



Klassen, Vererbung, Benutzereingabe

Gilbert Beyer und Annabelle Klarl

Zentralübung zur Vorlesung Einführung in die Informatik

<http://www.pst.ifi.lmu.de/Lehre/wise-11-12/infoeinf>



Inhalte der letzten Vorlesungen:

- Objektorientierte Programmierung
- Klassendeklarationen in Java
- Objektzustand, Gleichheit von Objekten
- Objekterzeugung und -benutzung
- Methodenaufruf, Parameterübergabe
- Klassenmethoden und -attribute
- Vererbung in Java



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?



Beispiel: Multiplikation

```
public static void main(String[] args) {  
  
    int a = 2;  
    int b = 5;  
  
    System.out.println("Wert von a: " + a);  
    System.out.println("Wert von b: " + b);  
  
    int product = a * b;  
    System.out.println("Produkt von a und b: " + product);  
  
}
```

- 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 with the browser.
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 provides a mechanism for exchanging 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-dimensional geometry.
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 runtime.
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 <i>beans</i> – 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.

- 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

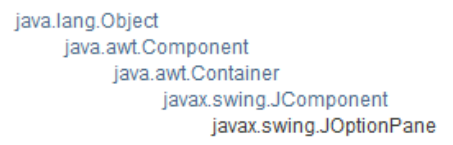
Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

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

javax.swing

Class JOptionPane



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 JOptionPane 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 static methods 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 convenient 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
```

- JOptionPane (Java Platform SE 7)
- 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 initialValue)
```

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

Parameters:

message - the Object to display

initialValue - the value used to initialize the input field

Since:

1.4

showInputDialog

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

- 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 initialValue)
```

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

Parameters:

message - the Object to display

initialValue - the value used to initialize the input field

Since:

1.4

showInputDialog

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

Methodenname
showInputDialog
ist überladen
(Overloading)

- JOptionPane (Java Platform SE 7)
- 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 initialValue)
```

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

Parameters:

message - the Object to display

initialValue - the value used to initialize the input field

Since:

1.4

showInputDialog

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



Klasse JOptionPane verwenden

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



Klasse JOptionPane verwenden

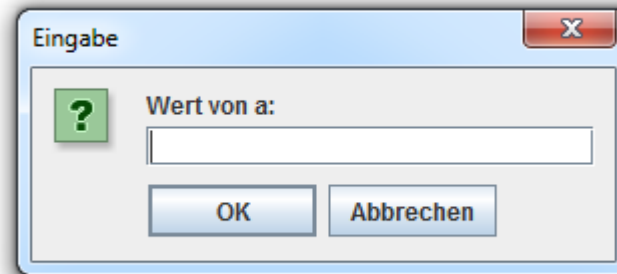
```
import javax.swing.*;  
  
public static void main(String[] args) {  
  
    String a = JOptionPane.showInputDialog("Wert von a: ");  
    String b = JOptionPane.showInputDialog("Wert von b: ");  
  
}
```



Modales Dialogfenster

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

Solange kann auch in möglichen anderen vorhandenen Fenstern der Anwendung nicht weitergearbeitet werden.





Multiplikation

```
import javax.swing.JOptionPane;  
  
public static void main(String[] args) {  
  
    String a = JOptionPane.showInputDialog("Wert von a: ");  
    String b = JOptionPane.showInputDialog("Wert von b: ");  
    // brauchen aber Integerwerte für die Berechnung  
    // eines Produkts, keine Strings  
  
}
```

Integer (Java Platform SE 7)

- java.io
- java.lang
- java.lang.annotation
- java.lang.instrument
- java.lang.invoke
- java.lang.management
- java.lang.ref
- java.lang.reflect
- java.math
- java.net
- java.nio
- java.nio.channels
- java.nio.channels.spi
- java.nio.charset

java.lang

Interfaces

- Appendable
- AutoCloseable
- CharSequence
- Cloneable
- Comparable
- Iterable
- Readable
- Runnable
- Thread.UncaughtExceptionHandler

Classes

- Boolean
- Byte
- Character
- Character.Subset
- Character.UnicodeBlock
- Class
- ClassLoader
- ClassValue
- Compiler
- Double
- Enum
- Float
- InheritableThreadLocal
- Integer
- Long
- Math
- Number
- Object

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 with `int` values.

Implementation note: The implementations of the "bit twiddling" methods (such as `highestOneBit` and `numberOfTrailingZeros`) are based on material from *Computer Systems: A Programmer's Perspective* (Addison Wesley, 2002).

Since:

JDK1.0

See Also:

Serialized Form

Field Summary

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

Integer (Java Platform SE 7)

- java.io
- java.lang
- java.lang.annotation
- java.lang.instrument
- java.lang.invoke
- java.lang.management
- java.lang.ref
- java.lang.reflect
- java.math
- java.net
- java.nio
- java.nio.channels
- java.nio.channels.spi
- java.nio.charset

java.lang

Interfaces

- Appendable
- AutoCloseable
- CharSequence
- Cloneable
- Comparable
- Iterable
- Readable
- Runnable
- Thread.UncaughtExceptionHandler

Classes

- Boolean
- Byte
- Character
- Character.Subset
- Character.UnicodeBlock
- Class
- ClassLoader
- ClassValue
- Compiler
- Double
- Enum
- Float
- InheritableThreadLocal
- Integer
- Long
- Math
- Number
- Object

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 ASCII minus sign '-' ('\u002D') to 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 argument were given 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 argument is the String whose value is to be extracted, and the second argument is the radix to use in interpreting the string. The first argument must be a signed integer in the radix specified by the second argument, exactly as if the arguments were given to the `parseInt(java.lang.String, int)` method. The second argument must be equal to one of the following: 2, 8, 10, 16, 32, 64, 128, 256, or a power of two greater than 256. The first argument 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:

Integer (Java Platform SE 7)

- java.io
- java.lang
- java.lang.annotation
- java.lang.instrument
- java.lang.invoke
- java.lang.management
- java.lang.ref
- java.lang.reflect
- java.math
- java.net
- java.nio
- java.nio.channels
- java.nio.channels.spi
- java.nio.charset

java.lang

Interfaces

- Appendable
- AutoCloseable
- CharSequence
- Cloneable
- Comparable
- Iterable
- Readable
- Runnable
- Thread.UncaughtExceptionHandler

Classes

- Boolean
- Byte
- Character
- Character.Subset
- Character.UnicodeBlock
- Class
- ClassLoader
- ClassValue
- Compiler
- Double
- Enum
- Float
- InheritableThreadLocal
- Integer
- Long
- Math
- Number
- Object

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 ASCII minus sign '-' ('\u002D') to 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 argument were given 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 argument is the String representing the integer to be parsed. The second argument is the radix to be used in interpreting the string. The first argument must be a signed integer in the radix specified by the second argument, exactly as if the arguments were given to the `parseInt(java.lang.String, int)` method. The second argument must be either 10, 8, 16, or 2, representing 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:



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);

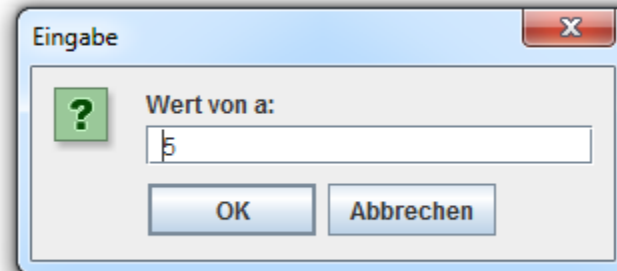
    s = JOptionPane.showInputDialog("Wert von b: ");
    int b = Integer.parseInt(s);

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



Parsen des Eingabe-Strings

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



```
Problems @ Javadoc Declaration Console X
<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)
```



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 als objektorientiertes Programm darstellen: In einer Anwendung sollen Schiffe und als Spezialfall davon Segelschiffe und Motorschiffe dargestellt werden können.



Aufgabe 2)

Für die verschiedenen Schiffstypen soll gelten:

Jedes Schiff hat eine Tonnage (gemessen in BRT).

Jedes Motorschiff hat zusätzlich eine Motorleistung (Kilowatt).

Jedes Segelschiff hat zusätzlich zur Tonnage eine Segelfläche (qm).

Segelschiffe haben jedoch keine Motorleistung, Motorschiffe

Haben keine Segelfläche, und andere Schiffe haben weder

das eine noch das andere.



Aufgabe 2)

Für die Anwendung soll außerdem gelten:

Informationen über die Eigenschaften der Schiffe sollen auf der Konsole ausgegeben werden können.



Klasse Schiff

```
public class Schiff {  
    private int tonnage;  
  
    public Schiff(int tonnage) {  
        this.tonnage = tonnage;  
    }  
  
    public String toString() {  
        return "Schiff[tonnage=" + tonnage + "];"  
    }  
}
```



Klasse SegelSchiff – Entwurf

```
public class SegelSchiff extends Schiff {
```

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

```
        ...
```

```
    }
```


```
}
```



The field Schiff.tonnage is not visible



Klasse SegelSchiff – Entwurf

```
public class SegelSchiff extends Schiff {  
    private int tonnage;  
    //falscher Ansatz, jetzt zwei Attribute tonnage vorhanden  
    //Schiff.tonnage und Segelschiff.tonnage  
  
    public SegelSchiff(int tonnage, int segelflaeche) {  
        this.tonnage = tonnage;  
  
         Implicit super constructor Schiff() is undefined. Must explicitly invoke another constructor  
  
    }  
  
}
```



Klasse SegelSchiff

```
public class SegelSchiff extends Schiff {  
    private int segelflaeche;  
  
    public SegelSchiff(int tonnage, int segelflaeche) {  
        super(tonnage);  
        this.segelflaeche = segelflaeche;  
    }  
  
    ...  
}
```



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() + "];"  
    }  
}
```



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() + "];  
    }  
}
```



Hauptklasse

```
public class Main {  
  
    public static void main(String[] args) {  
        final int SCHIFF1_TONNAGE = 1000;  
        final int SCHIFF2_TONNAGE = 2000;  
        final int SCHIFF3_TONNAGE = 3000;  
  
        final int SCHIFF2_SEGELFLAECHE = 200;  
        final int SCHIFF3_MOTORLEISTUNG = 30;  
  
        Schiff schiff1 = new Schiff(SCHIFF1_TONNAGE);  
        SegelSchiff schiff2 = new SegelSchiff(SCHIFF2_TONNAGE, SCHIFF2_SEGELFLAECHE);  
        MotorSchiff schiff3 = new MotorSchiff(SCHIFF3_TONNAGE, SCHIFF3_MOTORLEISTUNG);  
  
        System.out.println("Schiff 1: " + schiff1.toString());  
        System.out.println("Schiff 2: " + schiff2.toString());  
        System.out.println("Schiff 3: " + schiff3.toString());  
  
    }  
}
```



Ausgabe des Programms auf der Konsole

A screenshot of a Java IDE's console window. The window title bar shows 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output is as follows:

```
<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]]
```



Benutzereingabe zur Laufzeit

```
import javax.swing.JOptionPane;

public class Main {

    public static void main(String[] args) {
        //hier Konstantendeklarationen

        Schiff schiff1 = new Schiff(SCHIFF1_TONNAGE);
        SegelSchiff schiff2 = new SegelSchiff(SCHIFF2_TONNAGE, SCHIFF2_SEGELFLAECHE);
        MotorSchiff schiff3 = new MotorSchiff(SCHIFF3_TONNAGE, SCHIFF3_MOTORLEISTUNG);

        Schiff charterausflug;
        String buchung =
            JOptionPane.showInputDialog("Welches Schiff[Schiff, Motorschiff, Segelschiff]: ");
        if(buchung.equals("Schiff")) charterausflug = schiff1;
        else if (buchung.equals("Segelschiff")) charterausflug = schiff2;
        else charterausflug = schiff3; //Motorschiff

        System.out.println("Charterausflug mit: " + charterausflug.toString());
    }
}
```



Entscheidung z.B. für ein Segelschiff

The image shows a screenshot of a Windows-style dialog box titled "Eingabe". On the left side of the dialog, there is a green square icon containing a white question mark. To the right of this icon, the text "Welches Schiff[Schiff, Motorschiff, Segelschiff]:" is displayed. Below this text is a text input field containing the word "Segelschiff". At the bottom of the dialog, there are two buttons: "OK" and "Abbrechen".



Ausgabe des Programms auf der Konsole

A screenshot of a Java IDE's console window. The window title bar shows tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text reads: '<terminated> Main2 (1) [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (30.11.2011 00:21:06)' followed by the output 'Charterausflug mit: SegelSchiff[segelflaeche=200, Schiff[tonnage=2000]]'.

```
<terminated> Main2 (1) [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (30.11.2011 00:21:06)
Charterausflug mit: SegelSchiff[segelflaeche=200, Schiff[tonnage=2000]]
```



Dynamische Bindung

Die Variable `charterausflug` hat den Typ `Schiff`.

Zur Compile-Zeit ist jedoch nicht bekannt, welches Schiff über die Benutzereingabe ausgewählt werden wird.

Welche Methode schliesslich mit `charterausflug.toString()` bezeichnet wird, hängt in Java nicht vom Typ der Variablen ab, sondern vom Typ des Objekts, auf welches die Objektreferenz zur Laufzeit zeigt. Im Beispielprogramm kann also je nach Entscheidung des Nutzers zur Laufzeit die Methode `toString()` der Klasse `Schiff`, `Segelschiff` oder `Motorschiff` gemeint sein.