Item 185
git_comments:

1. * * Whether to carry over annotations on the copied constructors. * Currently Closure annotation members are not supported. * * @return true if copied constructor should keep constructor annotations
2. * * Whether to carry over parameter annotations on the copied constructors. * Currently Closure annotation members are not supported. * * @return true if copied constructor should keep parameter annotations
3. * * Copies all <tt>candidateAnnotations</tt> with retention policy {@link java.lang.annotation.RetentionPolicy#RUNTIME} * and {@link java.lang.annotation.RetentionPolicy#CLASS}. * <p> * Annotations with {@link org.codehaus.groovy.runtime.GeneratedClosure} members are not supported for now.
4. tag::inheritconstructors_parameter_annotations[]
5. tag::inheritconstructors_constructor_annotations[]
6. end::inheritconstructors_constructor_annotations[]
7. end::inheritconstructors_parameter_annotations[]

git_commits:

1. **summary:** Merge pull request #562 from paulk-asert/groovy7059
   message: Merge pull request #562 from paulk-asert/groovy7059 GROOVY-7059: @InheritConstructors should replicate annotations on super ...

github_issues:

github_issues_comments:

github_pulls:

1. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
2. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
3. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
   label: code-design
4. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
5. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
   label: code-design
6. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
7. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
8. **title:** rename Java8 package from vm8 to v8 for consistency
   body:
9. **title:** rename Java8 package from vm8 to v8 for consistency
   body:

github_pulls_comments:

1. +1
2. just two questions: (a) why not from v8 to vm8 (vmX is the older one, so I was wondering)? (b) why exclude v8 and vm8 in the build, if this is done to have only one of them?
3. **body:** I asked John to change the Java8 plugin package for consistency. We had used v5, v6, v7, and then vm8 (but so far only in 2.5.0-beta-1). I think 'vm8' is actually slightly more descriptive than 'v8' but doesn't match the convention now in many releases. Since all tests containing vm8 in the core module were plugin related it seems okay to rename them too. There are actually a few other spots in test

names/packages using the string 'vm8' which we could also change to v8 but that can be done separately I think.
label: code-design
4. After some more thought I wonder if this (2.5+) would be the time to start a `org.apache.groovy.vmplugin` or `org.apache.groovy.internal.vmplugin` package and start with a `vm8` package under it establishing it as a new standard naming (i.e., followed soon by `vm9`)? If there's interest in that I'll redo the PR, otherwise will merge this as is.
5. **body:** I think it would be confusing to have the plugins for different vm versions across different packages but I like your idea of doing a bit of a clean up. We are talking about a bunch of what should be internal classes, so I think we have some liberty to change but we'd probably want to do so with minimum disruption for anyone that might be sneaking in and using those "internal" classes. I'd be inclined to leave the `org.codehaus.groovy.vmplugin` package as is but remove `vm8` and deprecate everything else. Then create an `org.apache.groovy.internal.vmplugin` package as you suggest and bring across a merged Java7 class in the `vm7` package which merges the current v5/v6/v7 classes (later versions taking priority). The `vm8` package would of course reappear in the new package. We'd need to double check that didn't alter anything but I believe should be what we want for 2.5+. The tests can all stay in the `vmX` package variants.
label: code-design
6. In the end it does not matter all that much to me if we use v8 or vm8, just that it is only one of them ;) Should we go with org.apache.groovy.internal.vmplugin? We can do that. Frankly, some of the functionality in the plugins should probably move out of them again, since for example generics are really really part of all deployments of Groovy. But that is for a later PR.
7. PR #563 includes what I would think are "reasonably" compatible changes while moving to a new package and deprecating the old plugins.

github_pulls_reviews:

jira_issues:

1. **summary:** @InheritConstructors should replicate annotations on super constructors
description: I want to use Guice to inject some dependencies into a hierarchy of TestNG tests using {{@Inject}} constructor injection. Since many of these tests use exactly the same objects, {{@InheritConstructors}} would be perfect for them, but {{@InheritConstructors}} does not replicate the {{@Inject}} annotation on the superclass, causing Guice to complain about not having an injectable constructor. Constructors created with {{@InheritConstructors}} should replicate the annotations on the constructor methods they're copying.
2. **summary:** @InheritConstructors should replicate annotations on super constructors
description: I want to use Guice to inject some dependencies into a hierarchy of TestNG tests using {{@Inject}} constructor injection. Since many of these tests use exactly the same objects, {{@InheritConstructors}} would be perfect for them, but {{@InheritConstructors}} does not replicate the {{@Inject}} annotation on the superclass, causing Guice to complain about not having an injectable constructor. Constructors created with {{@InheritConstructors}} should replicate the annotations on the constructor methods they're copying.
3. **summary:** @InheritConstructors should replicate annotations on super constructors
description: I want to use Guice to inject some dependencies into a hierarchy of TestNG tests using {{@Inject}} constructor injection. Since many of these tests use exactly the same objects, {{@InheritConstructors}} would be perfect for them, but {{@InheritConstructors}} does not replicate the {{@Inject}} annotation on the superclass, causing Guice to complain about not having an injectable constructor. Constructors created with {{@InheritConstructors}} should replicate the annotations on the constructor methods they're copying.
label: code-design
4. **summary:** @InheritConstructors should replicate annotations on super constructors
description: I want to use Guice to inject some dependencies into a hierarchy of TestNG tests using {{@Inject}} constructor injection. Since many of these tests use exactly the same objects, {{@InheritConstructors}} would be perfect for them, but {{@InheritConstructors}} does not replicate the {{@Inject}} annotation on the superclass, causing Guice to complain about not having an injectable constructor. Constructors created with {{@InheritConstructors}} should replicate the annotations on the constructor methods they're copying.

jira_issues_comments:

1. The @Delegate annotation has methodAnnotations and parameterAnnotations flags to turn such copying of annotations on when delegating. Perhaps a similar thing here might be the way to go. Though, I think traits automatically copies such annotations and has no flags to disable.
2. Thanks for the suggestion. @InheritConstructors now has two additional boolean attributes: {{constructorAnnotations}} and {{parameterAnnotations}}. They default to false (current behavior) but set to true to enable annotation copying.