```
package rolodex;

import java.io.*;

/****
 *
 * Class Name is the heavyweight translation of the RSL atomic object
 * definition
 *                                                                   <pre>
 *     object Name is string;</tt>                                  </pre>
 *
 * "Heavy-weight" means that Name is a complete class, with a constructor,
 * string conversion method, and equals method.
 *                                                                     <p>
 * A lighter-weight class definition would have just a single public string
 * data member, with no methods.  Such a definition violates our convention of
 * no public data members.  A wimped-out translation would be to eliminate the
 * mnemonic identifier "Name" entirely from the design, replacing it directly
 * with type String.  This definition violates our convention of design-to-spec
 * traceability.
 *                                                                     <p>
 * The problem we're having here is one of RSL-to-Java translation.  The
 * problem is due to the lack a simple type definition construct in Java, such
 * as "typedef" in C/C++ or "type" in Pascal-class languages.  Such a construct
 * allows the programmer to equate a mnemonic name with an existing type.
 * E.g.,
 *                                                                   <pre>
 *     typedef String Name;                                         </pre>
 *
 * equates the identifier "Name" with the type identifier "String", such that
 * "Name" becomes a synonym for "String".  Absent such a construct in Java,
 * we're stuck with defining Name as a class or losing Name altogether by
 * systematically replacing it with String.
 *                                                                     <p>
 * A nice solution to this problem would be to define Name simply as
 *                                                                   <pre>
 *     class Name extends String{};                                 </pre>
 *
 * This would provide precisely the definition of Name that we're after here.
 * Unfortunately (and <em>aggravatingly</em>), class String is defined as
 * final, which precludes extending it.
 *
 */

public class Name implements Serializable {
    public Name(String value) {this.value = new String(value);}
    public String stringValue() {return value;}
    public boolean equals(Name n) {return value.equals(n.value);}
    protected String value;
}
```