jInfer Runner Module Description

Michal Klempa, Mário Mikula, Robert Smetana, Michal Švirec, Matej Vitásek
Advisors: RNDr. Irena Mlýnková, Ph.D., Martin Nečaský, Ph.D.

Praha, 2011

Target audience: developers willing to extend jInfer, specifically hack (or expand) the inference process.

Responsible developer:	Matej Vitásek
Required tokens:	cz.cuni.mff.ksi.jinfer.base.interfaces.inference.IGGenerator cz.cuni.mff.ksi.jinfer.base.interfaces.inference.SchemaGenerator cz.cuni.mff.ksi.jinfer.base.interfaces.inference.Simplifier org.openide.windows.IOProvider
Provided tokens:	none
Module dependencies:	Base
Public packages:	cz.cuni.mff.ksi.jinfer.runner cz.cuni.mff.ksi.jinfer.runner.properties

1 Introduction

`Runner` it the module responsible for inference process. The fact that the inference consists of exactly 3 steps as described in TODO link architecture is hard-wired in jInfer via this module.

2 Structure and operation

The main class responsible for the inference run is the `Runner` in `cz.cuni.mff.ksi.jinfer.runner` package. During its construction it loads the properties of the currently running project (via the `RunningProject` class) to find out which modules are selected for the inference. These modules are looked up and remembered - each new inference run should therefore use a new instance of runner.

The only public method in `Runner` is, unsurprisingly, `run()`. This method will start the first step of the inference process by invoking the selected *`IGGenerator`*'s `start()` method, as described in TODO link inference process in architecture. Callback methods `finishedIGGenerator()`, `finishedSimplifier()` and `finishedSchemaGenerator()` are responsible for invoking the following stages of inference, or in the latter case for presenting the resulting schema to the user and terminating the inference process.

Invocation of every step in the process is encapsulated in a NetBeans *task*: this is the responsibility of `runAsync()` method. First of all, this means that all the work is done in an asynchronous thread independent from the GUI. Second, NBP presents each such task as a progressbar in the bottom right corner of the window, and allows the user to cancel it. Should this happen, the currently running module detects this by checking for `Thread.interrupted()` and responds by throwing an `InterruptedException`. `Runner` catches this exception, terminates the inference and informs the user (`interrupted()` method).

Furthermore, should any unexpected exception occur while running one of the modules, this will get caught in `Runner` again. Inference will be interrupted and user will be notified - this is the responsibility of `unexpected()` method.

Finally, the generated schema is annotated in the end with a comment stating the current date and time, and the modules used in the process. This is the responsibility of `getCommentedSchema()` method.

2.1 Settings

Runner has NetBeans-wide settings determining what should happen after the schema is inferred and which rule displayer should be used. The options panel along with all the logic is in the `cz.cuni.mff.ksi.jinfer.runner.options` package.

2.2 Preferences

Runner naturally has project-wide preferences for selection of inference modules. The panel and its logic is in `cz.cuni.mff.ksi.` package.

3 Extensibility

For certain inference algorithms it might be necessary to completely change the number or order of inference modules. For example, it might be necessary to have a dynamic inference with arbitrary number of iterations over the same module (modules). In this case, *Runner* can serve as a template: adding e.g. a cleaner between *Simplifier* and *SchemaGenerator* would mean just copy-pasting members and methods currently associated to *Simplifier*.