

## **Wikiprint Book**

**Title: @Region**

**Subject: Java Path Finder - projects/jpf-aprop/Region**

**Version: 6**

**Date: 02/21/2013 09:36:47 AM**

## Table of Contents

TracNav	3
Introduction	3
Installing JPF	3
User Guide	3
Developer Guide	3
MJJ	3
Projects	3
About	4
<b>@Region</b>	<b>4</b>
Questions to get us started:	5
Related Work	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](#)

### [MJJ](#)

- [Mangling for MJJ](#)
- [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](#)

## **@Region**

@Region is an annotation with package or type target. It states that all annotated types or packages belongs to some abstract region with specified id. It is used in conjunction with Confined annotation.

...include security related issues

The annotation is not inheritable. The following is my justification for this:

Assume we have:

- class X has some final methods which manipulates object of class A. Class X belongs to regionX, class A is confined for regionX.
- class Y extends X and has some other final methods which manipulates object B. Class Y belongs to regionY, class B is confined for regionY
- class Z extends Y. Letting @Region be inheritable would allow Z to access object A. It may be legal, but it would be better to write it explicitly.

It should be possible to add dynamic region specification. It would look like the following:

```
Verify.startRegion("abc");  
// here come some instructions  
Verify.endRegion("abc");
```

Abstract regions are organized into the tree structure. Example of doing that is in

<http://bitbucket.org/rogaall/gsoc2010/src/tip/ConfinedTest/src/gov/nasa/jpf/test/confined/RegionSemantics/gov.nasa.jpf.test.confined.RegionSemantics> package.

### Questions to get us started:

- What can be defined as a 'region'?
- What annotations might be used in conjunction with the '@Region' annotation?
- Can regions be nested? If so, what are the implications of this?
- Can regions overlap? If so, what are the implications of this?
- How do regions work with the class hierarchy?
- How do regions work with other aspects of the Java language?

### Related Work

- ...