
JavaOne™

ORACLE

Right Decissions at Right Time

Jan Lahoda
Software Developer, NetBeans

NetBeans

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

Motivation

- APIs are rarely done optimally at the beginning
- As is the structure of the code and relationships between code elements
- Making changes is very difficult
- And gets more difficult as the project progresses
- Can these be made easier?

Motivation

- APIs are rarely done optimally at the beginning
- As is the structure of the code and relationships between code elements
- Making changes is very difficult
- And gets more difficult as the project progresses
- Can these be made easier? We think so!

Jackpot (3.0)

- Tool for custom declarative refactorings
- Custom Java-like DSL
- Makes API/structure/relationship changes much easier
- Can also be used to ask questions like
  How many clients are calling this method (in a specified context)?

Archie

- Standard extra-linguistic constraints for architecture and design
- Based on annotations to specify the constraints and annotation processors to enforce them
- Currently on early prototype level
- http://source.apidesign.org/hg/archie
- Devised by Petr Hřebejk

Archie - Design & Architectural Patterns

- `@Utility`: no instance methods/fields
- `@Final`: interface implementable by friends only
- `@Singleton`, `@Factory`
- `@RequiredName`: must have given name, name as a compile-time constant for use via reflection
- `@Namespace`: class that only contains other classes

Archie - Modularization

- Define and **change** the structure of modules easily
- `@Component, @API, @SPI, @Interface, @External`
- OSGi bundles/Java modules could be inferred

Codeviation

- Gathering, integrating and visualization of codebase metadata
- Sources of information include:
  Version control systems, Bug tracking system, Continuous buids & tests, Wiki, Email, Static analysis, Metrics & Statistics, Logs
- More or less on conceptual level so far
- Devised by Petr Hřebejk and Petr Zajac

Jackpot (3.0) - History

- Founded 2000 to improve IDEs&the way devs develop (by Tom Ball, Michael Van De Vanter, James Gosling)
- Incl. code transformation engine
  - Structure/AST based
  - Working prototype existed
  - Transformations: in Java or a custom declarative language
- Its write model adopted by NetBeans in 2006 (NB 6.0)

Jackpot 3.0

- Reviving the declarative language for transformations
- Language part in the upcoming NetBeans IDE 7.1
- For one-off refactorings use NBs Inspect&Refactor
- For library upgrades, place scripts in `META-INF/upgrade/*.hint`
- Can also create test for the scripts: `.../rule.test`

Jackpot 3.0 – The Language

- Basic rule format:

```
    <source-pattern> :: <conditions>
=> <target-pattern> :: <conditions>
=> <target-pattern> :: <conditions>
;;
```

- Example:

```
$f.toURL() :: $f instanceof java.io.File
=> $f.toURI().toURL()
;;
```


Jackpot 3.0 – Patterns: Basics

- Java expression, statement(s), class, variable, method
- Identifiers starting with `$` represent *variables*: a tree node will be bound to them: `$1`, `$lock`, etc.
- Identifiers starting and ending with `$` consume any number of tree nodes:
  - `java.util.Arrays.asList($param)`
  - `java.util.Arrays.asList($params$)`

Jackpot 3.0 – Patterns: Special Forms

- Statement: `$statement;` (more: `$statements$;`)
- 0 or 1: `if ($cond) $then; else $else$;`
- Modifiers: `$mods$ $type$ $variableName;`
- Multiple catches: `try {} $catches$ finally {}`
- More special forms for specific uses

Jackpot 3.0 – Conditions

- `$variable instanceof <type>`
- Predefined: `hasModifier($variable, PRIVATE)`
- Custom conditions (may use pattern matching)
- Condition result can be negated (`!`)
- `&&` works on condition results

Jackpot 3.0 – Target Patterns

- Similar to source patterns
- May use variables from source pattern
- Special form: empty == remove

Jackpot 3.0 – Demos

- `$string.equals("")  => $string.isEmpty()`
- `new String($string) => $string`

Jackpot 3.0 – Missing parts

- Custom handling of comments
- Support for coupled changes:

```
private ClassPath EMPTY = …; => <remove>
create(EMPTY)
=>
create(ClassPath.EMPTY)
```


Jackpot 3.0 – Prototypes

- Standalone command line tool
- Indexes for performance (when occurrences rare)
- Server support
  - More generally: server-side support for IDEs
- https://bitbucket.org/jlahoda/jackpot30/wiki/Home

Conclusions

- Doing architectural/design changes is difficult
- But can be made much easier using tools
- Interested in the tools? Contact:
  - Jan Lahoda (jan.lahodaAToracle.com)
  - Petr Hřebejk (petr.hrebejkAToracle.com)

Q&A


JavaOne™