Klassen, Vererbung, Benutzereingabe

Rolf Hennicker, Annabelle Klarl

Zentralübung zur Vorlesung
„Einführung in die Informatik: Programmierung und Softwareentwicklung“

`http://www.pst.ifi.lmu.de/Lehre/wise-12-13/infoeinf`

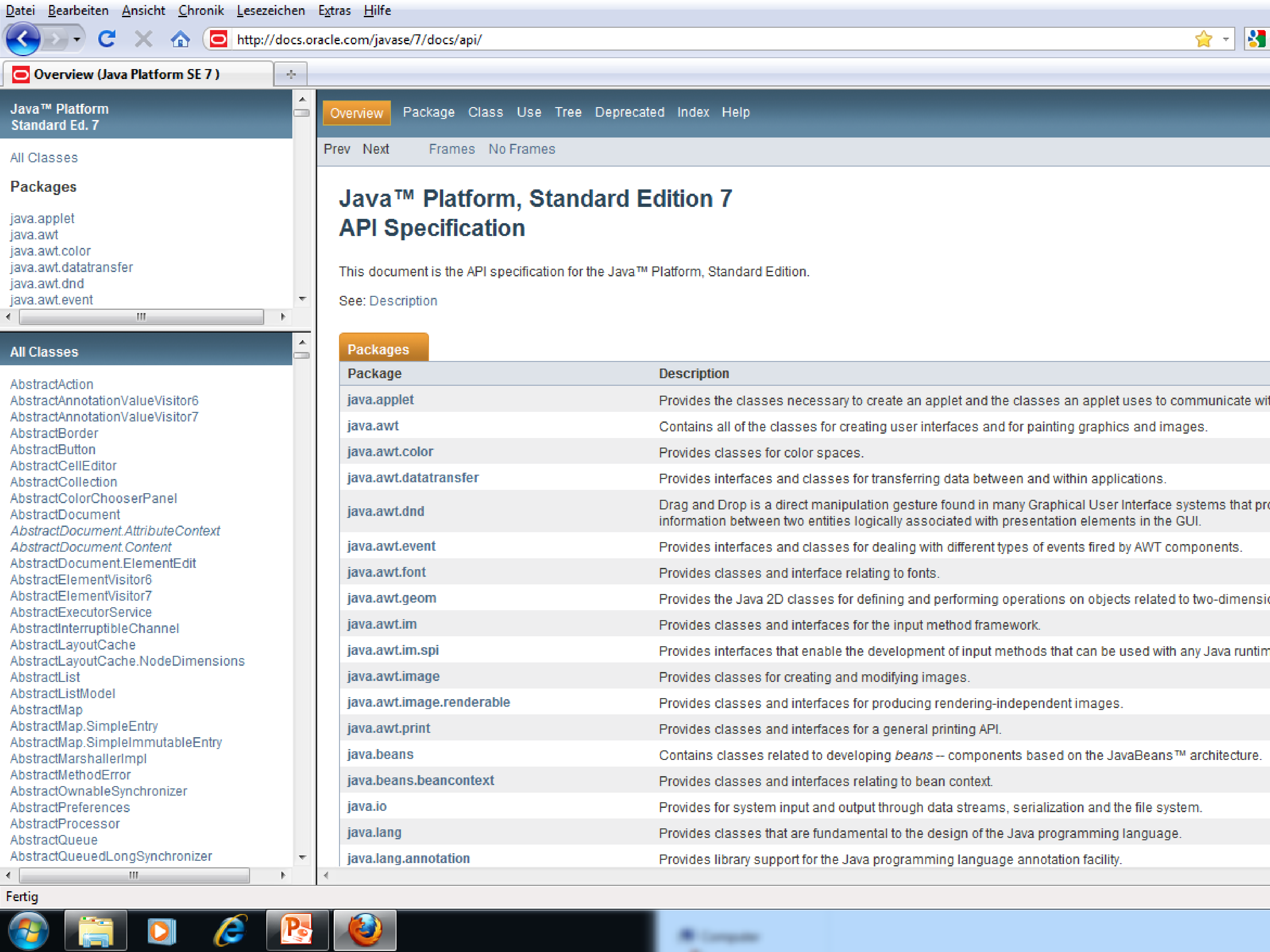
WS12/13

Aufgabe 1

Die bisherigen Anwendungen in den Übungen können verschiedene Dinge berechnen – aber sie sind noch nicht interaktiv. Wie kann eine **grafische Benutzereingabe** auf einfache Weise realisiert werden?

Bisher:

```
public static void main(String[] args) {
    int a = 2;
    int b = 5;

    int produkt = a * b;
    System.out.println("Produkt von a und b: " + produkt);
}
```


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://docs.oracle.com/javase/7/docs/api/
Overview (Java Platform SE 7)
Overview Package Class Use Tree Deprecated Index Help
Prev Next Frames No Frames
Fertig

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableEntry
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchronizer

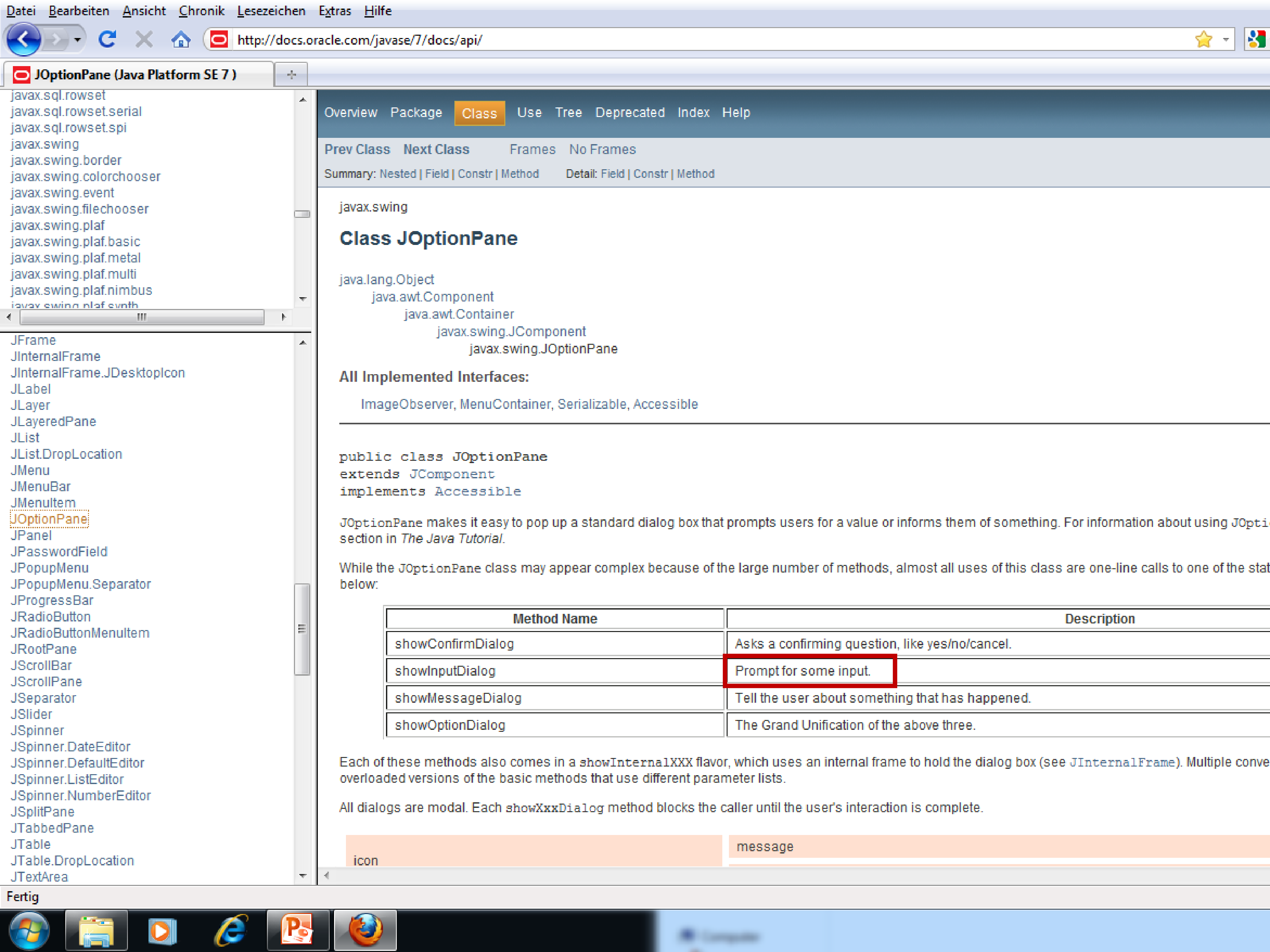
Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate wit
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that pro information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensio
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtim
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://docs.oracle.com/javase/7/docs/api/
JOptionPane (Java Platform SE 7)
Fertig

javax.sql.rowset
javax.sql.rowset.serial
javax.sql.rowset.spi
javax.swing
javax.swing.border
javax.swing.colorchooser
javax.swing.event
javax.swing.filechooser
javax.swing.plaf
javax.swing.plaf.basic
javax.swing.plaf.metal
javax.swing.plaf.multi
javax.swing.plaf.nimbus

JFrame
JInternalFrame
JInternalFrame.JDesktopIcon
JLabel
JLayer
JLayeredPane
JList
JList.DropLocation
JMenu
JMenuBar
JMenuItem
JOptionPane
JPanel
JPasswordField
JPopupMenu
JPopupMenu.Separator
JProgressBar
JRadioButton
JRadioButtonMenuItem
JRootPane
JScrollBar
JScrollPane
JSeparator
JSlider
JSpinner
JSpinner.DateEditor
JSpinner.DefaultEditor
JSpinner.ListEditor
JSpinner.NumberEditor
JSplitPane
JTabbedPane
JTable
JTable.DropLocation
JTextArea

Overview Package Class Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

javax.swing

Class JOptionPane

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JOptionPane

All Implemented Interfaces:

ImageObserver, MenuContainer, Serializable, Accessible

```
public class JOptionPane
extends JComponent
implements Accessible
```

`JOptionPane` makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. For information about using `JOpti`
section in *The Java Tutorial*.

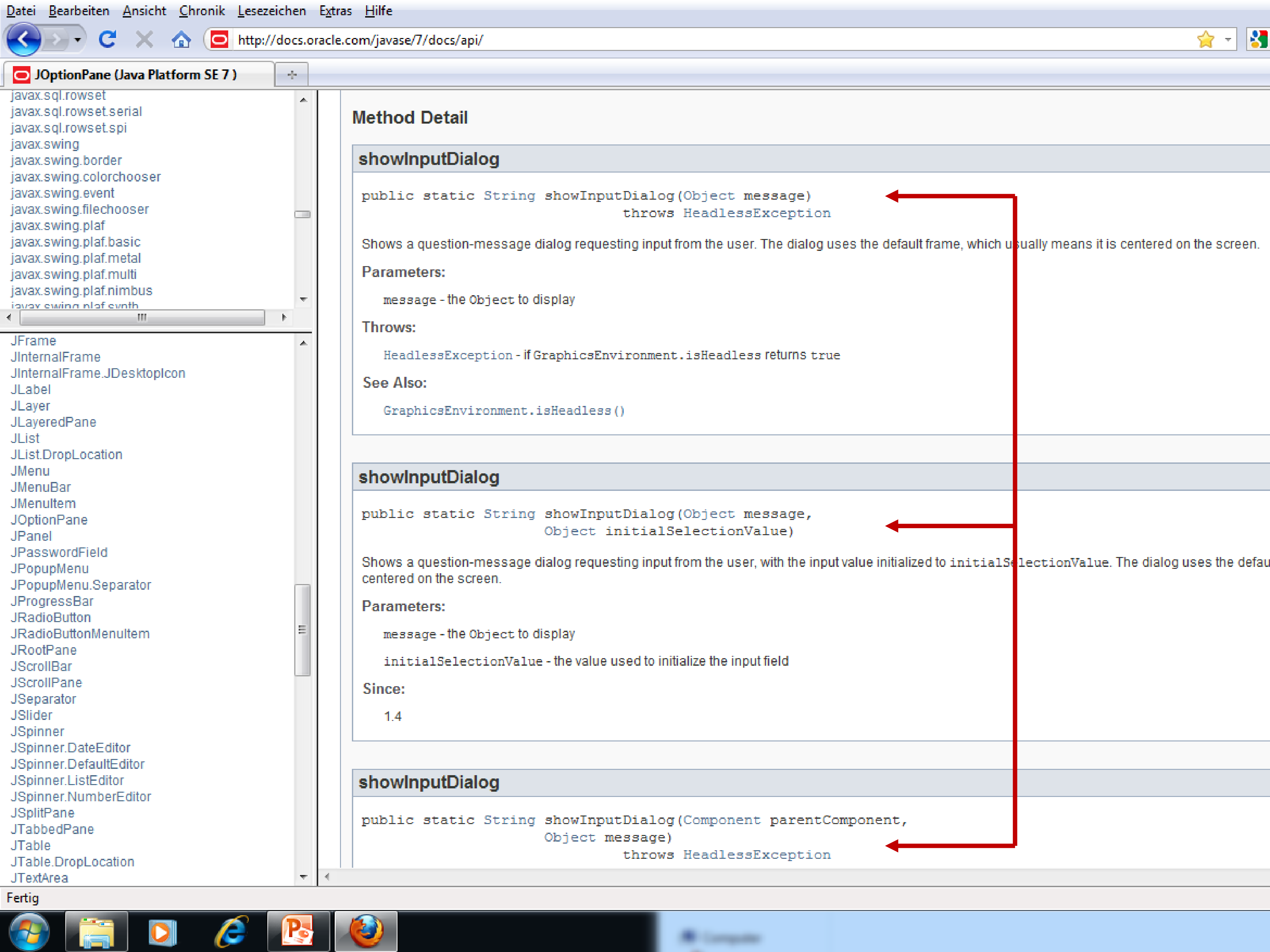
While the `JOptionPane` class may appear complex because of the large number of methods, almost all uses of this class are one-line calls to one of the stat
below:

Method Name	Description
showConfirmDialog	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

Each of these methods also comes in a `showInternalXXX` flavor, which uses an internal frame to hold the dialog box (see `JInternalFrame`). Multiple conve
overloaded versions of the basic methods that use different parameter lists.

All dialogs are modal. Each `showXxxDialog` method blocks the caller until the user's interaction is complete.

icon	message


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://docs.oracle.com/javase/7/docs/api/
JOptionPane (Java Platform SE 7)
Fertig

javax.sql.rowset
javax.sql.rowset.serial
javax.sql.rowset.spi
javax.swing
javax.swing.border
javax.swing.colorchooser
javax.swing.event
javax.swing.filechooser
javax.swing.plaf
javax.swing.plaf.basic
javax.swing.plaf.metal
javax.swing.plaf.multi
javax.swing.plaf.nimbus
javax.swing.plaf.synth

JFrame
JInternalFrame
JInternalFrame.JDesktopIcon
JLabel
JLayer
JLayeredPane
JList
JList.DropLocation
JMenu
JMenuBar
JMenuItem
JOptionPane
JPanel
JPasswordField
JPopupMenu
JPopupMenu.Separator
JProgressBar
JRadioButton
JRadioButtonMenuItem
JRootPane
JScrollBar
JScrollPane
JSeparator
JSlider
JSpinner
JSpinner.DateEditor
JSpinner.DefaultEditor
JSpinner.ListEditor
JSpinner.NumberEditor
JSplitPane
JTabbedPane
JTable
JTable.DropLocation
JTextArea

Method Detail

showInputDialog

```
public static String showInputDialog(Object message)
                              throws HeadlessException
```

Shows a question-message dialog requesting input from the user. The dialog uses the default frame, which usually means it is centered on the screen.

Parameters:

`message` - the `Object` to display

Throws:

`HeadlessException` - if `GraphicsEnvironment.isHeadless` returns `true`

See Also:

`GraphicsEnvironment.isHeadless()`

showInputDialog

```
public static String showInputDialog(Object message,
                                     Object initialSelectionValue)
```

Shows a question-message dialog requesting input from the user, with the input value initialized to `initialSelectionValue`. The dialog uses the defau
centered on the screen.

Parameters:

`message` - the `Object` to display

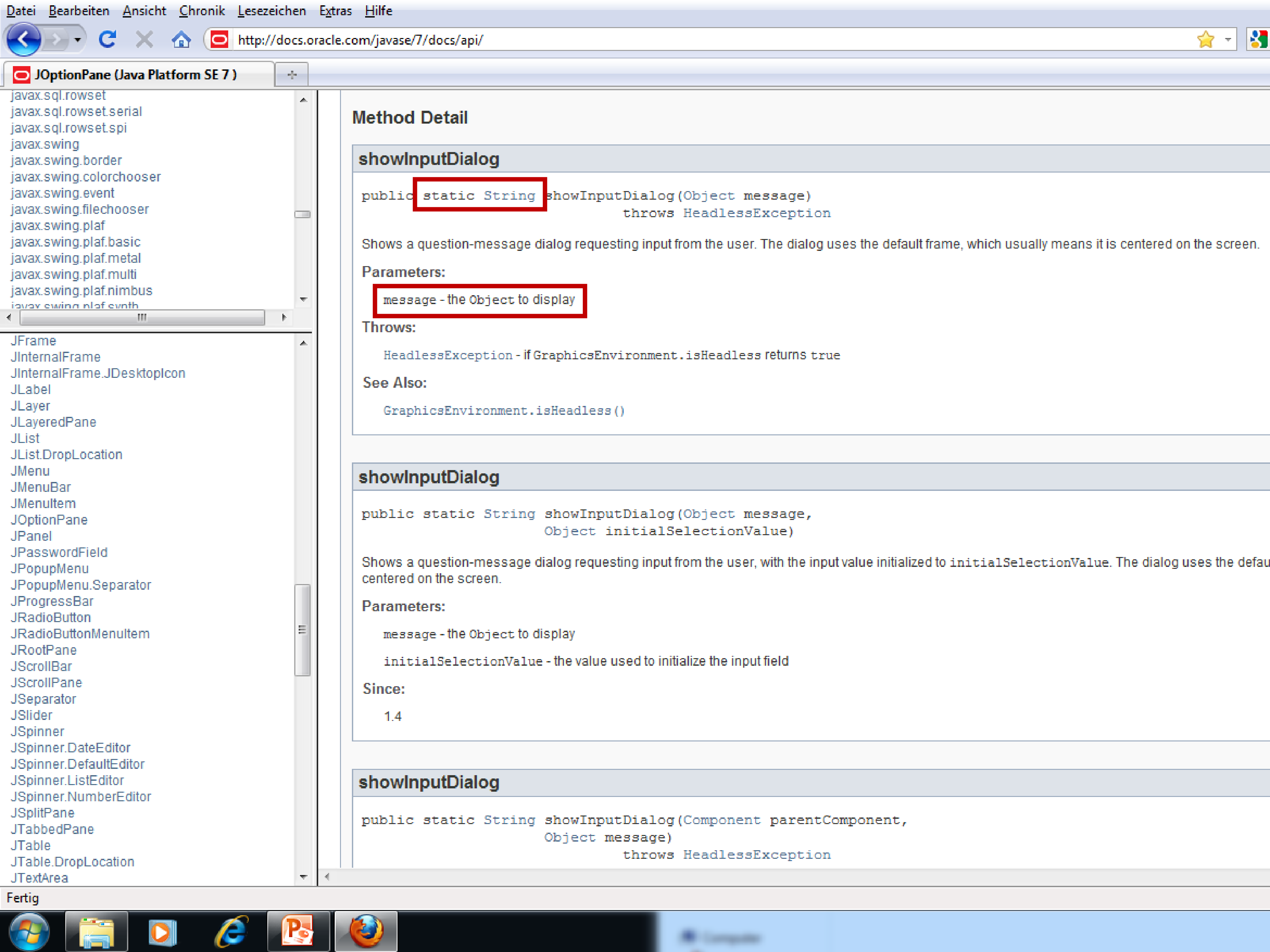
`initialSelectionValue` - the value used to initialize the input field

Since:

1.4

showInputDialog

```
public static String showInputDialog(Component parentComponent,
                                     Object message)
                              throws HeadlessException
```



Method Detail

showInputDialog

```
public static String showInputDialog(Object message)
                              throws HeadlessException
```

Shows a question-message dialog requesting input from the user. The dialog uses the default frame, which usually means it is centered on the screen.

Parameters:

`message` - the `Object` to display

Throws:

`HeadlessException` - if `GraphicsEnvironment.isHeadless` returns `true`

See Also:

`GraphicsEnvironment.isHeadless()`

showInputDialog

```
public static String showInputDialog(Object message,
                                     Object initialSelectionValue)
```

Shows a question-message dialog requesting input from the user, with the input value initialized to `initialSelectionValue`. The dialog uses the defau
centered on the screen.

Parameters:

`message` - the `Object` to display

`initialSelectionValue` - the value used to initialize the input field

Since:

1.4

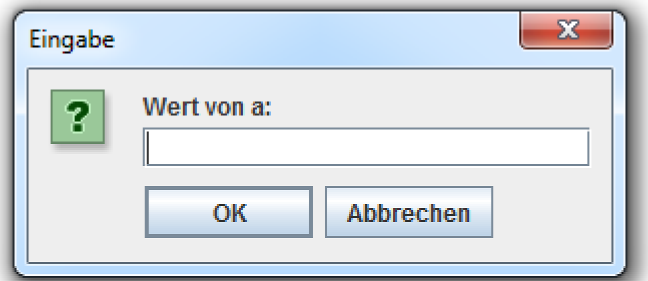
showInputDialog

```
public static String showInputDialog(Component parentComponent,
                                     Object message)
                              throws HeadlessException
```

Aufgabe 1: Modales Dialogfenster

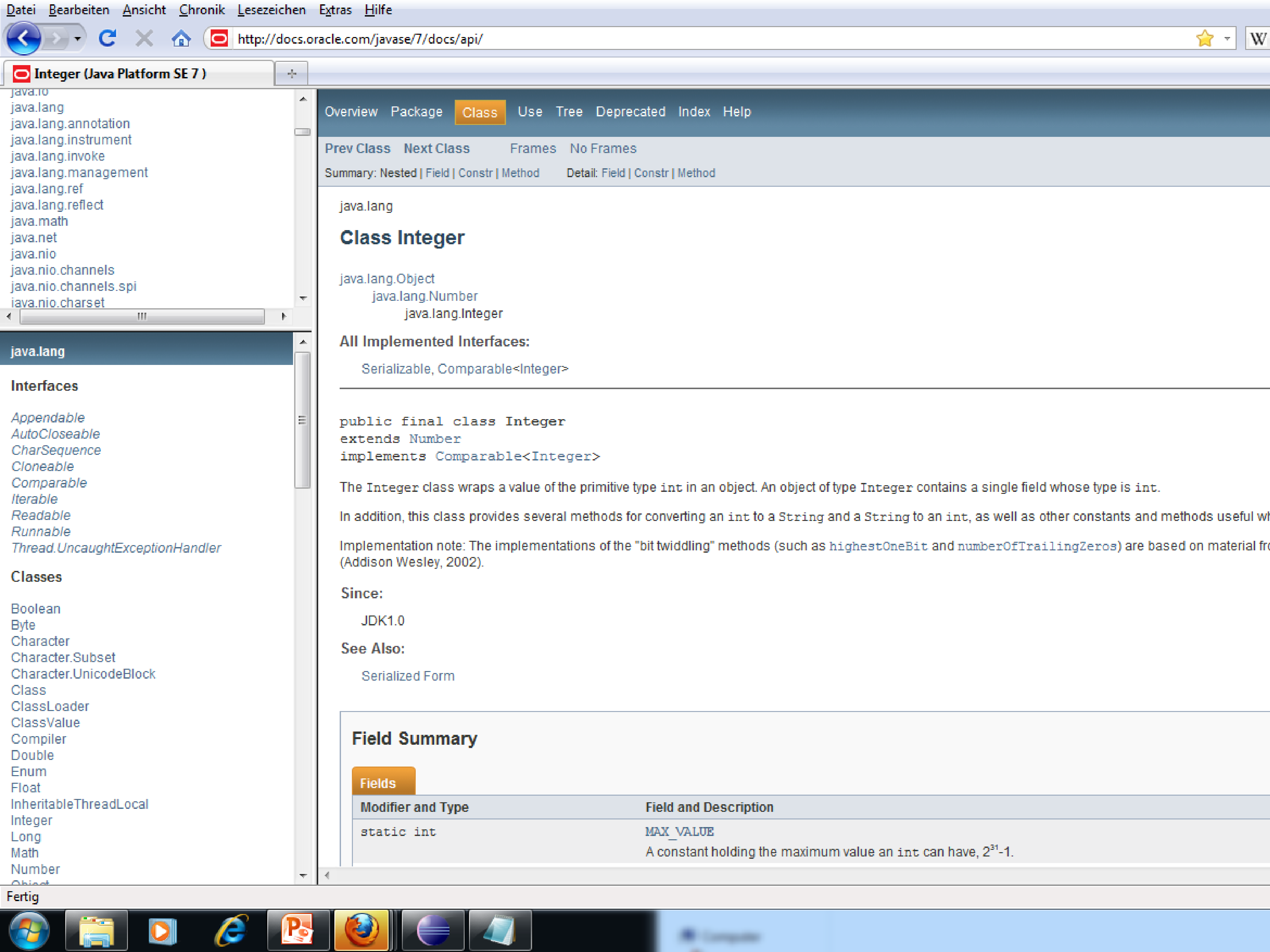
Wir brauchen für die Berechnung eines Produkts Integerwerte

```
import javax.swing.JOptionPane;

public static void main(String[] args) {
    String a = JOptionPane.showInputDialog("Wert von a: ");
    String b = JOptionPane.showInputDialog("Wert von b: ");
}
```


Eingabe

Wert von a:

Solange das Dialogfenster angezeigt wird, stoppt das Programm und wartet, bis der Nutzer einen Wert eingegeben hat.



java.lang

Class Integer

java.lang.Object
  java.lang.Number
    java.lang.Integer

All Implemented Interfaces:

Serializable, Comparable<Integer>

```
public final class Integer
extends Number
implements Comparable<Integer>
```

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`.

In addition, this class provides several methods for converting an `int` to a `String` and a `String` to an `int`, as well as other constants and methods useful wh

Implementation note: The implementations of the "bit twiddling" methods (such as `highestOneBit` and `numberOfTrailingZeros`) are based on material fr (Addison Wesley, 2002).

Since:

JDK1.0

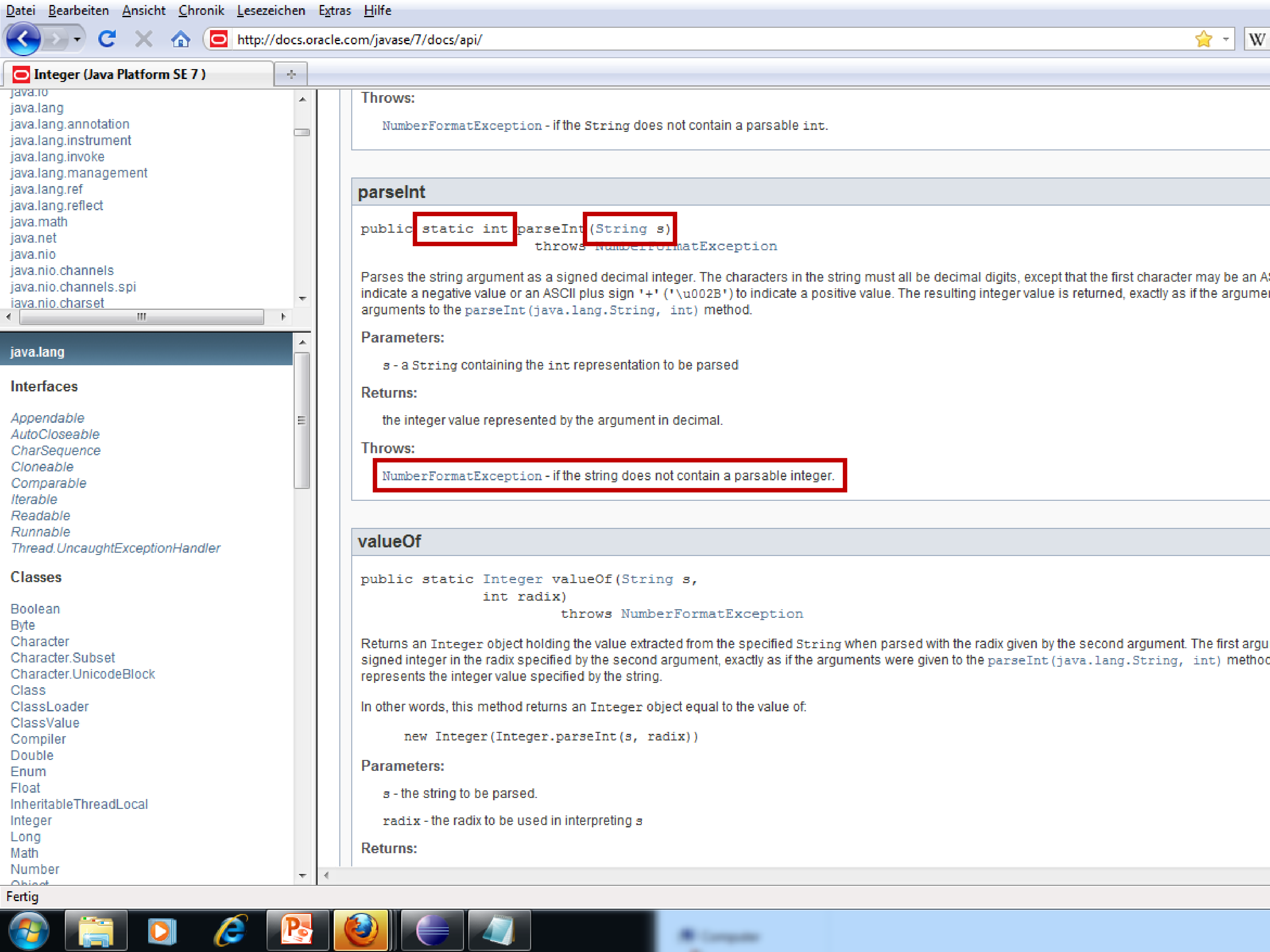
See Also:

Serialized Form

Field Summary

Fields

Modifier and Type	Field and Description
static int	MAX_VALUE A constant holding the maximum value an int can have, 2^{31}-1.



Throws:

NumberFormatException - if the String does not contain a parsable int.

parseInt

```
public static int parseInt(String s)
                throws NumberFormatException
```

Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an A
indicate a negative value or an ASCII plus sign '+' ('\u002B') to indicate a positive value. The resulting integer value is returned, exactly as if the argume
arguments to the parseInt(java.lang.String, int) method.

Parameters:

s - a String containing the int representation to be parsed

Returns:

the integer value represented by the argument in decimal.

Throws:

NumberFormatException - if the string does not contain a parsable integer.

valueOf

```
public static Integer valueOf(String s,
              int radix)
                   throws NumberFormatException
```

Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument. The first argu
signed integer in the radix specified by the second argument, exactly as if the arguments were given to the parseInt(java.lang.String, int) metho
represents the integer value specified by the string.

In other words, this method returns an Integer object equal to the value of:

```
new Integer(Integer.parseInt(s, radix))
```

Parameters:

s - the string to be parsed.

radix - the radix to be used in interpreting s

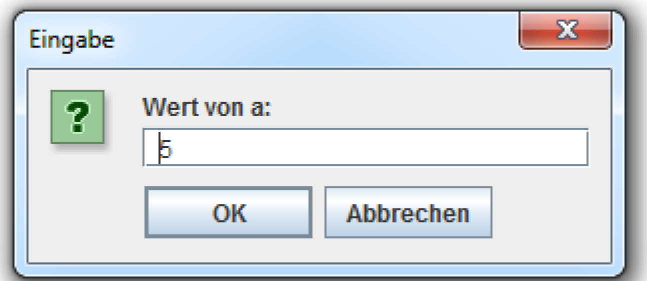
Returns:

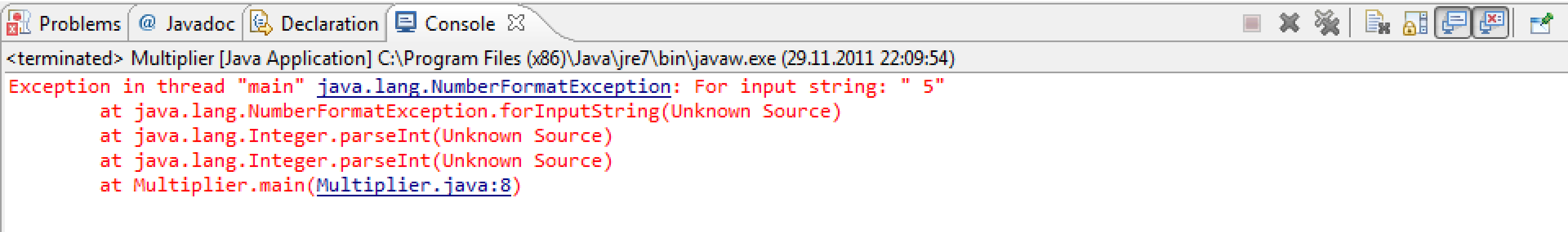
Aufgabe 1: Parsen des Eingabe-Strings

```
import javax.swing.JOptionPane;

public static void main(String[] args) {

    String s = JOptionPane.showInputDialog("Wert von a: ");
    int a = Integer.parseInt(s);

    s = JOptionPane.showInputDialog("Wert von b: ");
    int b = Integer.parseInt(s);

    int produkt = a * b;
    System.out.println("Produkt von a und b: " + produkt);
}
```

Aufgabe 1: Parsen des Eingabe-Strings

Was passiert, wenn der Nutzer in der Eile ein Leerzeichen vor der Ganzzahl eingibt? Oder eine Gleitkommazahl, oder einen Buchstaben?





```
<terminated> Multiplier [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (29.11.2011 22:09:54)
Exception in thread "main" java.lang.NumberFormatException: For input string: " 5"
	at java.lang.NumberFormatException.forInputString(Unknown Source)
	at java.lang.Integer.parseInt(Unknown Source)
	at java.lang.Integer.parseInt(Unknown Source)
	at Multiplier.main(Multiplier.java:8)
```

Aufgabe 1: Parsen des Eingabe-Strings

Weitere Methoden zum Parsen von Strings:

- **`Double.parseDouble`**`(`**`String`**` s)`
- **`Float.parseFloat`**`(`**`String`**` s)`
- **`Boolean.parseBoolean`**`(`**`String`**` s)`

Aufgabe 2

Wie kann man folgenden Ausschnitt der Realität mit Hilfe von Klassen darstellen?

In einer Anwendung sollen Schiffe und als Spezialfall davon Segelschiffe und Motorschiffe dargestellt werden können:

- Jedes **Schiff** hat eine **Tonnage** (gemessen in BRT).
- Jedes **Motorschiff** hat zusätzlich eine **Motorleistung** (Kilowatt).
- Jedes **Segelschiff** hat zusätzlich eine **Segelfläche** (m^2).
- Segelschiffe haben jedoch keine Motorleistung, Motorschiffe haben keine Segelfläche.

Aufgabe 2: Klasse Schiff

```
public class Schiff {
    private int tonnage;

    public Schiff(int tonnage) {
        this.tonnage = tonnage;
    }

    public String toString() {
        return "Schiff[tonnage=" + tonnage + "]";
    }
}
```

Aufgabe 2: Klasse SegelSchiff – fehlerhafter Konstruktor

```
public class SegelSchiff extends Schiff {
    private int segelflaeche;

    public SegelSchiff(int tonnage, int segelflaeche) {
        this.tonnage = tonnage;
        ...
    }

}
```

The field Schiff.tonnage is not visible

Aufgabe 2: Klasse SegelSchiff

```
public class SegelSchiff extends Schiff {
    private int segelflaeche;

    public SegelSchiff(int tonnage, int segelflaeche) {
        super(tonnage);
        this.segelflaeche = segelflaeche;
    }

    public String toString() { //überschriebene Version
        return "SegelSchiff[segelflaeche=" + segelflaeche +
               ", " + super.toString() + "]";
    }
}
```

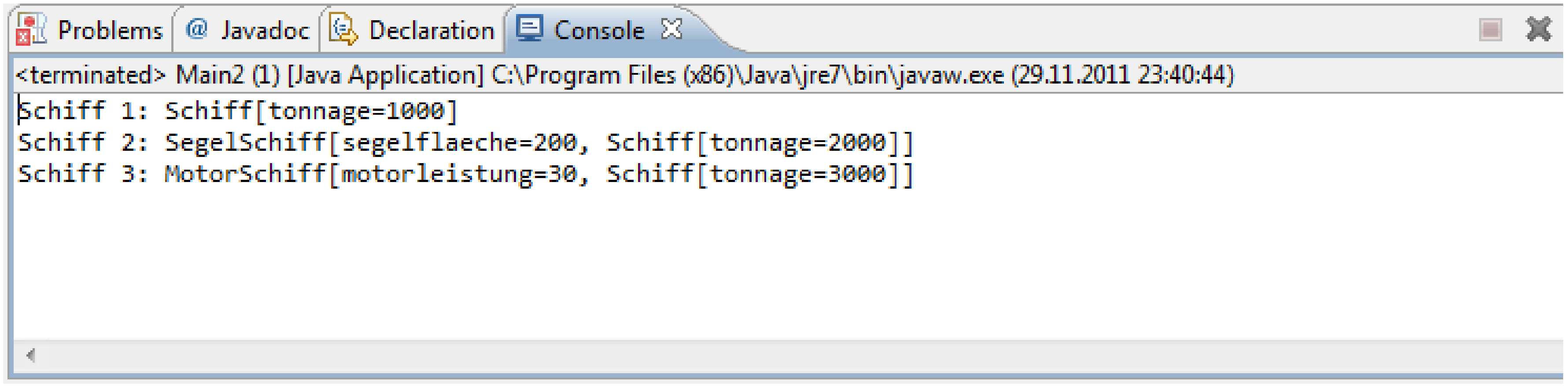
Aufgabe 2: Klasse MotorSchiff

```
public class MotorSchiff extends Schiff {
    private int motorleistung;

    public MotorSchiff(int tonnage, int motorleistung) {
        super(tonnage);
        this.motorleistung = motorleistung;
    }

    public String toString() { //überschriebene Version
        return "MotorSchiff[motorleistung=" + motorleistung +
               ", " + super.toString() + "]";
    }
}
```

Aufgabe 2: Hauptklasse

```
public class Main {
  public static void main(String[] args) {
     int schiff1tonnage = 1000;
     int schiff2tonnage = 2000;
     int schiff3tonnage = 3000;

     int schiff2segelflaeche = 200;
     int schiff3motorleistung = 30;

     Schiff schiff1 = new Schiff(schiff1tonnage);
     SegelSchiff schiff2 = new SegelSchiff(schiff2tonnage,schiff2segelflaeche);
     MotorSchiff schiff3 = new MotorSchiff(schiff3tonnage,schiff3motorleistung);

     System.out.println("Schiff 1: " + schiff1.toString());
     System.out.println("Schiff 2: " + schiff2.toString());
     System.out.println("Schiff 3: " + schiff3.toString());
  }
}
```

Aufgabe 2: Ausgabe des Programms auf der Konsole



```
<terminated> Main2 (1) [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (29.11.2011 23:40:44)
Schiff 1: Schiff[tonnage=1000]
Schiff 2: SegelSchiff[segelflaeche=200, Schiff[tonnage=2000]]
Schiff 3: MotorSchiff[motorleistung=30, Schiff[tonnage=3000]]
```