Tutorial: APIs

Purpose

Let you practise designing and criticising APIs. (I think this one may be quite hard. It would be helpful if you let me know, maybe on the questionnaire at the end of term, whether you find it refreshingly non-trivial or hopelessly confusing... We will be back to more straightforward tutorial work next week.)

Exercise

NB you need to have watched the Bloch lecture before you proceed.

1. Consider again the Party class of the other tutorials (Week 5 for example). Write down its API in the current state. What do you think of it? Is anything missing? Can you make any improvements?

2. Recall what you know about Java applets (the java.applet.Applet class). (If that's not much, do this with the Getting Started with Applets tutorial `http://docs.oracle.com/javase/tutorial/deployment/applet/getStarted.html` at hand.) Represent the most important methods on a UML protocol state machine diagram.

3. Imagine you have been asked to design a class `Date` with the headline specification "The class Date represents a specific instant in time, with millisecond precision." Do you see any problem with this headline? Even if you do, attempt to design an API for this class. Note that a major decision you will need to take is: should a Date object be mutable or immutable? (What difference does this make to your API?)

   Compare your draft with the actual content of the JDK7 java.util.Date class, see `http://docs.oracle.com/javase/7/docs/api/java/util/Date.html`. Look particularly at the Deprecated methods and compare them with their suggested replacements. (A method is marked Deprecated if the designers now think they can offer a better alternative to using it; it is not immediately removed, in order to avoid breaking client code that uses it, but the suggestion is that new code should not use it, and that eventually it may be removed.) Can you guess at why the original methods have been deprecated and why the replacement mechanisms are considered better?

   Now read some of the criticism on the web of the java.util.Date API design, for example here: `http://www.jroller.com/cpurdy/entry/the_seven_habits_of_highly`. Note that a rewritten version is expected in Java 8, see e.g. `http://www.infoq.com/news/2012/09/jsr310-java8` !

Extension exercises

1. Imagine you are designing a class to represent a File. What functionality should it have? Without consulting any File classes you may have used, draft an API for this class.

2. Now compare your API with the one in the JDK7 listed here: `http://docs.oracle.com/javase/7/docs/api/`. You will probably find that it provides capabilities you didn't think of, but focus first on how it provides the capabilities you *did* think of. Are there any interesting differences? Which is better?

3. Follow the link to the `java.nio.file` package and study its design (it's new in JDK7). Here's a blog post about it: `http://codingjunkie.net/java7-file-revolution/`; you might want to follow links e.g. to the `com.google.common.io` package.