```java
/*
 * AbstractController.java
 *
 * Created on January 22, 2007, 8:41 AM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package Controlador;

import Modelo2.AbstractModel;
import Vista2.AbstractViewPanel;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.lang.reflect.Method;
import java.util.ArrayList;

/**
 * This class provides base level functionality for each controller. This includes the
 * ability to register multiple models and views, propogating model change events to
 * each of the views, and providing a utility function to broadcast model property
 * changes when necessary.
 * @author Robert Eckstein
 */
public abstract class AbstractController implements PropertyChangeListener {

    //  Vectors that hold a list of the registered models and views for this controller.

    private ArrayList<AbstractViewPanel> registeredViews;
    private ArrayList<AbstractModel> registeredModels;


    /** Creates a new instance of Controller */
    public AbstractController() {
        registeredViews = new ArrayList<AbstractViewPanel>();
        registeredModels = new ArrayList<AbstractModel>();
    }


    /**
     * Binds a model to this controller. Once added, the controller will listen for all
     * model property changes and propogate them on to registered views. In addition,
     * it is also responsible for resetting the model properties when a view changes
     * state.
     * @param model The model to be added
     */
    public void addModel(AbstractModel model) {
        registeredModels.add(model);
        model.addPropertyChangeListener(this);
    }

    /**
     * Unbinds a model from this controller.
     * @param model The model to be removed
     */
    public void removeModel(AbstractModel model) {
        registeredModels.remove(model);
        model.removePropertyChangeListener(this);
    }


    /**
     * Binds a view to this controller. The controller will propogate all model property
     * changes to each view for consideration.
     * @param view The view to be added
     */
    public void addView(AbstractViewPanel view) {
        registeredViews.add(view);
    }

    /**
```

```
     * Unbinds a view from this controller.
     * @param view The view to be removed
     */
    public void removeView(AbstractViewPanel view) {
        registeredViews.remove(view);
    }



    //  Used to observe property changes from registered models and propogate
    //  them on to all the views.

    /**
     * This method is used to implement the PropertyChangeListener interface. Any model
     * changes will be sent to this controller through the use of this method.
     * @param evt An object that describes the model's property change.
     */
    public void propertyChange(PropertyChangeEvent evt) {
        System.out.println("EN CONTROLADOR:"+evt    );

        for (AbstractViewPanel view: registeredViews) {
            view.modelPropertyChange(evt);
        }
    }


    /**
     * Convienence method that subclasses can call upon to fire off property changes
     * back to the models. This method used reflection to inspect each of the model
     * classes to determine if it is the owner of the property in question. If it
     * isn't, a NoSuchMethodException is throws (which the method ignores).
     *
     * @param propertyName The name of the property
     * @param newValue An object that represents the new value of the property.
     */
    protected void setModelProperty(String propertyName, Object newValue) {

        for (AbstractModel model: registeredModels) {
            try {

                Method method = model.getClass().
                    getMethod("set"+propertyName, new Class[] {
                                                    newValue.getClass()
                                                }
                              );

                System.out.println(method);
                method.invoke(model, newValue);

            } catch (Exception ex) {
                //  Handle exception
            }
        }
    }


}
```