SE3030 – Software Architecture

3rd Year – Semester 01

Lecturer: By Udara Samaratunge

Apache Felix (OSGi) Lab Sheet

- **Go to the download page and download and extract the Felix release**
  - ***http://felix.apache.org/site/downloads.cgi***
  - ***http://apache.cs.utah.edu//felix/org.apache.felix.main.distribution-4.0.3.zip***

- **Go to the Felix usage page to learn how to launch the Felix framework**
  - ***http://felix.apache.org/site/apache-felix-framework-usage-documentation.html***

Dictionary Service Bundle

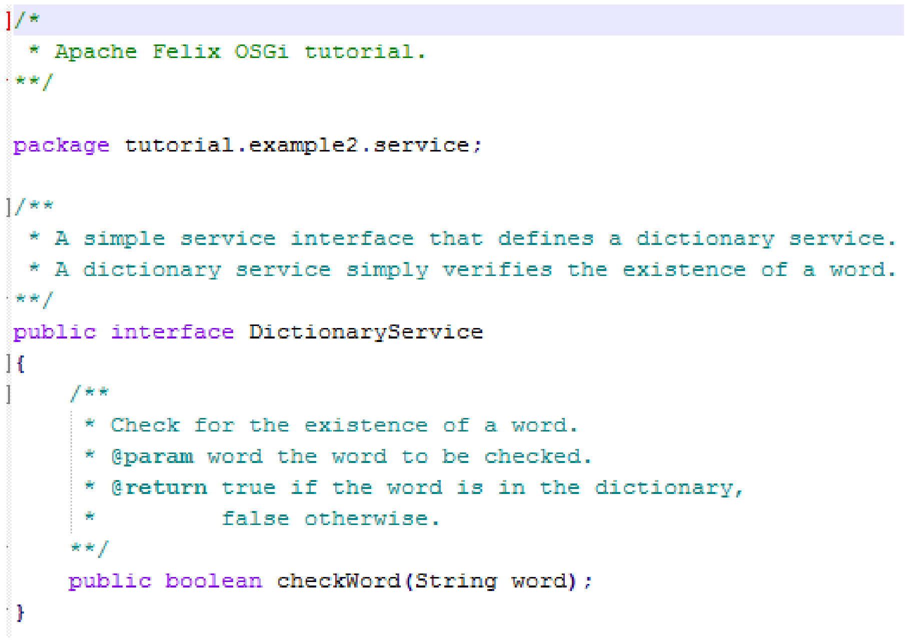
1) Creates a bundle that implements an OSGi service.
2) Implementing an OSGi service is a two-step process
3) First we must define the interface of the service and then we must define an implementation of the service interface.

Client service bundle

To consume the dictionary service.

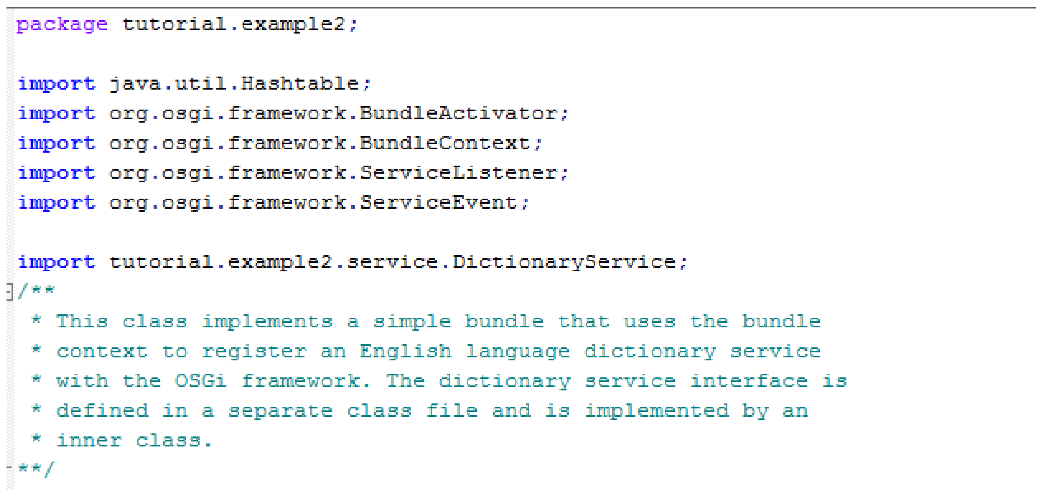
Export dictionary service and use the service

1) Dictionary service interface in a file called *DictionaryService.java*

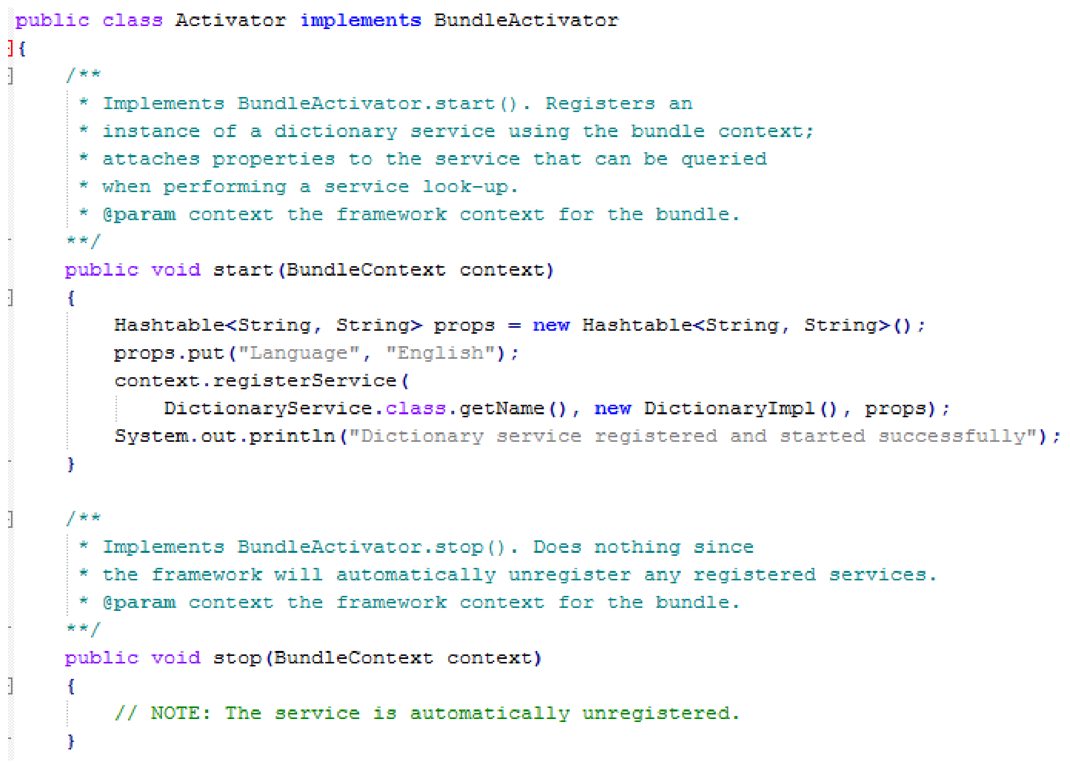


```java
/*
 * Apache Felix OSGi tutorial.
**/

package tutorial.example2.service;

/**
 * A simple service interface that defines a dictionary service.
 * A dictionary service simply verifies the existence of a word.
**/
public interface DictionaryService
{
    /**
     * Check for the existence of a word.
     * @param word the word to be checked.
     * @return true if the word is in the dictionary,
     *         false otherwise.
    **/
    public boolean checkWord(String word);
}
```

2) In the following source code, the bundle uses its bundle context to register the dictionary service. We implement the dictionary service as an inner class of the bundle activator class, but we could have also put it in a separate file. The source code for our bundle is as follows in a file called Activator.java:

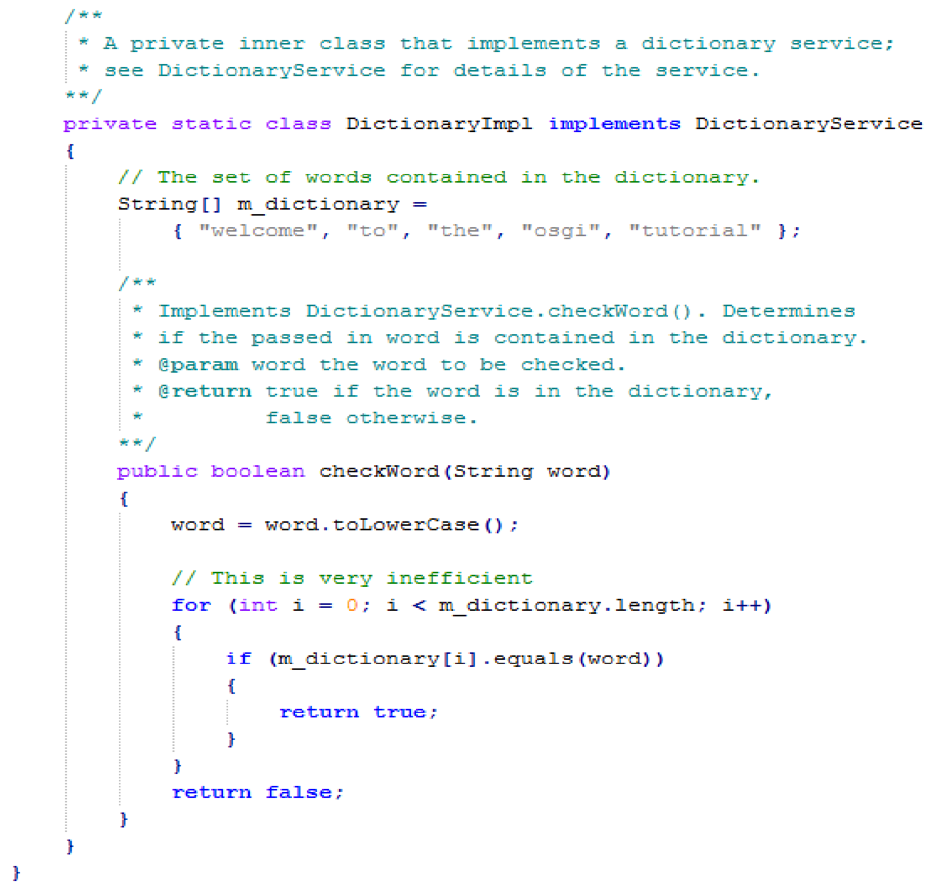


```java
package tutorial.example2;

import java.util.Hashtable;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceListener;
import org.osgi.framework.ServiceEvent;

import tutorial.example2.service.DictionaryService;
/**
 * This class implements a simple bundle that uses the bundle
 * context to register an English language dictionary service
 * with the OSGi framework. The dictionary service interface is
 * defined in a separate class file and is implemented by an
 * inner class.
**/
```



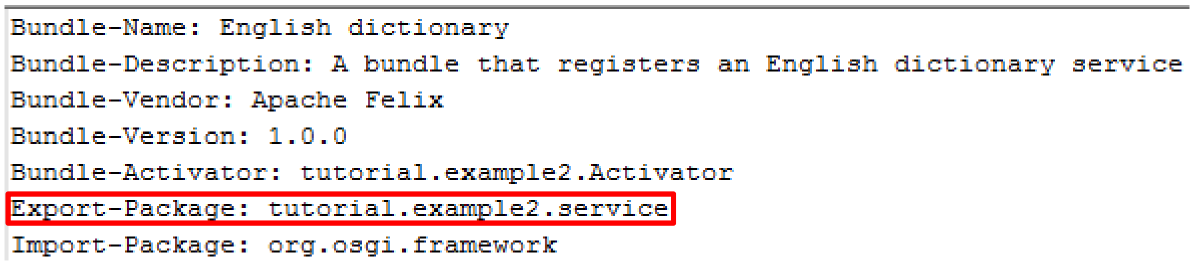
```
public class Activator implements BundleActivator
{
    /**
     * Implements BundleActivator.start(). Registers an
     * instance of a dictionary service using the bundle context;
     * attaches properties to the service that can be queried
     * when performing a service look-up.
     * @param context the framework context for the bundle.
    **/
    public void start(BundleContext context)
    {
        Hashtable<String, String> props = new Hashtable<String, String>();
        props.put("Language", "English");
        context.registerService(
            DictionaryService.class.getName(), new DictionaryImpl(), props);
        System.out.println("Dictionary service registered and started successfully");
    }

    /**
     * Implements BundleActivator.stop(). Does nothing since
     * the framework will automatically unregister any registered services.
     * @param context the framework context for the bundle.
    **/
    public void stop(BundleContext context)
    {
        // NOTE: The service is automatically unregistered.
    }
```

3) You have to create inner class that implements *DictionaryService* interface.



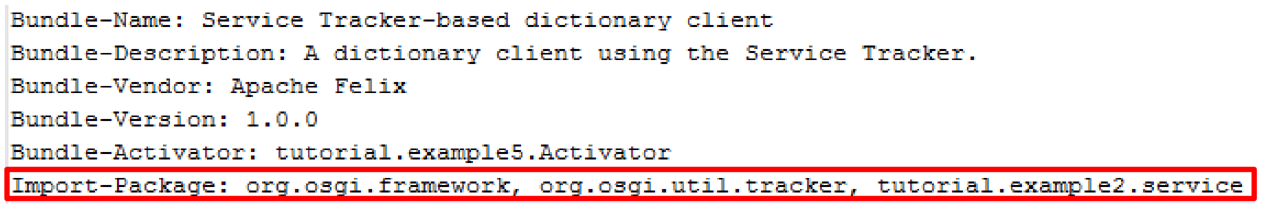
```
    /**
     * A private inner class that implements a dictionary service;
     * see DictionaryService for details of the service.
    **/
    private static class DictionaryImpl implements DictionaryService
    {
        // The set of words contained in the dictionary.
        String[] m_dictionary =
            { "welcome", "to", "the", "osgi", "tutorial" };

        /**
         * Implements DictionaryService.checkWord(). Determines
         * if the passed in word is contained in the dictionary.
         * @param word the word to be checked.
         * @return true if the word is in the dictionary,
         *         false otherwise.
        **/
        public boolean checkWord(String word)
        {
            word = word.toLowerCase();

            // This is very inefficient
            for (int i = 0; i < m_dictionary.length; i++)
            {
                if (m_dictionary[i].equals(word))
                {
                    return true;
                }
            }
            return false;
        }
    }
}
```

4) We must create a manifest.mf file that contains the meta-data for our bundle; the manifest file contains the following (manifest_example2.mf) for Dictionary Service



```
Bundle-Name: English dictionary
Bundle-Description: A bundle that registers an English dictionary service
Bundle-Vendor: Apache Felix
Bundle-Version: 1.0.0
Bundle-Activator: tutorial.example2.Activator
Export-Package: tutorial.example2.service
Import-Package: org.osgi.framework
```

5) Manifest class of consumer is as follow (manifest_example5.mf)



```
Bundle-Name: Service Tracker-based dictionary client
Bundle-Description: A dictionary client using the Service Tracker.
Bundle-Vendor: Apache Felix
Bundle-Version: 1.0.0
Bundle-Activator: tutorial.example5.Activator
Import-Package: org.osgi.framework, org.osgi.util.tracker, tutorial.example2.service
```

6) We specify which class is used to activate our bundle via the Bundle-Activator attribute.

7) Export-Package attribute makes it possible for other bundles to import our dictionary service interface.
8) The Import-Package attribute informs the framework of the bundle's dependencies on external packages.

9) All bundles with an activator must import org.osgi.framework since it contains the core OSGi class definitions.

10) To compile our source code, we need the felix.jar file (found in Felix' bin directory) in our class path.

```
javac -cp C:\Udara\felix-framework-4.0.3\bin\felix.jar *.java service\*.java
```

11) This command compiles all source files and generates class files.

12) After compiling, we need to create a JAR file containing the generated package directories.

13) We will also add our manifest file that contains the bundle's meta-data to the JAR file. To create the JAR file, we issue the command.

```
jar cfm example2.jar manifest.mf -C C:\Udara\tutorial \tutorial\example2
```

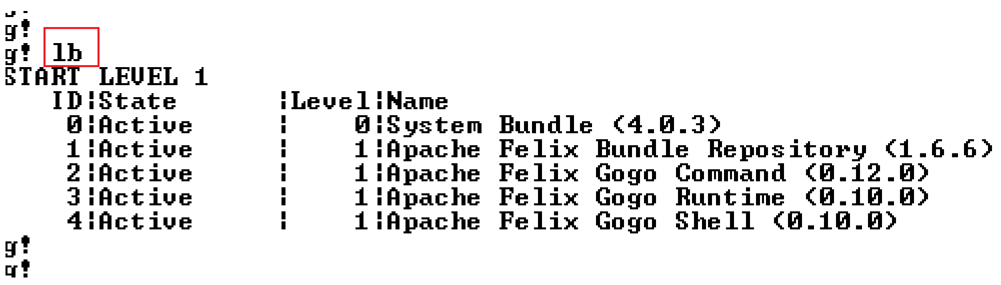
14) Once the JAR file is created, we are ready to install and start the bundle.

Service installation and subscribe.

1) When we start Felix, it asks for a profile name, we will put all of our bundles in a **profile** named **tutorial**.

2) After running Felix, we should make sure that the bundle from Example 1 is active.

3) We can use the Felix lb shell command to get a list of all bundles, their state, and their bundle identifier number.



```
g!
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
g!
g!
```

4) Use below command to Login Apachi Felix shell

   /felix-framework-4.0.3/> java -jar bin/felix.jar

5) If the Example 1 bundle is not active, we should start the bundle using the **start** command
   a. Bundle's identifier number that is displayed by the lb command.
   b. Now we can install and start our dictionary service bundle using bundle number.

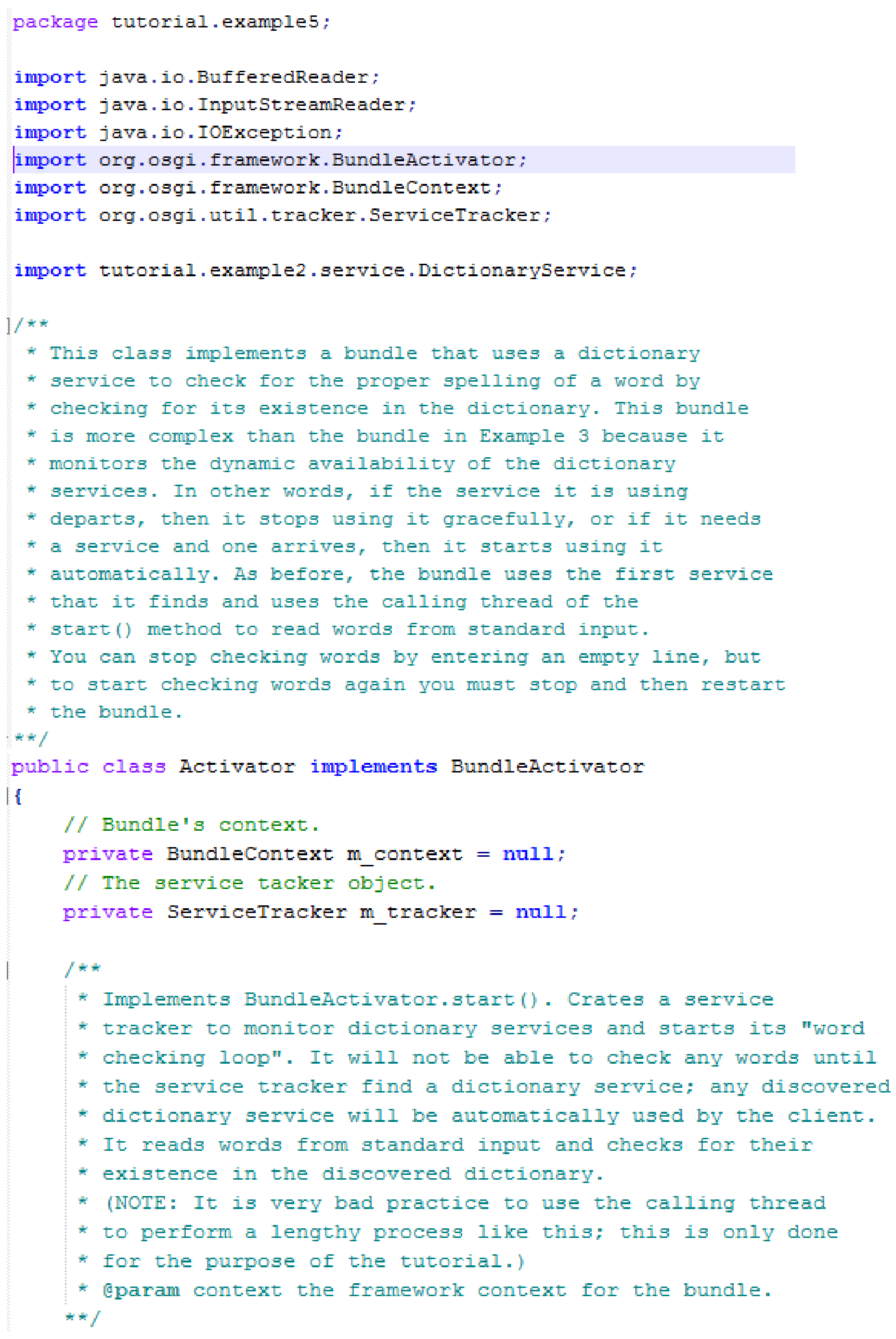
6) Assuming that we created our bundle in the directory c:\tutorial, we can install and start it in Felix' shell using the following command:

```
install file:/C:/Udara/tutorial/example2.jar

start file:/C:/Udara/tutorial/example2.jar
```

7) It is also possible to install and start the bundle in two steps by using the Felix install and start shell commands. To stop the bundle, use the Felix stop shell command.

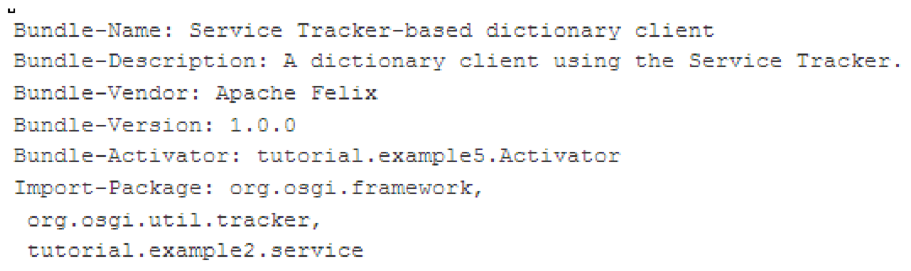
8) To subscribe the service create **Service Tracker Dictionary client bundle** and use the installed service as below.

Service Tracker Dictionary client bundle



```java
package tutorial.example5;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.util.tracker.ServiceTracker;

import tutorial.example2.service.DictionaryService;

/**
 * This class implements a bundle that uses a dictionary
 * service to check for the proper spelling of a word by
 * checking for its existence in the dictionary. This bundle
 * is more complex than the bundle in Example 3 because it
 * monitors the dynamic availability of the dictionary
 * services. In other words, if the service it is using
 * departs, then it stops using it gracefully, or if it needs
 * a service and one arrives, then it starts using it
 * automatically. As before, the bundle uses the first service
 * that it finds and uses the calling thread of the
 * start() method to read words from standard input.
 * You can stop checking words by entering an empty line, but
 * to start checking words again you must stop and then restart
 * the bundle.
**/
public class Activator implements BundleActivator
{
    // Bundle's context.
    private BundleContext m_context = null;
    // The service tacker object.
    private ServiceTracker m_tracker = null;

    /**
     * Implements BundleActivator.start(). Crates a service
     * tracker to monitor dictionary services and starts its "word
     * checking loop". It will not be able to check any words until
     * the service tracker find a dictionary service; any discovered
     * dictionary service will be automatically used by the client.
     * It reads words from standard input and checks for their
     * existence in the discovered dictionary.
     * (NOTE: It is very bad practice to use the calling thread
     * to perform a lengthy process like this; this is only done
     * for the purpose of the tutorial.)
     * @param context the framework context for the bundle.
    **/
public void start(BundleContext context) throws Exception
{
    m_context = context;

    // Create a service tracker to monitor dictionary services.
    m_tracker = new ServiceTracker(
        m_context,
        m_context.createFilter(
            "(&(objectClass=" + DictionaryService.class.getName() + ")" +
            "(Language=*))"),
        null);
    m_tracker.open();

    try
    {
        System.out.println("Enter a blank line to exit.");
        String word = "";
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        // Loop endlessly.
        while (true)
        {
            // Ask the user to enter a word.
            System.out.print("Enter word: ");
            word = in.readLine();

            // Get the selected dictionary service, if available.
            DictionaryService dictionary = (DictionaryService) m_tracker.getService();

            // If the user entered a blank line, then
            // exit the loop.
                if (word.length() == 0)
                {
                    break;
                }
                // If there is no dictionary, then say so.
                else if (dictionary == null)
                {
                    System.out.println("No dictionary available.");
                }
                // Otherwise print whether the word is correct or not.
                else if (dictionary.checkWord(word))
                {
                    System.out.println("Correct.");
                }
                else
                {
                    System.out.println("Incorrect.");
                }
            }
        } catch (Exception ex) { }
    }

    /**
     * Implements BundleActivator.stop(). Does nothing since
     * the framework will automatically unget any used services.
     * @param context the framework context for the bundle.
    **/
    public void stop(BundleContext context)
    {
    }
}
```

1) Create a separate manifest file as below for this service.



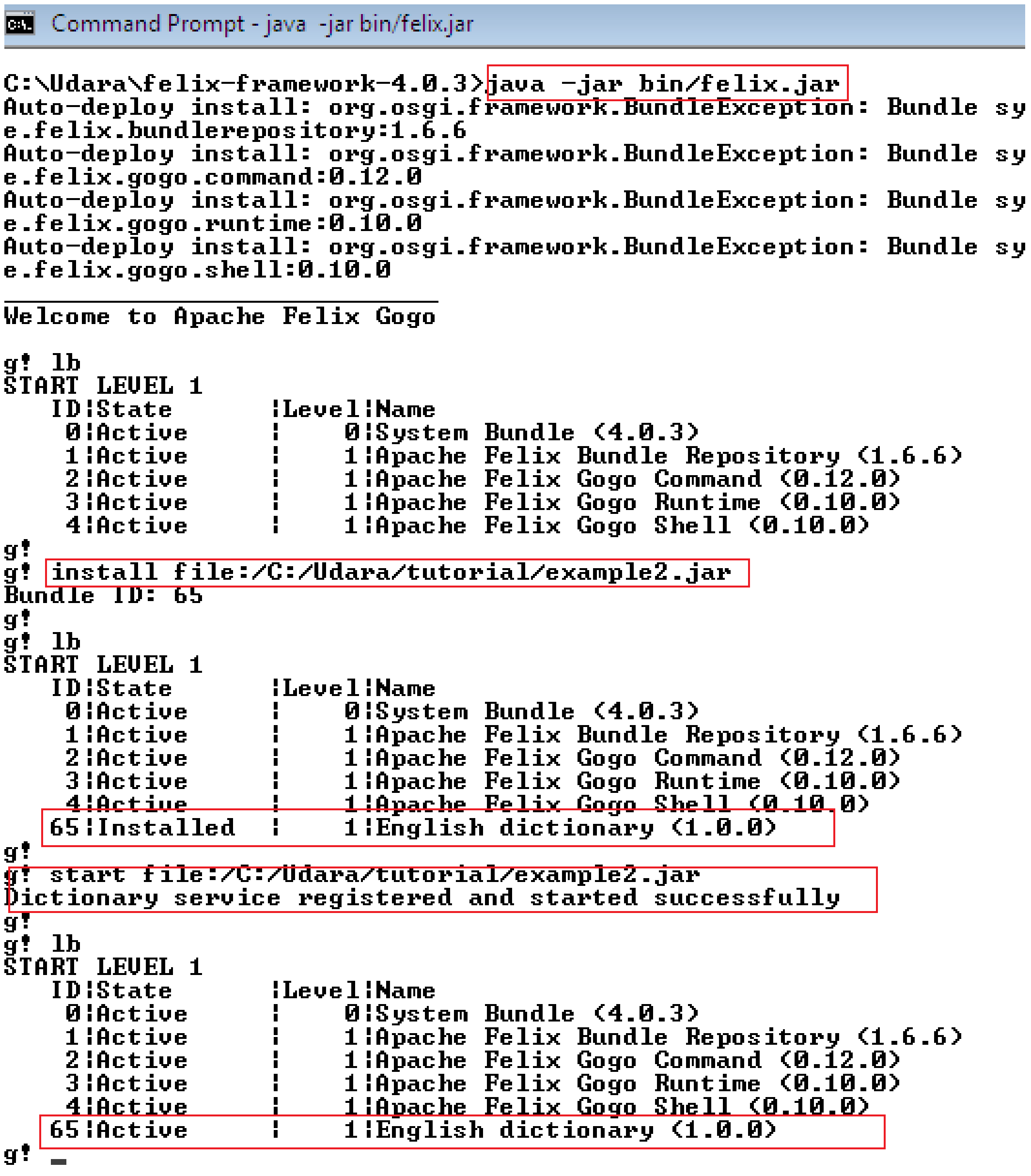
```
Bundle-Name: Service Tracker-based dictionary client
Bundle-Description: A dictionary client using the Service Tracker.
Bundle-Vendor: Apache Felix
Bundle-Version: 1.0.0
Bundle-Activator: tutorial.example5.Activator
Import-Package: org.osgi.framework,
 org.osgi.util.tracker,
 tutorial.example2.service
```

2) You have to import the Dictionary service package you created the in previous example 2 for the manifest.mf file as above.

3) Create a separate bundle with compiling above source file and bundle it as example5.jar.

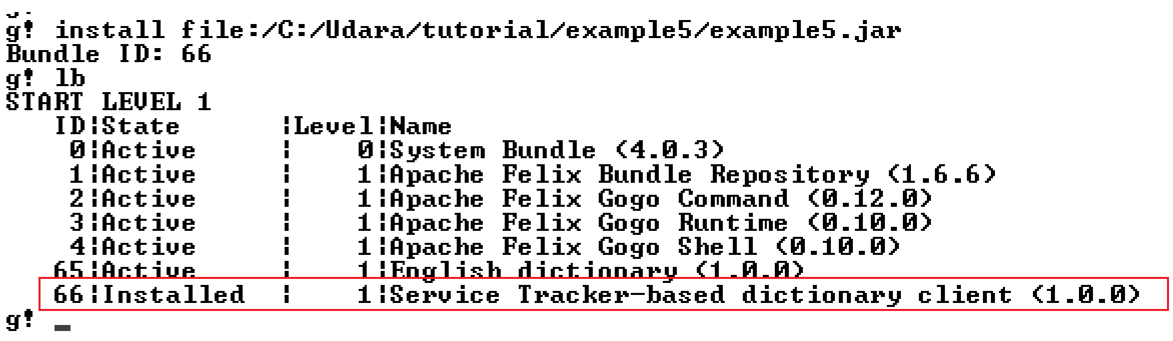
Some screen shots of final out put

1) Start to install services.



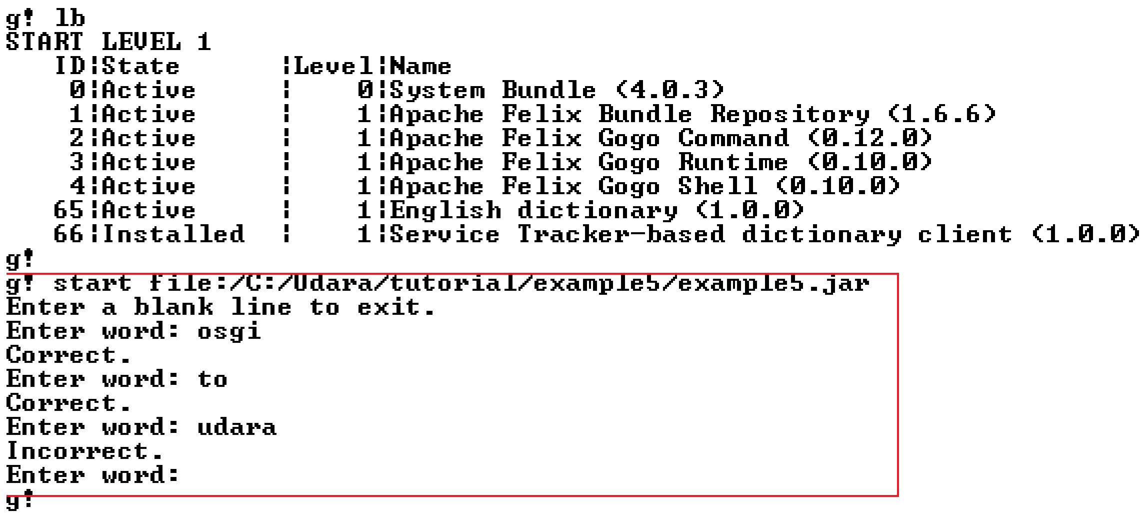
```
C:\Udara\felix-framework-4.0.3>java -jar bin/felix.jar
Auto-deploy install: org.osgi.framework.BundleException: Bundle sy
e.felix.bundlerepository:1.6.6
Auto-deploy install: org.osgi.framework.BundleException: Bundle sy
e.felix.gogo.command:0.12.0
Auto-deploy install: org.osgi.framework.BundleException: Bundle sy
e.felix.gogo.runtime:0.10.0
Auto-deploy install: org.osgi.framework.BundleException: Bundle sy
e.felix.gogo.shell:0.10.0
____________________________
Welcome to Apache Felix Gogo

g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
g!
g! install file:/C:/Udara/tutorial/example2.jar
Bundle ID: 65
g!
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
   65|Installed  |    1|English dictionary (1.0.0)
g!
g! start file:/C:/Udara/tutorial/example2.jar
Dictionary service registered and started successfully
g!
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
   65|Active     |    1|English dictionary (1.0.0)
g! _
```

2) Install client service to subscribe dictionary service.



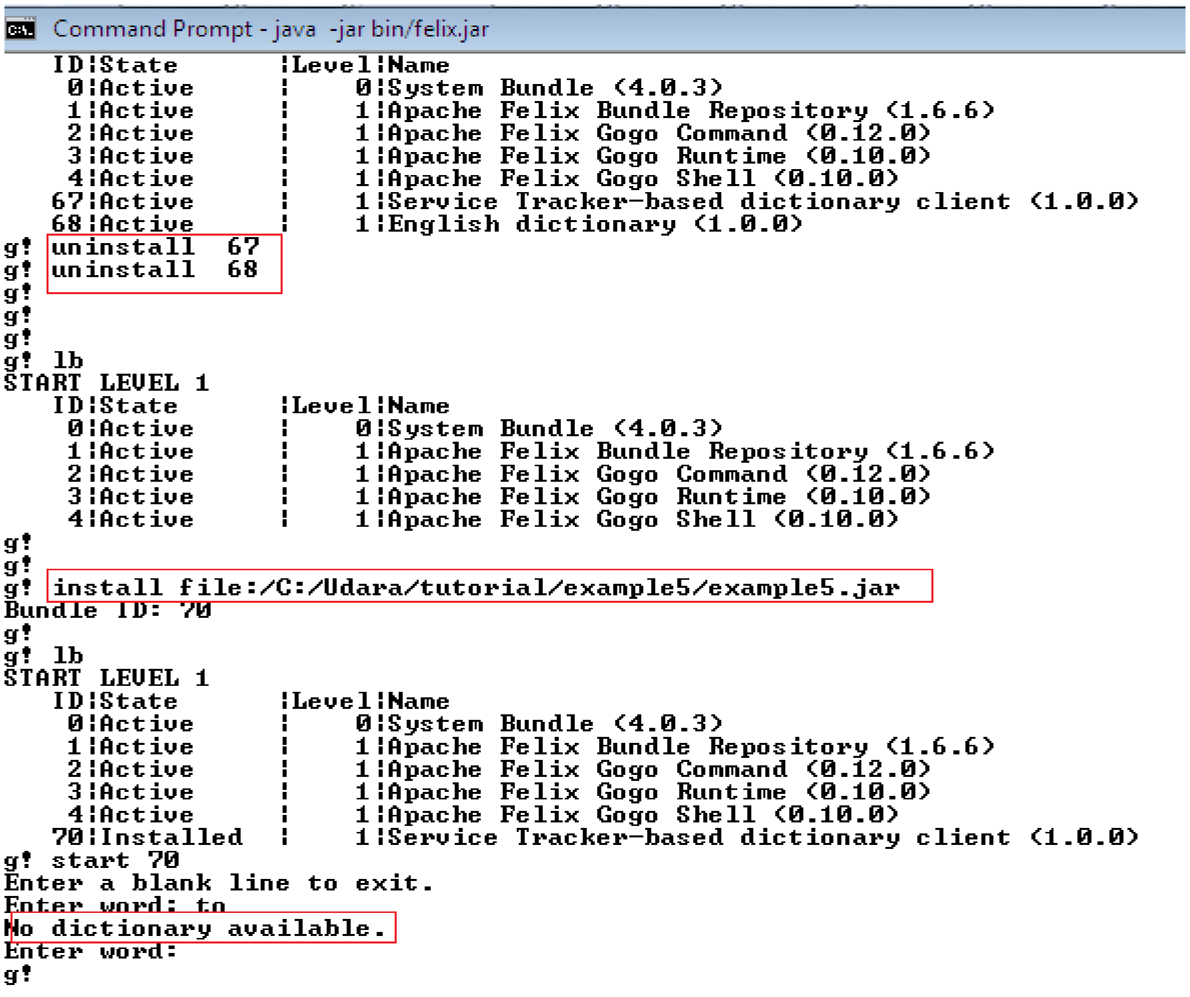
```
g! install file:/C:/Udara/tutorial/example5/example5.jar
Bundle ID: 66
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
   65|Active     |    1|English dictionary (1.0.0)
   66|Installed  |    1|Service Tracker-based dictionary client (1.0.0)
g! _
```

3) Start the client service and subscribe the dictionary service.



```
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (4.0.3)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
    2|Active     |    1|Apache Felix Gogo Command (0.12.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
   65|Active     |    1|English dictionary (1.0.0)
   66|Installed  |    1|Service Tracker-based dictionary client (1.0.0)
g!
g! start file:/C:/Udara/tutorial/example5/example5.jar
Enter a blank line to exit.
Enter word: osgi
Correct.
Enter word: to
Correct.
Enter word: udara
Incorrect.
Enter word:
g!
```

4) Unregister the dictionary service and client attempt to access the service.



Command Prompt - java -jar bin/felix.jar

```
    ID|State      |Level|Name
     0|Active     |    0|System Bundle (4.0.3)
     1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
     2|Active     |    1|Apache Felix Gogo Command (0.12.0)
     3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
     4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
    67|Active     |    1|Service Tracker-based dictionary client (1.0.0)
    68|Active     |    1|English dictionary (1.0.0)
g! uninstall  67
g! uninstall  68
g!
g!
g!
g! lb
START LEVEL 1
    ID|State      |Level|Name
     0|Active     |    0|System Bundle (4.0.3)
     1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
     2|Active     |    1|Apache Felix Gogo Command (0.12.0)
     3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
     4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
g!
g!
g! install file:/C:/Udara/tutorial/example5/example5.jar
Bundle ID: 70
g!
g! lb
START LEVEL 1
    ID|State      |Level|Name
     0|Active     |    0|System Bundle (4.0.3)
     1|Active     |    1|Apache Felix Bundle Repository (1.6.6)
     2|Active     |    1|Apache Felix Gogo Command (0.12.0)
     3|Active     |    1|Apache Felix Gogo Runtime (0.10.0)
     4|Active     |    1|Apache Felix Gogo Shell (0.10.0)
    70|Installed  |    1|Service Tracker-based dictionary client (1.0.0)
g! start 70
Enter a blank line to exit.
Enter word: to
No dictionary available.
Enter word:
g!
```