies).

   Hint: You can load a class whose name you know with `java.lang.Class.forName()`. The `java.lang.Class` class offers a rich interface that enables you to inspect the interface of any class.

   Apply this program to a set of classes and interfaces as test input. You may also use the program on itself.

2. Write a program that reads a class name and a list of arguments, and creates an object of that class where the read arguments are passed to the constructor.

   Hint: Treat arguments as strings. A `java.lang.Class` can enumerate its constructors. Choose a constructor with the appropriate parameter count. Then, find the parameter types. To create typed argument objects, call the appropriate constructors that take a string as their only argument. Call dynamic constructors using

   `java.lang.reflect.Constructor.newInstance()`.

3. Write a `JUnit` test to help grade the internal correctness of a student's submitted program for a hypothetical assignment.

   Make the tests fail if the class under test has any of the following:

   - more than four fields
   - any non-private fields
   - any fields of type `ArrayList`
   - fewer than two private helper methods
   - any method that has a `throws` clause
   - any method that returns an `int`
   - missing a zero-argument constructor

4. Normally it is up to the programmer to write a `toString()` method for each class one creates. This exercise is about writing a general `toString` method once and for all. As part of the reflection API for java, it is possible to find out which fields exist for a given object, and to get their values. The purpose of this exercise is to make a `toString` method that gives a printed representation of any object, in such a manner that all fields are printed, and references to other objects are handled as well.

   To solve this exercise, you will need to examine the `java.lang.reflect` API. You will (probably) end up with around 50–60 lines of code, including that necessary for trying it out.