```
package rolodex;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/****
 *
 * Class OKConfirmChangeButtonListener is attached to the OK button in the
 * confirm change dialog.  It is an example of referential transparency of
 * design.  See the documentation for the <a href="OKAddButtonListener.html">
 * OKAddButtonListener </a> for discussion of the concept of referential
 * transparency in an object-oriented design.
 *
 */

public class OKConfirmChangeButtonListener extends OKAddButtonListener {

    /**
     * Construct this with the given rolodex model and parent dialog view and
     * grandparent rolodex view.  Access to the model is for calling its change
     * function.  Access to the parent dialog is for gathering data to be sent
     * to Rolodex.change.  Access to the grandparent view is for reporting
     * input error messages.  See the specs of actionPerformed below for
     * further discussion.
     */
    public OKConfirmChangeButtonListener(Rolodex r, ConfirmChangeDialog cncd,
            RolodexMenuUI rmui) {

        /*
         * Call the parent constructor.
         */
        super(r, cncd, rmui);

        /*
         * For convenience, save a downcast version of parent view.
         */
        this.cncd = (ConfirmChangeDialog) acd;

    }

    /**
     * Respond to a press of the OK button by calling Rolodex.change.  The id
     * of the card to change is obtained from the parent ConfirmChangeDialog.
     * This id was saved from the original change dialog (see
     * <href="ConfirmChangeDialog.html#showCardAndSaveId(Card c, Id id)">
     * ConfirmChangeDialog.showCardAndSaveId() </a>.  The card data sent as the
     * second parameter to Rolodex.change are gathered from the JTextFields for
     * the card fields.  These text fields are also obtained from the parent
     * ConfirmChangeDialog.
     */
    public void actionPerformed(ActionEvent e) {

        try {
            r.change(cncd.getOriginalId(), new Card(
                new Name(cncd.getName()),       // Card name as a String
                new Id(cncd.getId()),           // Card id as an Integer
                new Age(cncd.getAge()),         // Card age as an Integer
                new Sex(cncd.getSex()),         // Sex as an enum literal
                new Address(cncd.getAddr())     // Address as a String
            ));
        }
        catch (ChangeInputErrors cie) {
            /*
             * Display zero or more error messages, as appropriate.
             */
            rmui.showErrorDialog(cie.getErrorList());
        }
        catch (AddInputErrors aie) {
            /*
             * Display zero or more error messages, as appropriate.
             */
            rmui.showErrorDialog(aie.getErrorList());
        }
        catch (DeleteInputError die) {
            /*
             * This cannot happen given validity checking logic of
             * Rolodex.change(), q.v.  Viz., the only way DeleteInputError will
             * be thrown is if Rolodex.delete is called with the id of a card
             * that is not in the rolodex.  However, this cannot happen here,
             * because the precond checking of Rolodex.change ensures that a
             * card of the given id is in the rolodex before delete is called.
             */
        }

        /*
         * Notify the Rolodex observers if it has changed.  Per the specs of
         * the Java Observable class, notifyObservers only performs the
         * notification if the observed object has actually changed.
         */
        r.notifyObservers();

    }

    /** The parent view */
    protected ConfirmChangeDialog cncd;
}
```