Competency Examination: Advanced Java

urn:za:co:solms:exams:java:advanced:003

Table of Contents

Theory

Annotations

1. Explain in your own words the significance of the introduction of *Annotations* in Java 5.

2. Think of two innovative uses of Java annotations, and briefly explain how you see it being used.

Test-driven development

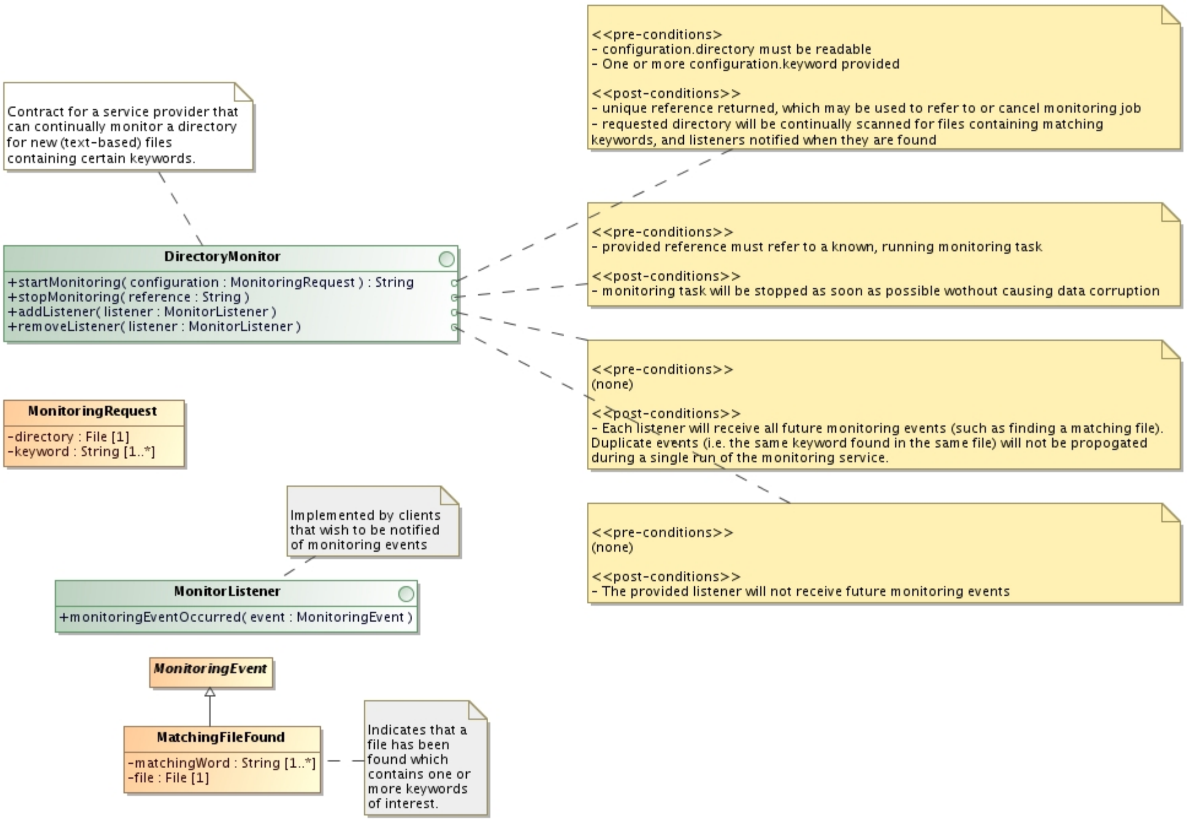
1. Explain test-driven development in your own words.

2. How would you justify the initial time and cost overhead of putting unit tests in place (which could easily increase the time spent on initial coding by 50% or more).

Practical

Imagine a highly simplified search engine / monitoring service, which allows the user to register “alerts” on a particular directory, in order to be notified when a file containing certain keywords are found.

The contract for this service provider offers services to start/stop monitoring a directory for files containing certain keywords, as well as registering clients in order to have them receive events as and when matching files are found. This service must continually run, i.e. if a new file (with sought-after keywords) is placed in a directory being monitored, clients should receive an event indicating this within a reasonable time frame.


DirectoryMonitor
+startMonitoring(configuration : MonitoringRequest) : String
+stopMonitoring(reference : String)
+addListener(listener : MonitorListener)
+removeListener(listener : MonitorListener)
MonitoringRequest
-directory : File [1]
-keyword : String [1..*]
MonitorListener
+monitoringEventOccurred(event : MonitoringEvent)
MonitoringEvent
MatchingFileFound
-matchingWord : String [1..*]
-file : File [1]

Contract for a service provider that can continually monitor a directory for new (text-based) files containing certain keywords.

<<pre-conditions>
- configuration.directory must be readable
- One or more configuration.keyword provided

<<post-conditions>>
- unique reference returned, which may be used to refer to or cancel monitoring job
- requested directory will be continually scanned for files containing matching keywords, and listeners notified when they are found

<<pre-conditions>>
- provided reference must refer to a known, running monitoring task

<<post-conditions>>
- monitoring task will be stopped as soon as possible wothout causing data corruption

<<pre-conditions>>
(none)

<<post-conditions>>
- Each listener will receive all future monitoring events (such as finding a matching file). Duplicate events (i.e. the same keyword found in the same file) will not be propogated during a single run of the monitoring service.

<<pre-conditions>>
(none)

<<post-conditions>>
- The provided listener will not receive future monitoring events

Implemented by clients that wish to be notified of monitoring events

Indicates that a file has been found which contains one or more keywords of interest.

Contract Implementation

Implement the above contract in Java (i.e. write the classes and interfaces necessary to express only the contract), adhering to the formal principles of design-by-contract.

Service Provider Implementation

Create an implementation of the `DirectoryMonitor` contract with Java, ensuring that all pre- and post-conditions are met. Any mechanism may be used for examining the files (keeping in mind the potential performance benefits of using Java NIO together with regular expressions).

In order to continually scan each directory (and to do so concurrently for multiple directories being monitored) the multi-threading facilities of Java would need to be used. The implementor may choose between traditional threading features or the Java 5 Concurrency framework.

The implementation should be implemented in a test-driven manner, in order to allow for the functionality to be verified in an automatable and repeatable manner.

Apache Ant must be used to manage the deliverable and tasks around the project.

Simplistic application client

In as simple a manner as possible, write a small Java application which takes command-line arguments regarding a directory to be monitored, and the keywords which must be searched for. It must use the components built in the previous section to then

- monitor the requested directory for files containing the given keywords, and
- notify the user continually as such files appear in the directory (either graphically, or on the text console).

The app does not require the functionality to register multiple monitoring tasks, cancel tasks, or exit gracefully (CTRL+C will suffice).