

Java API

A compilation of selected constructors and methods in some common Java classes.

CONTENTS	Page
Miscellaneous	1
Strings	3
Data structures	5
Streams and files	9
Observer	14
Threads and timers	14
GUI	16

Miscellaneous

class java.lang.Object:

protected Object clone()
Creates and returns a copy of this object.

boolean equals(Object obj)
Indicates whether some other object is "equal to" this one.

Protected Class getClass()
Returns the runtime class of this Object.

int hashCode()
Returns a hash code value for the object.

void notify()
Wakes up a single thread that is waiting on this object's monitor.

void notifyAll()
Wakes up all threads that are waiting on this object's monitor.

String toString()
Returns a string representation of the object.

void wait()
Causes the current thread to wait until another thread invokes the **notify()** method or the **notifyAll()** method for this object.

void wait(long timeout)
Causes the current thread to wait until either another thread invokes the **notify()** method or the **notifyAll()** method for this object, or a specified amount of time has elapsed.

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

interface java.lang.Comparable<T>

int compareTo(T obj)
Compares this object with the specified object for order.

interface java.util.Comparator<T>

int compare(T obj1, T obj2)
Compares its two arguments for order.

class java.lang.Integer: analogous for long, float, double, char, boolean, etc.

static int parseInt(String s) throws NumberFormatException
 Parses the string argument as a signed decimal integer.

class java.util.Scanner:

Scanner(File source)
Constructs a new Scanner that produces values scanned from the specified file.

Scanner(InputStream source)

Constructs a new Scanner that produces values scanned from the specified input stream.

Scanner(String source)

Constructs a new Scanner that produces values scanned from the specified string.

boolean hasNext()

Returns true if this scanner has another token in its input.

boolean hasNext(String pattern)

Returns true if the next token matches the pattern constructed from the specified string.

boolean hasNextInt() analogous for long, float, double, char, boolean, etc.

Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the **nextInt()** method.

boolean hasNextLine()

Returns true if there is another line in the input of this scanner.

String next()

Finds and returns the next complete token from this scanner.

String nextLine()

Advances this scanner past the current line and returns the input that was skipped.

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

Strings

class java.lang.String:

String()
Initializes a newly created String object so that it represents an empty character sequence.

String(char[] value)
Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

String(byte[] bytes, int offset, int length)
Constructs a new String by decoding the specified subarray of bytes starting at offset.

String(String original)
Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

char charAt(int index)
Returns the character at the specified index.

int compareTo(String anotherString)
Compares two strings lexicographically.

int compareToIgnoreCase(String str)
Compares two strings lexicographically, ignoring case differences.

String concat(String str)
Concatenates the specified string to the end of this string.

boolean contentEquals(StringBuffer sb)
Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.

static String copyValueOf(char[] data)
Returns a String that represents the character sequence in the array specified.

boolean endsWith(String suffix)
Tests if this string ends with the specified suffix.

boolean equals(Object anObject)
Compares this string to the specified object.

boolean equalsIgnoreCase(String anotherString)
Compares this String to another String, ignoring case considerations.

byte[] getBytes()
Encodes this String into a sequence of bytes, storing the result into a new byte array.

int hashCode()
Returns a hash code for this string.

int indexOf(int ch)
Returns the index within this string of the first occurrence of the specified character.

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

int indexOf(int ch, int fromIndex)
Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

int indexOf(String str)
Returns the index within this string of the first occurrence of the specified substring.

int indexOf(String str, int fromIndex)
Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

int lastIndexOf(int ch)
Returns the index within this string of the last occurrence of the specified character.

int lastIndexOf(int ch, int fromIndex)
Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

int lastIndexOf(String str)
Returns the index within this string of the rightmost occurrence of the specified substring.

int lastIndexOf(String str, int fromIndex)
Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.

int length()
Returns the length of this string.

boolean matches(String regex)
Tells whether or not this string matches the given regular expression.

String replace(char oldChar, char newChar)
Returns a new string resulting from replacing all occurrences of **oldChar** in this string with **newChar**.

String replaceAll(String regex, String replacement)
Replaces each substring of this string that matches the given regular expression with the given replacement.

String replaceFirst(String regex, String replacement)
Replaces the first substring of this string that matches the given regular expression with the given replacement.

String[] split(String regex)
Splits this string around matches of the given regular expression.

boolean startsWith(String prefix)
Tests if this string starts with the specified prefix.

String substring(int beginIndex, int endIndex)
Returns a new string that is a substring of this string.

char[] toCharArray()
Converts this string to a new character array.

String toLowerCase()
Converts all of the characters in this String to lower case.

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```

String toUpperCase()
    Converts all of the characters in this String to upper case.

String trim()
    Returns a copy of the string, with leading and trailing whitespace omitted.

Data Structures
class java.util.ArrayList<E>

ArrayList()
    Constructs an empty list.

void add(int index, E element)
    Inserts the specified element at the specified position in this list.

boolean add(E o)
    Appends the specified element to the end of this list.

void clear()
    Removes all of the elements from this list.

Object clone()
    Returns a shallow copy of this list.

boolean contains(E elem)
    Returns true if this list contains the specified element.

boolean equals(Object o)
    Returns true if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are equal.

E get(int index)
    Returns the element at the specified position in this list.

int indexOf(E elem)
    Searches for the first occurrence of the given argument, testing for equality using the equals method.

boolean isEmpty()
    Tests if this list has no elements.

Iterator iterator()
    Returns an iterator over the elements in this list in proper sequence.

Object remove(int index)
    Removes the element at the specified position in this list.

Object set(int index, E element)
    Replaces the element at the specified position in this list with the specified element.

int size()
    Returns the number of elements in this list.

E[] toArray()
    Returns an array containing all of the elements in this list in the correct order.

```

```

E[] toArray(E[] a)
    Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

class java.util.LinkedList<E>

LinkedList()
    Constructs an empty list.

boolean addFirst(E o)
    Inserts the specified element at the beginning of this list.

boolean addLast(E o)
    Appends the specified element to the end of this list.

E getFirst()
    Returns the first element in this list.

E getLast()
    Returns the last element in this list.

E removeFirst()
    Removes and returns the first element from this list.

E removeLast()
    Removes and returns the last element from this list.

void add(int index, E element)
    Inserts the specified element at the specified position in this list.

boolean add(E o)
    Appends the specified element to the end of this list.

void clear()
    Removes all of the elements from this list.

Object clone()
    Returns a shallow copy of this list.

boolean contains(E elem)
    Returns true if this list contains the specified element.

boolean equals(Object o)
    Returns true if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are equal.

E get(int index)
    Returns the element at the specified position in this list.

int indexOf(E elem)
    Searches for the first occurrence of the given argument, testing for equality using the equals method.

boolean isEmpty()
    Tests if this list has no elements.

Iterator iterator()
    Returns an iterator over the elements in this list in proper sequence.

```

```

Object remove(int index)
    Removes the element at the specified position in this list.

Object set(int index, E element)
    Replaces the element at the specified position in this list with the specified element.

int size()
    Returns the number of elements in this list.

E[] toArray()
    Returns an array containing all of the elements in this list in the correct order.

E[] toArray(E[] a)
    Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

interface java.util.Iterator<E>

boolean hasNext()
    Returns true if the iteration has more elements. (In other words, returns true if next would return an element rather than throwing an exception.)

E next()
    Returns the next element in the iteration.

Throws: NoSuchElementException - iteration has no more elements.

void remove()
    Removes from the underlying collection the last element returned by the iterator. This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

```

interface java.lang.Iterable<E>:

```

Iterator<E> iterator()
    Returns an iterator over a set of elements of type T.

```

class java.util.HashSet<E>, class java.util.TreeSet<E>:

```

HashSet()
    Constructs a new, empty HashSet.

```

```

TreeSet()
    Constructs a new, empty TreeSet sorted according to the ordering defined by the compareTo method in class E.

```

```

TreeSet(Comparator<E> comparator)
    Constructs a new, empty tree set, sorted according to the specified comparator.

```

Methods (both classes)

```

boolean add(E o)
    Adds the specified element to this set if it is not already present.

void clear()
    Removes all of the elements from this set.

boolean contains(E o)
    Returns true if this set contains the specified element.

boolean isEmpty()
    Returns true if this set contains no elements.

Iterator<E> iterator()
    Returns an iterator over the elements in this set.

boolean remove(E o)
    Removes the specified element from this set if it is present.

int size()
    Returns the number of elements in this set (its cardinality).

class java.util.HashMap<K,V>, class java.util.TreeMap<K,V>:


```

HashMap()
 Constructs an empty HashMap.

TreeMap()
 Constructs an empty TreeMap.

Methods (both classes)
void clear()
 Removes all mappings from this map.

boolean containsKey(Object key)
 Returns true if this map contains a mapping for the specified key.

boolean containsValue(Object value)
 Returns true if this map maps one or more keys to the specified value.

Set<Map.Entry<K,V>> entrySet()
 Returns a collection view of the mappings contained in this map.

V get(Object key)
 Returns the value to which the specified key is mapped in this hash map, or null if the map contains no mapping for this key.

boolean isEmpty()
 Returns true if this map contains no key-value mappings.

Set<K> keySet()
 Returns a set view of the keys contained in this map.

V put(K key, V value)
 Associates the specified value with the specified key in this map.

V remove(Object key)
 Removes the mapping for this key from this map if present.

```


```

```
int size()
    Returns the number of key-value mappings in this map.

Collection<V> values()
    Returns a collection view of the values contained in this map.

interface java.util.Map.Entry<K,V>:

K getKey()
    Returns the key corresponding to this entry.

V getValue()
    Returns the value corresponding to this entry.
```

Streams and Files

General

```
void close()
    Closes the stream.

class java.io.FileReader:

FileReader(String fileName)
    Creates a new FileReader, given the name of the file to read from.

int read() throws IOException
    Reads a single character.

int read(char[] cbuf, int off, int len) throws IOException
    Reads characters into a portion of an array

long skip(long n) throws IOException
    Skips n characters.

class java.io.BufferedReader:

BufferedReader(Reader in)
    Creates a buffering character-input stream that uses a input buffer.

int read() throws IOException
    Reads a single character.

int read(char[] cbuf, int off, int len) throws IOException
    Reads characters into a portion of an array.

String readLine() throws IOException
    Reads a line of text.
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
long skip(long n) throws IOException
    Skips characters.

class java.io.FileWriter:

FileWriter(String fileName) throws IOException
    Constructs a FileWriter object given a file name.

FileWriter(String fileName, boolean append) throws IOException
    Constructs a FileWriter object given a file name, with a boolean indicating whether or not to append the data written to the end of the file rather than the beginning.

void write(char[] cbuf, int off, int len)
    Writes a portion of an array of characters.

void write(int c)
    Writes a single character.

void write(String str, int off, int len)
    Writes a portion of a string.

void flush()
    Flushes the stream.

class java.io.BufferedWriter:

BufferedWriter(Writer out)
    Creates a buffered character-output stream that uses a default-sized output buffer.

void flush()
    Flushes the stream.

void newLine()
    Writes a line separator.

void write(char[] cbuf, int off, int len)
    Writes a portion of an array of characters.

void write(int c)
    Writes a single character.

void write(String s, int off, int len)
    Writes a portion of a String.

class java.io.FileInputStream:

FileInputStream(String name)
    Creates a FileInputStream by opening a connection to an actual file, the file named by the path name name in the file system.
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
int read()
    Reads a byte of data from this input stream.

int read(byte[] b, int off, int len)
    Reads up to len bytes of data from this input stream into an array of bytes.

long skip(long n)
    Skips over and discards n bytes of data from the input stream.

class java.io.DataInputStream:

DataInputStream(InputStream in)
    Creates a DataInputStream that uses the specified underlying InputStream.

int read(byte[] b)
    Reads some number of bytes from the contained input stream and stores them into the buffer array b.

int read(byte[] b, int off, int len)
    Reads up to len bytes of data from the contained input stream into an array of bytes.

boolean readBoolean()
    Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.

byte readByte()
    Reads and returns one input byte.

char readChar()
    Reads two input bytes and returns a char value.

double readDouble()
    Reads eight input bytes and returns a double value.

float readFloat()
    Reads four input bytes and returns a float value.

int readInt()
    Reads four input bytes and returns an int value.

String readLine()
    Reads the next line of text from the input stream.

long readLong()
    Reads eight input bytes and returns a long value.

short readShort()
    Reads two input bytes and returns a short value.

int skipBytes(int n)
    Makes an attempt to skip over n bytes of data from the input stream,
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
discarding the skipped bytes.

class java.io.ObjectInputStream:

ObjectInputStream(InputStream in)
    Creates an ObjectInputStream that reads from the specified InputStream.

Object readObject()
    Read an object from the ObjectInputStream.

class java.io.BufferedInputStream:

BufferedInputStream(InputStream in)
    Creates a BufferedInputStream that reads from the specified InputStream.

int read()
    See the general contract of the read method of InputStream.

int read(byte[] b, int off, int len)
    Reads bytes from this byte-input stream into the specified byte array, starting at the given offset.

void reset()
    See the general contract of the reset method of InputStream.

long skip(long n)
    See the general contract of the skip method of InputStream.

class java.io.InputStreamReader:

InputStreamReader(InputStream in)
    Creates an InputStreamReader that uses the default charset.

int read()
    Reads a single character.

int read(char[] cbuf, int offset, int length)
    Reads characters into a portion of an array.

class java.io.FileOutputStream:

FileOutputStream(File file)
    Creates a file output stream to write to the file represented by the specified File object.

void write(byte[] b, int off, int len)
    Writes len bytes from the specified byte array starting at offset off to this file output stream.

void write(int b)
    Writes the specified byte to this file output stream.
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
class java.io.DataOutputStream:
DataOutputStream(OutputStream out)
    Creates a new data output stream to write data to the specified underlying output stream.

void write(byte[] b, int off, int len)
    Writes len bytes from the specified byte array starting at offset off
    to the underlying output stream.

void write(int b)
    Writes the specified byte (the low eight bits of the argument b)
    to the underlying output stream.

void writeBoolean(boolean v)
    Writes a boolean to the underlying output stream as a 1-byte value.

void writeByte(int v)
    Writes a byte to the underlying output stream as a 1-byte value.

void writeBytes(String s)
    Writes out the string to the underlying output stream as a sequence of bytes.

void writeChar(int v)
    Writes a char to the underlying output stream as a 2-byte value, high byte first.

void writeChars(String s)
    Writes a string to the underlying output stream as a sequence of characters.

void writeDouble(double v)
    Converts the double argument to a long using the doubleToLongBits method in
    class Double, and then writes that long value to the underlying output stream
    as an 8-byte quantity, high byte first.

void writeFloat(float v)
    Converts the float argument to an int using the floatToIntBits method in class
    Float, and then writes that int value to the underlying output stream as a 4-byte
    quantity, high byte first.

void writeInt(int v)
    Writes an int to the underlying output stream as four bytes, high byte first.

void writeLong(long v)
    Writes a long to the underlying output stream as eight bytes, high byte first.

void writeShort(int v)
    Writes a short to the underlying output stream as two bytes, high byte first.

class java.io.ObjectOutputStream:
ObjectOutputStream(OutputStream out)
    Creates an ObjectOutputStream that writes to the specified OutputStream.
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
void writeObject(Object obj)
    Write the specified object to the ObjectOutputStream.

class java.io.BufferedOutputStream:
BufferedOutputStream(OutputStream out)
    Creates a new buffered output stream to write data to the specified underlying
    output stream.

void flush()
    Flushes this buffered output stream.

void write(byte[] b, int off, int len)
    Writes len bytes from the specified byte array starting at offset off to this buffered
    output stream.

void write(int b)
    Writes the specified byte to this buffered output stream.

Observer
class java.util.Observable:
void addObserver(Observer o)
    Adds an observer to the set of observers for this object.

protected void setChanged()
    Marks this Observable object as having been changed.

void notifyObservers()
    If this object has changed, then notify all of its observers.

interface java.util.Observer:
void update(Observable o, Object arg)
    This method is called whenever the observed object is changed.
```

Threads and timers

interface java.lang.Runnable:

```
void run()
    When an object implementing interface Runnable is used to create a thread,
    starting the thread causes the object's run method to be called in that separately
    executing thread.
```

class java.lang.Thread:

```
Thread()
    Allocates a new Thread object.
```

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
Thread(Runnable object)
    Allocates a new Thread object.

void start()
    Causes this thread to begin execution; the Java Virtual Machine calls the run method
    of this thread.

static void sleep(long millis) throws InterruptedException
    Causes the currently executing thread to sleep for the specified number of milliseconds.

static boolean interrupted()
    Tests whether the current thread has been interrupted.

void join()
    Waits for this thread to die.

class javax.swing.Timer:
Timer(int delay, ActionListener listener)
    Creates a Timer and initializes both the initial delay and between-event delay to delay
    milliseconds.

void start()
    Starts the Timer, causing it to start sending action events to its listeners.

void stop()
    Stops the Timer, causing it to stop sending action events to its listeners.

class java.util.Timer:
Timer()
    Creates a new timer.

void schedule(TimerTask task, long delay, long period)
    Schedules the specified task for repeated fixed-delay execution,
    beginning at the specified delay.
```

void cancel()
 Terminates this timer, discarding any currently scheduled tasks.

class java.util.TimerTask:

protected TimerTask()
 Creates a new timer task.

abstract void run()
 The action to be performed by this timer task.

boolean cancel()
 Cancels this timer task.

Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012

```
GUI
class javax.swing.JFrame:
JFrame()
    Constructs a new frame that is initially invisible.

Container getContentPane()
    Returns the contentPane object for this frame.

void setTitle(String title)
    Sets the title for this frame to the specified string.

void setJMenuBar(JMenuBar menubar)
    Sets the menubar for this frame.

void setLayout(LayoutManager manager)
    Sets the LayoutManager.

class javax.swing.JMenuBar:
JMenuBar()
    Creates a new menu bar.

JMenu add(JMenu c)
    Appends the specified menu to the end of the menu bar.

class javax.swing.JMenu:
JMenu(String s)
    Constructs a new JMenu with the supplied string as its text.

JMenuItem add(JMenuItem menuItem)
    Appends a menu item to the end of this menu.

class javax.swing.JMenuItem:
JMenuItem(String text)
    Creates a JMenuItem with the specified text.

void addActionListener(ActionListener l)
    Adds an ActionListener to this menu item.

class javax.swing.JPanel:
JPanel(LayoutManager layout)
    Create a new JPanel with the specified layout manager.

Component add(Component comp)
    Appends the specified component to this container.

class javax.swing.JLabel:
Version: 2011-10-18 Objektorienterad programmering DAT042 2011/2012
```

```

JLabel(String text)
    Creates a JLabel instance with the specified text.

void setText(String text)
    Defines the single line of text this component will display.

String getText()
    Returns the label's text.

class javax.swing.JButton:
JButton(String text)
    Creates a JButton instance with the specified text.

void setText(String text)
    Defines the single line of text this component will display.

String getText()
    Returns the button's text.

void setIcon(Icon icon)
    Sets the button's default icon.

void setEnabled(boolean b)
    Enables (or disables) the button.

addActionListener(ActionListener l)
    Adds the specified action listener to receive action events from this button.

class javax.swing.JTextField:
String getText()
    Returns the text contained in this textfield.

void setText(String t)
    Sets the text of this textfield to the specified text.

void setEditable(boolean b)
    Sets whether or not this TextComponent should be editable.

void addMouseListener(ActionListener l)
    Adds the specified action listener to receive action events from this textfield.

void addKeyListener(KeyListener l)
    Adds the specified key listener to receive key events from this textfield.

```

```

class javax.swing.JTextArea:
JTextArea(int rows, int columns)
    Constructs a new empty TextArea with the specified number of rows and columns.

void append(String str)
    Appends the given text to the end of the document.

String getText()
    Returns the text contained in this textfield.

void setText(String t)
    Sets the text of this textarea to the specified text. Empties the area if t is null.

void setEditable(boolean b)
    Sets whether or not this TextComponent should be editable.

interface java.awt.event.ActionListener:
void actionPerformed(ActionEvent e)
    Invoked when an action occurs.

class java.awt.event.ActionEvent:
String getActionCommand()
    Returns the command string associated with this action.

Object getSource()
    Returns the object on which the event occurred.

interface java.awt.event.KeyListener:
void keyReleased(KeyEvent e)
    Invoked when a key has been released.

class java.awt.event.KeyEvent:
Object getSource()
    The object on which the Event initially occurred.

char getKeyChar()
    Returns the character associated with the key in this event.

Layout managers
All layout manager classes implement the interface java.awt.LayoutManager

class java.awt.BorderLayout:
BorderLayout(int hgap, int vgap)
    Constructs a border layout with the specified gaps between components.

```

Fields:

```

public static final String CENTER
    The center layout constraint (middle of container).
public static final String EAST
    The east layout constraint (right side of container).
public static final String WEST
    The west layout constraint (left side of container).
public static final String NORTH
    The north layout constraint (top of container).
public static final String SOUTH
    The south layout constraint (bottom of container).

```

class java.awt.BoxLayout:

BoxLayout(Container target, int axis)
 Creates a layout manager that will lay out components along the given axis.

Parameters:

```

target - the container that needs to be laid out
axis - the axis to lay out components along. Can be one of: BoxLayout.X_AXIS or
BoxLayout.Y_AXIS

```

class java.awt.FlowLayout:

FlowLayout(int align, int hgap, int vgap)
 Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps. The value of the alignment argument must be one of FlowLayout.LEFT, FlowLayout.RIGHT, or FlowLayout.CENTER.

class java.awt.GridLayout:

GridLayout(int rows, int cols, int hgap, int vgap)
 Creates a grid layout with the specified number of rows, columns, horizontal and vertical gaps. All components in the layout are given equal size.

Popup dialogues

class javax.swing.JOptionPane:

**static void showMessageDialog(Component parentComponent,
 Object message, String title, int messageType)**
 Brings up a dialog that displays a message using a default icon determined by the messageType parameter.

static String showInputDialog(Object message)
 Shows a question-message dialog requesting input from the user.

**static int showConfirmDialog(Component parentComponent,
 Object message, String title, int optionType,
 int messageType)**
 Brings up a dialog, where the number of choices is determined by the optionType parameter.

message types	
static int ERROR_MESSAGE	Used for error messages.
static int INFORMATION_MESSAGE	Used for information messages.
static int QUESTION_MESSAGE	Used for questions.
static int WARNING_MESSAGE	Used for warning messages.
option types used for showConfirmDialog:	
public static final int YES_NO_OPTION	
public static final int YES_NO_CANCEL_OPTION	
public static final int OK_CANCEL_OPTION	
public static final int YES_OPTION	
public static final int NO_OPTION	
public static final int CANCEL_OPTION	
public static final int OK_OPTION	
return values:	
public static final int YES_OPTION	Return value from class method if YES is chosen.
public static final int NO_OPTION	Return value from class method if NO is chosen.
public static final int CANCEL_OPTION	Return value from class method if CANCEL is chosen.
public static final int OK_OPTION	Return value from class method if OK is chosen.