Programmieren 1
Lernteam Semesterwoche 9

Aufgabe 1:
Bearbeiten Sie die Aufgaben 5.61 und 5.62 in Ihrem Buch.

Aufgabe 2:
Öffnen Sie das Projekt Lab-Classes des Kapitel 1. Im Konstruktor der Klasse LabClass wird eine maximale Klassengrösse angegeben. Wie wird die Einhaltung der Klassengrösse sichergestellt? Ist das notwendig, wenn Sie sich die entsprechende Feld Definition ansehen? Gibt es eine andere mögliche Felddefinition, die die Einhaltung einer maximalen Anzahl Studierender erfordern würde?
→ Ja, die Klassengrösse wird überprüft mit:

```
if(students.size() == capacity) {
       System.out.println("The class is full, you cannot enrol.");
}
else {
       students.add(newStudent);
}
```

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.

```
//Lösungsbeispiel (Änderungen wurden rot markiert)

import java.util.*;

public class LabClass
{
    private String instructor;
    private String room;
    private String timeAndDay;
    private HashMap<String, Student> studentsMap;
    private int capacity;

    public LabClass(int maxNumberOfStudents)
    {
        instructor = "unknown";
        room = "unknown";
        timeAndDay = "unknown";
        studentsMap = new HashMap<String, Student>();
        capacity = maxNumberOfStudents;
    }

    public void enrollStudent(Student newStudent)
    {
        if(studentsMap.size() == capacity) {
            System.out.println("The class is full, you cannot enrol.");
        }
        else {
            studentsMap.put(newStudent.getStudentID(), newStudent);
        }
    }

    public int numberOfStudents()
    {
        return studentsMap.size();
    }
```

```
    public void setRoom(String roomNumber)
    {
        room = roomNumber;
    }

    public void setTime(String timeAndDayString)
    {
        timeAndDay = timeAndDayString;
    }

    public void setInstructor(String instructorName)
    {
        instructor = instructorName;
    }

    public void printList()
    {
        System.out.println("Lab class " + timeAndDay);
        System.out.println("Instructor: " + instructor + "   room: " + room);
        System.out.println("Class list:");

        Set<String> setMap = studentsMap.keySet();
        for(String ID : setMap) {
            System.out.println("ID="+ID+" Name="+studentsMap.get(ID).getName());
        }
        System.out.println("Number of students: " + numberOfStudents());
    }
}
```

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.
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 Sie Methode fak() als statische Methode.
Testen Sie beide Varianten.

```
public class Fak
{
    public static void main(String[] args)
    {
       long n = Long.parseLong(args[0].trim());      //Umwandlung von String → long
       Fak f = new Fak();
       System.out.println(f.fak(n));
    }

    public Fak()
    {

    }

    public long fak(long n)
    {
        if(n==0 || n==1)            //Rekursionsbedingung
        {
            return 1;               // Rekursionsbasis
        }
        else
        {
            return n * fak(n-1);    //Rekursionsvorschrift
        }
    }
}
```

Aufgabe 5:

Studieren Sie die API Dokumentation der Klasse LinkedList.
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.
Dokumentieren Sie Ihre Klasse sowie jede Methode Ihrer Klasse mit Hilfe von javadoc.

Zusatzaufgabe: Rekursive Fibonnaci-Funktion

Verwendung einer rekursiven Methode, welche die mathematischen Definition 1:1 übernimmt.

```
//Lösungsbeispiel

private  int fibonacci(int n)
{
 if(n<=1)
 {
      return n;                                          //Rekursionsbasis
 }
 else
 {
      return(fibonacci(n-1) + fibonacci(n+2));      //Rekursionsvorschrift
 }
}
```