MouserAdapter

```
this.tableGadget.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseReleased(MouseEvent e) { }

        @Override
        public void mousePressed(MouseEvent e) {
                if (e.getClickCount() == 2)
                        editSelectedGadget();
        }

        @Override
        public void mouseExited(MouseEvent e) { }

        @Override
        public void mouseEntered(MouseEvent e) { }

        @Override
        public void mouseClicked(MouseEvent e) { }
});
```

DocumentListener

```
this.textField.getDocument().addDocumentListener(new DocumentListener() {
        @Override
        public void removeUpdate(DocumentEvent e) {
                if (textField.getText().length() == 0)
                        sorter.setRowFilter(null);
        }

        @Override
        public void insertUpdate(DocumentEvent e) {
                sorter.setRowFilter(RowFilter.regexFilter(textField.getText()));
        }

        @Override
        public void changedUpdate(DocumentEvent e) { }
});
```

ActionListener

```
this.btnGadget.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
                GadgetDetailDialog dialog = new GadgetDetailDialog(frame, null, library);
                dialog.setVisible(true);
        }
});
```

Observable

```
public class X extends Observable {
        private String eier;

        public String getEier() {
                return this.eier;
        }

        public void setEier(String eier) {
                this.eier = eier;
                setChanged();
                nofityObservers("optional; Eier changed");
        }
}
```

TableModel / Observer

```
public class GadgetTableModel extends AbstractTableModel implements Observer {
        private static final long serialVersionUID = 1L;
        private Library library;
        private List<Gadget> gadgets;
        private String[] columns = {"Inventar-Nr.", "Name", "Hersteller",
                                                                "Preis", "Zustand"};

        public GadgetTableModel(Library library) {
                this.library = library;
                this.gadgets = this.library.getGadgets();
                this.library.addObserver(this);
        }

        @Override
        public int getRowCount() {
                return this.gadgets.size();
        }

        @Override
        public int getColumnCount() {
                return columns.length;
        }

        @Override
        public Object getValueAt(int rowIndex, int columnIndex) {
                Gadget gadget = this.gadgets.get(rowIndex);
                switch(columnIndex) {
                case 0: return gadget.getInventoryNumber();
                case 1: return gadget.getName();
                case 2: return gadget.getManufacturer();
                case 3: return gadget.getPrice();
                case 4: return gadget.getCondition();
                case 5: return gadget.getCondition();
                default: return null;
                }
        }

        @Override
        public String getColumnName(int columnIndex) {
                return this.columns[columnIndex];
        }

        @Override
        public void update(Observable o, Object arg) {
                if ("gadgetadd".equals(arg)) {
                        fireTableRowsInserted(getRowCount() - 2, getRowCount() - 1);
                } else {
                        fireTableCellUpdated(getRowCount() - 1, 0);
                }
        }
}
```