Running the CachedClientLookup

While it is OK to poll the local cache, the cache itself must get its contents from lookup services, and in general it is not OK to poll these because doing so involves possibly heavy network traffic. The cache gets its information by registering itself as a listener for service events from the lookup services. The lookup services will then call `notify()` on the cache listener. This is a remote call from the remote lookup service to the local cache, done (probably) using an RMI stub. In fact, the Sun implementation of `ServiceDiscoveryManager` uses a nested class, `ServiceDiscoveryManager.LookupCacheImpl.LookupListener`, which has an RMI stub.

In order for the cache to actually work, it is necessary to set the RMI codebase property `java.rmi.server.codebase` to a suitable location for the class files (such as an HTTP server), and to make sure that the class `net/jini/lookup/ServiceDiscoveryManager$LookupCacheImpl$LookupListener_Stub.class` is accessible from this codebase. The stub file may be found in the library `lib/jini-ext.jar` in the Jini 1.1 distribution. It has to be extracted from there and placed in the codebase using a command such as this:

```
unzip jini-ext.jar \
'net/jini/lookup/ServiceDiscoveryManager$LookupCacheImpl$LookupListener_Stub.class'
\
-d /home/WWW/htdocs/classes
```

Note that the specification just says this type of thing has to be done but does not descend to details about the class name—that is left to the documentation of the `ServiceDiscoveryManager` as implemented by Sun. If another implementation is made of the Jini classes, then it would probably use a different remote class.

Monitoring Changes to the Cache

The cache uses remote events to monitor the state of lookup services. It includes a local mechanism to pass some of these changes to a client by means of the `ServiceDiscoveryListener` interface:

```
package net.jini.lookup;
interface ServiceDiscoveryListener {
    void serviceAdded(ServiceDiscoveryEvent event);
    void serviceChanged(ServiceDiscoveryEvent event);
    void serviceRemoved(ServiceDiscoveryEvent event);
}
```

with event

```
package net.jini.lookup;
class ServiceDiscoveryEvent extends EventObject {
    ServiceItem getPostEventServiceItem();
    ServiceItem getPreEventServiceItem();
}
```

Clients are not likely to be interested in all events generated by lookup services, even for services in which they are interested. For example, if a new service registers itself with ten