Aufgabe 1:

Bearbeiten Sie die Aufgaben 5.61 und 5.62 in Ihrem Buch.

5.61:

```
/**
* Star Wars
*
* @author Christian Bontekoe, Felix Rohrer
* @version 1.0
*/
public class NameGenerator
{

    /**
     * Constructor for objects of class NameGenerator
     */
    public NameGenerator()
    {

    }

    /**
     * Generate Star Wars Name
     *
     * @param  surname:
     * @param  prename:
     */
    public void generateStarWarsName(String surname, String prename, String maidname, String city)
    {
        String tmpPrename = surname.substring(0,3) + prename.substring(0,2);
        String tmpSurname = maidname.substring(0,2) + city.substring(0,3);
        System.out.println("first name: " + tmpPrename + "\n" + "last name: " + tmpSurname);
    }
}
```

5.62:

toUpperCase erstellt ein neuer String, dieser muss gespeichert und dann ausgegeben werden.

```
public void printUpper(String s)
{
  s = s.toUpperCase();
  System.out.println(s);
}
```

Aufgabe 2:

Öffnen Sie das Projekt Lab-Classes vom Kapitel 1.
Im Konstruktor der Klasse `LabClass` wird eine maximale Klassengrösse angegeben. Wie wird die Einhaltung der Klassengrösse sichergestellt?

In der Methode enrollStudent() wird überprüft ob die Klassengrösse noch nicht erreicht ist.

Ist das notwendig, wenn Sie sich die entsprechende Feld-Definition ansehen?

Nein, die ArrayList kann unendlich viele Elemente enthalten.

Gibt es eine andere mögliche Felddefinition, die die Einhaltung einer maximalen Anzahl Studierender erfordern würde?

Ja, wenn z.B. ein Array of Strings verwendet würde.

String[] students = new String[20];

Aufgabe 3:

Die Klasse `LabClass` des Lab-Classes Projekts ist so zu verändern, dass der Zugriff auf einzelne Studierende nicht mehr über den Index, sondern über die Matrikelnummer (String) erfolgt. Welche Bibliotheksklasse verwenden Sie hierzu? Skizzieren Sie Ihre Lösung auf Papier.

Mittels HashMap:

```
private HashMap<String, String> students = new HashMap<String, String>();

students.put(name, matrikelnumer);
students.get(matrikelnumer);
```

Aufgabe 4:

Schreiben Sie in der Klasse, welche die Methode `fak()` implementiert, eine Methode `main()`. Diese ruft die Methode zur Fakultätsberechnung für den Wert 5 auf und gibt das Ergebnis mit `System.out.println()` aus. Kompilieren Sie Ihre Klasse von der Konsole (unter Windows: Start -> Ausführen -> cmd) aus (javac) und starten Sie sie mit java von der Konsole aus.1
Hinweis: Für diese Aufgabe gibt es zwei Möglichkeiten: Entweder Sie erzeugen ein Objekt Ihrer Klasse in der Methode `main()` und rufen die Methode `fak()` für dieses Objekt auf, oder Sie definieren Methode `fak()` als statische Methode.
Testen Sie beide Varianten.

```
    /**
     * Main
     * @param args
     */
    public static void main(String[] args)
    {
        Fakultaet objFak = new Fakultaet();
        System.out.println(objFak.fak(5));

    }


public static long fak(long n)
{ … }

System.out.println(fak(5));
```

Aufgabe 5:

Implementieren Sie eine Klasse `Person` mit einer Klassenvariablen, welche als Wert die Anzahl erzeugter Objekte der Klasse Person enthält.

```
/**
 * Write a description of class Person here.
 *
 * @author Christian Bontekoe, Felix Rohrer
 * @version 1.0
 */
public class Person
{
    private static int countPerson;

    /**
     * Constructor for objects of class Person
     */
    public Person()
    {
        // initialise instance variables
        countPerson++;
    }

    /**
     * Print out Count Person
     */
    public void printCountPerson()
    {
        System.out.println("Anzahl Personen: " + countPerson);
    }
}
```

Aufgabe 6:

Studieren Sie die API Dokumentation der Klasse LinkedList.

http://download.oracle.com/javase/6/docs/api/java/util/LinkedList.html

Schreiben Sie eine Klasse MyList, die folgende Eigenschaft hat:

- ein Attribut, das eine LinkedList von Strings enthält.

und einige der folgenden Methoden zur Verfügung stellt:

- eine Methode, um die Anzahl Elemente der Liste auszugeben.
- eine Methode, die ein Element zu Beginn der Liste einfügt.
- eine Methode, die ein Element am Ende der Liste einfügt.
- eine Methode, die das Element am Beginn der Liste löscht.
- eine Methode, die das Element am Ende der Liste löscht.
- eine Methode, die ausgibt, ob ein bestimmter String bereits in der Liste vorhanden ist.
- eine Methode, die den Index des ersten Vorkommens eines bestimmten Strings in der Liste ausgibt.
- eine Methode, die alle Elemente der Liste vom ersten bis zum letzten Element ausgibt. Verwenden Sie hierzu den ListIterator.
- eine Methode, die alle Elemente der Liste vom letzten bis zum ersten Element ausgibt. Verwenden Sie hierzu den ListIterator.

```
import java.util.LinkedList;
import java.util.ListIterator;

/**
* Write a description of class MyList here.
*
* @author Felix Rohrer, Christian Bontekoe
* @version 1.0
*/
public class MyList
{
    private LinkedList<String> myList;

    /**
     * Constructor for objects of class MyList
     */
    public MyList()
    {
        myList = new LinkedList<String>();
        myList.add("Test");
        myList.add("Hans");
        myList.add("Fritz");
        myList.add("Susi");
    }

    /**
     * Count Object in List
     */
    public void countObjInList()
    {
        System.out.println("Anzahl Einträge: " + myList.size());
    }

    /**
     * Insert first Object
     */
    public void insertFirstObj()
    {
        myList.addFirst("ERSTER");
    }

    /**
     * Insert last Object
     */
    public void insertLastObj()
    {
        myList.addLast("LETZTER");
    }

    /**
     * Delete first Object
     */
    public void deleteFirstObj()
```

```
        {
            myList.removeFirst();
        }

        /**
         * Delete last Object
         */
        public void deleteLastObj()
        {
            myList.removeLast();
        }

        /**
         * Contains the Objects "ERSTER"
         */
        public void containObj()
        {
            if (myList.contains("ERSTER"))
            {
                System.out.println("Der Eintrag ist schon vorhanden!");
            }
            else
            {
               System.out.println("Der Eintrag ist NICHT vorhanden!");
            }
        }

        /**
         * Index of first Object "ERSTER"
         */
        public void indexFirstObj()
        {
            System.out.println("Der Index des ersten Objektes lautet: " + myList.indexOf("ERSTER"));
        }

        /**
         * Print all Objects (forward)
         */
        public void printAllObjs()
        {
            ListIterator<String> itr = myList.listIterator();
            while(itr.hasNext())
            {
                System.out.println(itr.next());
            }
        }

        /**
         * Print all Objects (reverse)
         */
        public void printAllObjsRev()
        {
            ListIterator<String> itr = myList.listIterator(myList.size());
            while(itr.hasPrevious())
            {
                System.out.println(itr.previous());
            }
        }
    }
```

Dokumentieren Sie Ihre Klasse sowie jede Methode Ihrer Klasse mit Hilfe von javadoc.

Class MyList

```
java.lang.Object
  └MyList
```

```
public class MyList
extends Object
```

Write a description of class MyList here.

Version:

1.0

Author:

Felix Rohrer, Christian Bontekoe

Constructor Summary

MyList()	Constructor for objects of class MyList

Method Summary

void	**containObj**() Contains the Objects "ERSTER"
void	**countObjInList**() Count Object in List
void	**deleteFirstObj**() Delete first Object
void	**deleteLastObj**() Delete last Object
void	**indexFirstObj**() Index of first Object "ERSTER"
void	**insertFirstObj**() Insert first Object
void	**insertLastObj**() Insert last Object
void	**printAllObjs**() Print all Objects (forward)
void	**printAllObjsRev**() Print all Objects (reverse)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

1.1.1 MyList

`public MyList()`

Constructor for objects of class MyList

Method Detail

1.1.2 containObj

`public void containObj()`

Contains the Objects "ERSTER"

1.1.3 countObjInList

`public void countObjInList()`

Count Object in List

1.1.4 deleteFirstObj

`public void deleteFirstObj()`

Delete first Object

1.1.5 deleteLastObj

`public void deleteLastObj()`

Delete last Object

1.1.6 indexFirstObj

`public void indexFirstObj()`

Index of first Object "ERSTER"

1.1.7 insertFirstObj

`public void insertFirstObj()`

Insert first Object

1.1.8 insertLastObj

`public void insertLastObj()`

Insert last Object

1.1.9 printAllObjs

`public void printAllObjs()`

Print all Objects (forward)

1.1.10 printAllObjsRev

`public void printAllObjsRev()`

Print all Objects (reverse)