

Wikiprint Book

Title: @Const and ConstChecker

Subject: Java Path Finder - projects/jpf-aprop/Const

Version: 1

Date: 02/21/2013 07:13:25 AM

Table of Contents

TracNav	3
Introduction	3
Installing JPF	3
User Guide	3
Developer Guide	3
MJI	3
Projects	3
About	4
@Const and ConstChecker	4
Properties	5
Aliases	5

TracNav

- [JPFWiki - Welcome Page](#)

Introduction

- [What is JPF](#)
- [Testing vs model checking](#)
- [Random Example](#)
- [Race Example](#)
- [JPF classification](#)

Installing JPF

- [System requirements](#)
- [Download snapshots](#)
- [Download repositories](#)
- [Create site.properties](#)
- [Install NetBeans IDE plugin](#)
- [Install Eclipse IDE plugin](#)
- [Building and testing](#)

User Guide

- [Application Types](#)
- [JPF Components](#)
- [Configuring JPF](#)
- [Running JPF](#)
- [JPF Output](#)
- [The JPF API](#)

Developer Guide

- [Design](#)
- [Choice Generator](#)
- [Partial Order Reduction](#)
- [Attributes](#)
- [Listener](#)

MJI

- [Mangling for MJI](#)
- [Bytecode Factory](#)
- [Logging](#)
- [Report](#)
- [Embedded](#)
- [JPF tests](#)
- [JPF project layout](#)
- [Create a JPF project](#)
- [Coding Conventions](#)
- [Hosting update site](#)

Projects

- [jpf-core](#)
- [jpf-actor](#)
- [jpf-awt](#)
- [jpf-awt-shell](#)

- [jpf-concurrent](#)
- [jpf-cv](#)
- [jpf-delayed](#)
- [jpf-guided-test](#)
- [jpf-mango](#)
- [jpf-racefinder](#)
- [jpf-rtembed](#)
- [jpf-statechart](#)
- [net-iocache](#)
- [jpf-aprop](#)
- [jpf-numeric](#)
- [jpf-symbc](#)
- [jpf-concolic](#)
- [jpf-symbc-load?](#)
- [jpf-extended-test-gen](#)
- [jpf-parallel-spf?](#)
- [eclipse-jpf](#)
- [netbeans-jpf](#)
- [jpf-inspector](#)
- [jpf-shell](#)
- [jpf-template](#)
- [jpf-trace-server](#)
- [standard NB example](#)
- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

[About](#)

- [About this Wiki](#)
- [About the Mailing Lists](#)
- [About the Development Process?](#)
- [About the Repository?](#)
- [How to Contribute](#)
- [JPF contributor account](#)
- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

@Const and ConstChecker

This annotation can be used for whole classes, fields and methods. If it is used in a class scope, it means that none of the fields are allowed to change outside static init or constructor invocations:

```
import gov.nasa.jpf.annotation.Const;

@Const static class ConstObj {
    int d;
```

```

ConstObj () {
    initialize();
}

void initialize() { // Ok to call from within ctor, error if called outside
    d = 42;
}

...
}
}

```

Using `@Const` for single fields is similar to using `final` attributes, except of that `final` only allows to assign values to such fields from within constructors, which prohibits functional decomposition of complex constructors.

If the annotation is used in method scope, fields of the defining class are not allowed to be assigned from within this method, which includes nested method invocations:

```

class SomeClass {
    int d;

    void set() {
        d = 42;
    }

    @Const void dontDoThis() { // 'd' not allowed to be changed directly or indirectly within this method
        set();
    }
}

```

Properties

To let JPF check for `@Const` violations, the associated `gov.nasa.jpf.aprop.listener.ConstChecker` has to be configured:

```
listener=.aprop.listener.ConstChecker
```

Alternatively, the `ConstChecker` can be included in the JPF autoload listener set

```

listener.autoload=gov.nasa.jpf.annotation.Const, ...

listener.gov.nasa.jpf.annotation.Const=.aprop.listener.ConstChecker
...

```

Aliases

Although we extend the semantics of "const", the JSR-305 `javax.annotation.concurrent.Immutable` can be used with the same listener