

```

/**
 * Class Name      : Registry.java
 * Version         : Build 01.00
 * Date of Change  : 2000 MAR 11
 * Author          : Don Baldwin
 * Copyright       : 2000 ASR Strategic Resources
 * Description     : This class manages the registry of addresses currently
 *                  loaded from the database.
 *
 *
 * Version History:
 *
 * Version         Date           Author
 * Build 01.00    2000 Mar 11    Ulf Hedlund
 *
 *                  Started the class.
 *
 * Known Problems:None
 */

```

```

import java.util.Vector;
import java.util.Enumeration;
import java.sql.SQLException;

```

```

public class Registry
{
    private static Vector    vecAddresses;
    private static Registry objRegistry = null;

    /**
     * Hiding this to make sure people receive the Registry
     * object through the getRegistry() method.
     */
    private Registry()
    {
        vecAddresses = new Vector();
    }

    /**
     * Returns the registry. Creates one first if the
     * method is called for the first time.
     */
    public static Registry getRegistry()
    {
        if(objRegistry == null)
        {
            objRegistry = new Registry();
        }

        return objRegistry;
    }

    /**
     * Adds the passed Address to the list.
     */
    public static void newAddress(Address anObjAddress)
    {
        vecAddresses.addElement(anObjAddress);
    }

    /**
     * Removes the old address from the list, then adds the
     * new address to it.
     */
    public static void updateAddress(Address anObjAddress)
    {

```

```

    long    longId;

    longId    = anObjAddress.getId();

    removeAddress(longId);
    newAddress(anObjAddress);
}

/**
 * Removes the address with the passed id from the list.
 */
public static void removeAddress(long aLongId)
{
    Enumeration    allAddresses;

    allAddresses    = vecAddresses.elements();

    // Search for an address with matching id.
    while(allAddresses.hasMoreElements())
    {
        Address objTempAddress;
        long    longId;

        objTempAddress = (Address)allAddresses.nextElement();
        longId    = objTempAddress.getId();

        // If we found one, remove it and return.
        if(longId == aLongId)
        {
            vecAddresses.removeElement(objTempAddress);
            return;
        }
    }
}

/**
 * Sets the registry to all addresses in the DB.
 */
public static void readAllAddresses()
{
    try
    {
        vecAddresses = Address.getAllAddresses();
    }
    catch(SQLException excSQL)
    {
        System.out.println("SQL Exception: "+excSQL);
    }
}

/**
 * Returns all addresses in the list.
 */
public static Vector getAddressess()
{
    return vecAddresses;
}

/**
 * Sets the address list to be equal to the passed vector.
 */
public static void setAddressess(Vector aVecAddresses)
{

```

```
vecAddresses = aVecAddresses;
}

/**
 * Find an address with the passed id, then return it.
 */
public static Address getAddress(long aLongId)
{
    Enumeration    allAddresses;

    allAddresses = vecAddresses.elements();

    while(allAddresses.hasMoreElements())
    {
        Address objAddress = (Address)allAddresses.nextElement();

        if(objAddress.getId() == aLongId)
        {
            return objAddress;
        }
    }
    return null;
}

} // END Registry.java
```