Här finns utdrag från Javas API för de klasser som ni kan tänkas ha användning för lösa uppgifterna på tentan.

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.

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 o1, T o2)`
Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

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

interface java.io.Serializable

`empty interface`

Data Structures

interface java.util.List<E>:

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

`boolean` **`add`**`(E element)`
Appends the specified element to the end of this list (optional operation).

`void` **`clear`**`()`
Removes all of the elements from this list (optional operation).

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

`boolean` **`equals`**`(Object obj)`
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 element)`
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.

`E` **`remove`**`(int index)`
Removes the element at the specified position in this list(optional operation) .

`E` **`set`**`(int index, E element)`
Replaces the element at the specified position in this list with the specified element (optional operation) .

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

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

`E[]` **`toArray`**`(E[] arr)`
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.HashSet<E> och **class java.util.TreeSet<E>:**

HashSet()
Constructs a new, empty HashSet.

HashSet(Collection<? extends E> c)
Constructs a new set containing the elements in the specified collection.

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

TreeSet(Collection<? extends E> c)
Constructs a new tree set containing the elements in the specified collection, sorted according to the natural ordering of its elements.

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> och **class java.util.TreeMap<K,V>:**

HashMap()
Constructs an empty HashMap.

HashMap(Map<? extends K, ? extends V> m)
Constructs a new HashMap with the same mappings as the specified Map.

TreeMap()
Constructs an empty TreeMap.

TreeMap(Map<? extends K, ? extends V> m)
Constructs a new TreeMap containing the same mappings as the given map, ordered according to the natural ordering of its keys.

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.

Iteratorer

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.

Observable och Observer

class java.util.Observable:

`void` **`addObserver`**`(Observer obs)`
Adds an observer to the set of observers for this object.

`void` **`deleteObserver`**`(Observer obs)`
Delete an observer from 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.

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

interface java.util.Observer:

`void` **`update`**`(Observable obs, Object arg)`
This method is called whenever the observed object is changed.

PropertyChangeSupport och PropertyChangeListener

class java.beans.PropertyChangeSupport:

`PropertyChangeSupport(Object source)`
Constructs a PropertyChangeSupport object.

`void addPropertyChangeListener(PropertyChangeListener l)`
Add a PropertyChangeListener to the listener list.

`void removePropertyChangeListener(PropertyChangeListener l)`
Remove a PropertyChangeListener from the listener list.

`void firePropertyChange(String name, Object old, Object new)`
Report a bound property update to any registered listeners.

interface java.beans.PropertyChangeListener:

`void propertyChange(PropertyChangeEvent e)`
This method gets called when a bound property is changed.

class java.beans.PropertyChangeEvent:

`Object getNewValue()`
Gets the new value for the property, expressed as an Object.

`Object getOldValue()`
Gets the old value for the property, expressed as an Object.

`Object getSource()`
Gets the object on which the Event initially occurred.

Threads and Runnable

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.

`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.

`void interrupt()`

If this thread is blocked in an invocation of the `wait()`, `wait(long)`, or `wait(long,int)` methods of the `Object` class, or of the `join()`, `join(long)`, `join(long, int)`, `sleep(long)`, or `sleep(long,int)` methods of this class, then its interrupt status will be cleared and it will receive an `InterruptedException`.

I/O: Streams and Files

General

`void close() throws IOException`
Closes the stream.

class java.io.FileReader:

`FileReader(String fileName) throws FileNotFoundException`
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.

`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) throws IOException`
Writes a portion of an array of characters.

`void write(int c) throws IOException`
Writes a single character.

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

`void flush() throws IOException`
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() throws IOException`
Flushes the stream.

`void newLine() throws IOException`
Writes a line separator.

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

`void write(int c) throws IOException`
Writes a single character.

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

class java.io.FileInputStream:

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

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

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

`long skip(long n) throws IOException`
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) throws IOException`
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) throws IOException`
Reads up to `len` bytes of data from the contained input stream into an array of bytes.

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

`byte readByte() throws IOException`
Reads and returns one input byte.

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

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

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

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

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

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

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

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

class java.io.ObjectInputStream:

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

`Object readObject() throws IOException, ClassNotFoundException`
Read an object from the ObjectInputStream.

class java.io.BufferedInputStream:

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

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

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

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

`long skip(long n) throws IOException`
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() throws IOException`
Reads a single character.

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

class java.io.FileOutputStream:

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

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

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

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) throws IOException`
Writes `len` bytes from the specified byte array starting at offset `off` to the underlying output stream.

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

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

`void` **`writeByte`**`(int v) throws IOException`
Writes out a `byte` to the underlying output stream as a 1-byte value.

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

`void` **`writeChar`**`(int v) throws IOException`
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) throws IOException`
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) throws IOException`
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) throws IOException`
Writes an `int` to the underlying output stream as four bytes, high byte first.

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

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

class java.io.ObjectOutputStream:

`ObjectOutputStream``(OutputStream out) throws IOException`
Creates an ObjectOutputStream that writes to the specified OutputStream.

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

class java.io.BufferedOutputStream:

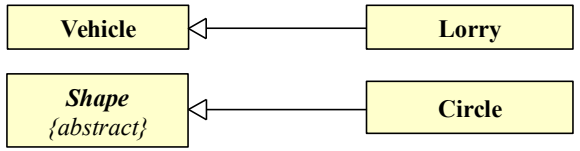
`BufferedOutputStream``(OutputStream out)`
Creates a new buffered output stream to write data to the specified underlying output stream.

`void` **`flush`**`() throws IOException`
Flushes this buffered output stream.

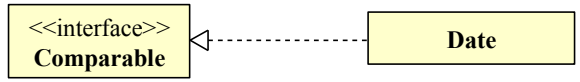
`void` **`write`**`(byte[] b, int off, int len) throws IOException`
Writes `len` bytes from the specified byte array starting at offset `off` to this buffered output stream.

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

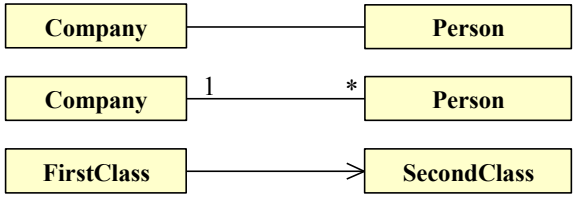
Generalisering:



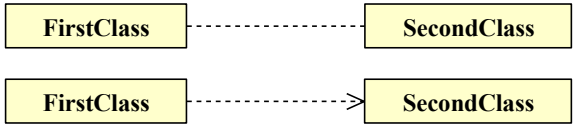
Realisering:



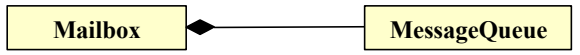
Association:



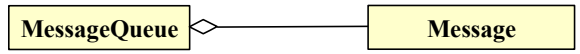
Beroende:



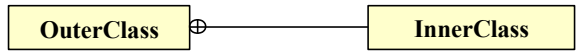
Komposition:



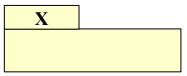
Aggregation:



Inkapsling:



Paket:



Synlighet:

\+ public
\- private
\# protected
~ paketsynlighet