JBND Client-Side _Entity.java Template

This template requires the JBND library or source code be included on the Client-Side of the application. NSArray is not parameterized so if you want to use it with 5.4, slight modifications might be beneficial.

```
// <$GEN_PREFIX$><$classNameWithoutPackage$>.java
//
// Created by eogenerator
//
// DO NOT EDIT.  Make changes to <$classNameWithoutPackage$>.java instead.
//
// This EO was generated to perform standard JBND behaviors
//        1. All setter methods will not set new values if they are identical to old values.
//        2. All setter methods invoke EOFDataObject.fireDataObjectEvent(String key, Object oldValue).
//        3. All attribute and relationship keys are stored as static final Strings.
//        4. A default constructor is provided to automatically call the superclass constructor
//        that accepts an EOClassDescription argument, used to avoid bugs in the client side EOF.

#if ($entity.superclassPackageName)
package $entity.superclassPackageName;
#end

import com.webobjects.foundation.*;
import com.webobjects.eocontrol.*;
import java.math.BigDecimal;
import java.util.*;
import org.jbnd.support.JBNDUtil;
import org.jbnd.eof.EOFDataObject;
import org.jbnd.event.DataObjectEvent;
import org.jbnd.event.DataObjectEvent.Type;

/**
 * DO NOT EDIT!
 */
@SuppressWarnings("all")
public abstract class ${entity.prefixClassNameWithoutPackage} extends #if ($entity.parentSet)${entity.parent.
classNameWithDefault}#elseif ($EOGenericRecord)${EOGenericRecord}#else EOFDataObject#end {

        /** Entity name */
        public static final String ENTITY_NAME = "$entity.name";

        /** Attribute keys */
        #foreach ($attribute in $entity.sortedClientClassAttributes)
        public static final String ${attribute.uppercaseUnderscoreName}_KEY = "$attribute.name";
        #end

        /** Relationship keys */
        #foreach ($relationship in $entity.sortedClientClassRelationships)
        public static final String ${relationship.uppercaseUnderscoreName}_KEY = "$relationship.name";
        #end

        /**
         * Constructor. Creates an EO with the <tt>EOClassDescription</tt> of the
         * relevant entity, to ensure a proper client side behavior.
         */
        public ${entity.prefixClassNameWithoutPackage}() {
                super(EOClassDescription.classDescriptionForEntityName("$entity.name"));
        }

        /**
         * Constructor used in inheritance situations, when this entity is the parent.
         *
         * @param cd The class description of the sub-entity.
         */
        protected ${entity.prefixClassNameWithoutPackage}(EOClassDescription cd) {
                super(cd);
        }

#foreach ($attribute in $entity.sortedClientClassAttributes)
```

```
#if (!$attribute.inherited)
        public $attribute.javaClassName ${attribute.name}() {
                return ($attribute.javaClassName)storedValueForKey(${attribute.uppercaseUnderscoreName}_KEY);
        }

        public void set${attribute.capitalizedName}($attribute.javaClassName aValue) {
                // extract old value
                $attribute.javaClassName oldValue = ${attribute.name}();

                // do not perform EOF work unless values are different
                if(JBNDUtil.equals(oldValue, aValue)) return;

                takeStoredValueForKey(aValue, ${attribute.uppercaseUnderscoreName}_KEY);
                fireDataObjectEvent(${attribute.uppercaseUnderscoreName}_KEY, oldValue, DataObjectEvent.Type.
ATTRIBUTE_CHANGE);
        }
#end
#end

#foreach ($relationship in $entity.sortedClientClassToOneRelationships)
#if (!$relationship.inherited)
        public $relationship.actualDestination.classNameWithDefault ${relationship.name}() {
                return ($relationship.actualDestination.classNameWithDefault)storedValueForKey(${relationship.
uppercaseUnderscoreName}_KEY);
        }

        public void set${relationship.capitalizedName}($relationship.actualDestination.classNameWithDefault
aValue) {
                // get old value
                $relationship.actualDestination.classNameWithDefault oldValue = ${relationship.name}();

                // do not perform EOF work unless values are different
                if(JBNDUtil.equals(oldValue, aValue)) return;

                takeStoredValueForKey(aValue, ${relationship.uppercaseUnderscoreName}_KEY);
                fireDataObjectEvent(${relationship.uppercaseUnderscoreName}_KEY, oldValue, DataObjectEvent.Type.
TO_ONE_CHANGE);
        }
#end
#end

#foreach ($relationship in $entity.sortedClientClassToManyRelationships)
#if (!$relationship.inherited)
        public NSArray<${relationship.actualDestination.classNameWithDefault}> ${relationship.name}() {
                return (NSArray<${relationship.actualDestination.classNameWithDefault}>)storedValueForKey
(${relationship.uppercaseUnderscoreName}_KEY);
        }

        public void set${relationship.capitalizedName}(NSArray<${relationship.actualDestination.
classNameWithDefault}> aValue) {
                Object oldValue = ${relationship.name}().clone();
                takeStoredValueForKey(aValue, ${relationship.uppercaseUnderscoreName}_KEY);
                fireDataObjectEvent(${relationship.uppercaseUnderscoreName}_KEY, oldValue, DataObjectEvent.Type.
TO_MANY_CHANGE);
        }

        public void addTo${relationship.capitalizedName}(${relationship.actualDestination.classNameWithDefault}
object) {
                if(${relationship.name}().contains(object)) return;
                includeObjectIntoPropertyWithKey(object, ${relationship.uppercaseUnderscoreName}_KEY);
                fireDataObjectEvent(${relationship.uppercaseUnderscoreName}_KEY, object, DataObjectEvent.Type.
TO_MANY_ADD);
        }

        public void removeFrom${relationship.capitalizedName}(${relationship.actualDestination.
classNameWithDefault} object) {
                if(!${relationship.name}().contains(object)) return;
                excludeObjectFromPropertyWithKey(object, ${relationship.uppercaseUnderscoreName}_KEY);
                fireDataObjectEvent(${relationship.uppercaseUnderscoreName}_KEY, object, DataObjectEvent.Type.
TO_MANY_REMOVE);
        }
```

```
#end
#end
}
```