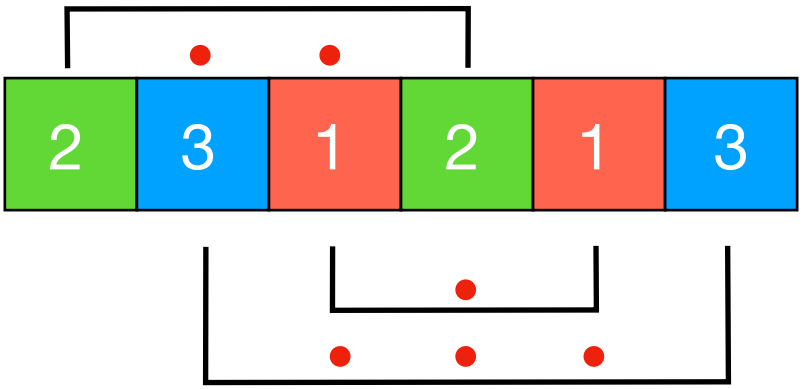
4. Praktikum: Rekursion

"To iterate is human, to recurse divine."

(L. Peter Deutsch)



Bearbeitungszeitraum:	11.01.21 - 25.01.21, 9.15 Uhr
Gesamtpunktzahl:	5 Abgabepunkte + (4 + 12) Leistungspunkte
Praktikumsziel:	Sie üben die Verwendung rekursiver Funktionen (Methoden).
Voraussetzungen:	Kenntnisse aus der Vorlesung Nutzung des Java-Compilers und der Java *virtual machine* Strukturierte Anweisungen, (statische) Methoden, Felder, Rekursion
Aufgabe:	Implementieren Sie pro Teilaufgabe ein Java-Programm, welches das jeweils beschriebene Problem löst.
Abgabe:	Die vollständig implementierte Klassen **`Roman.java`** und **`Riddle.java`** senden Sie wie üblich bei **Moodle** als Lösung ein. *Bitte verwenden Sie genau diesen Dateinamen und achten Sie auf Groß- und Kleinschreibung.* ***ACHTUNG: Nur Gruppenabgaben möglich***

Teilaufgabe 1: Römische Zahlen (4 Punkte)

Implementieren Sie in einem Java-Programm **`Roman.java`** eine Funktion

```
static String roman(int n)
```

die (positive) ganze Zahlen im Bereich von 1 bis 5000 im Römischen Zahlensystem (als String) in der üblichen Schreibweise (Ziffernzeichen ***IVXLCDM***) gemäß

https://de.wikipedia.org/wiki/Römische_Zahlschrift#Subtraktionsregel

darstellt. Überlegen Sie sich insbesondere, wie dabei Rekursion zum Einsatz kommen kann.
Benutzen Sie dieses Quelltextfragment als Ausgangspunkt:

```
public class Roman {
    static String roman(int n) { /* TODO */ }
    public static void main(String[] args) {
       // Behandlung fehlender oder falscher Eingabeparameter
       assert(1<=N && N<=5000);
       System.out.println(roman(N));
    }
}
```

Es folgen einige Beispielaufrufe, die das Format der Ausgabe verbindlich vorgeben. Falls ein Parameter beim Programmaufruf angegeben wurde, können Sie davon ausgehen, dass es sich um eine ganze Zahl handelt.

```
% java Roman
Bitte eine Zahl als Parameter angeben.
% java Roman 0
Die Zahl muss zwischen 1 und 5000 liegen.
% java Roman 1
I
% java Roman 5000
MMMMM
```

Teilaufgabe 2: Ein Zahlenrätsel (12 Punkte)

In einem Feld der Länge 2*N sollen die Zahlen von 1 bis N so platziert werden, dass jede Zahl genau zweimal vorkommt und zwischen den beiden Platzierungen jeder Zahl n exakt n andere Zahlen in dem Feld stehen. Die Abbildung unter dem Titel der Aufgabe zeigt eine Lösung für N=3.

Implementieren Sie eine Klasse **`Riddle`**, die das Problem für 0 < N <= 15 löst, indem es entweder

- die Zeichenkette "keine Loesung“ ausgibt, wenn keine Lösung existiert oder
- falls Lösungen existieren für N <10 alle (verschiedenen) Lösungen nacheinander (im weiter unten angegebenen Format) zeilenweise gefolgt von einer Zeile mit der Anzahl der gefundenen Lösungen ausgibt oder
- falls Lösungen existieren für N>=10 nur die Anzahl der verschiedenen Lösungen ausgibt.

Als verschieden gelten zwei Lösungen dann, wenn sie NICHT durch Spiegelung (rückläufiges Auslesen des Feldes) aufeinander identisch abgebildet werden können. In diesem Sinne sind für N=3 die Lösungen

```
2 3 1 2 1 3   und
3 1 2 1 3 2
```

identisch. In diesem Fall soll NUR die Lösung ausgegeben werden, die mit der kleinsten Zahl beginnt (hier die erste von beiden).

Überlegen Sie sich eine Lösung, die das Problem geeignet rekursiv angeht! Versuchen Sie dazu, Ihre Vorgehensweise beim Lösen mit Papier und Bleistift (und Radiergummi?) so zu formulieren, dass ein rekursiver Ansatz erkennbar wird.

Halten Sie sich an das Format der folgenden (korrekten) Musterausgaben:

```
% java Riddle 1
keine Loesung
% java Riddle 2
keine Loesung
% java Riddle 3
231213
eine Loesung
% java Riddle 7
23726351417654
26721514637543
23627345161475
36713145627425
26327435614175
26325734615147
24723645317165
46171435623725
52462754316137
```

```
35723625417164
27423564371516
25623745361417
52642753461317
41617435263275
41716425327635
34673245261715
15146735423627
15163745326427
15173465324726
16135743625427
14167345236275
14156742352637
15167245236473
16172452634753
17126425374635
17125623475364
26 Loesungen
% java Riddle 10
keine Loesung
% java Riddle 11
17792 Loesungen
```

Hinweise

Auch wenn Java prinzipiell die Verwendung von Umlauten (und vielen anderen Sonderzeichen) erlaubt, sollten Sie im Praktikum generell darauf verzichten, um sich nicht vom sog. *encoding* abhängig zu machen. Das betrifft übrigens auch Texte in Kommentaren!

FAQ

1. **Q:** Was sind denn Abgabe- und Leistungspunkte?
   A: Wenn Sie die Praktikumsordnung gelesen haben, sollte klar sein, dass sie alle Aufgaben abgeben müssen. Die Abgabepunkte erhalten Sie dann, wenn Sie eine erkennbar ernst gemeinte Abgabe eingereicht haben. Die eigentlich in der jeweiligen Aufgabe geforderte Leistung wird dann in Abhängigkeit von der Erfüllung mit Leistungspunkten bewertet.

2. **Q:** Gibt es ein Zeitlimit für die Ausführung der **`main`**-Funktion ?
   A: Ja, für N<=12 sollte das Resultat in unter 2 Sekunden vorliegen. Über 12 darf es (u.U. sehr viel) länger dauern.

3. **Q:** Wenn ich in *eclipse* eine neue Klasse anlege (siehe 1. Aufgabe) kann ich diese einem sog. *Package* zuordnen. Soll/darf ich davon Gebrauch machen?
   A: Nein! Benutzen Sie immer die Voreinstellung (*default*), weil sich ansonsten Ihr Programm nicht im aktuellen Verzeichnis des Quelltextes ausführen lässt! Da dies bei der Korrektur zu aufwändigen Anpassungen ihrer Lösung führt, wird es u.U. mit Punktabzug 'bestraft' - vermeiden Sie es daher.

4. **Q:** Wer ist L. Peter Deutsch?
   A: Lesen Sie unter https://de.wikipedia.org/wiki/L._Peter_Deutsch nach.

ACHTUNG

Um die Korrektur der Aufgaben durch die Tutoren zu erleichtern, gelten wie immer (d.h. auch bei allen weiteren Aufgaben) die folgenden Regeln:

- Eingesandte Java-Programme, die nicht vom Java-(Referenz-)Compiler akzeptiert werden (d.h. Fehler bei der Übersetzung liefern) werden ohne weitere Inspektion mit 0 (Leistungs-) Punkten bewertet! Ob Sie die 5

Abgabepunkte erhalten, entscheiden wir in Abhängigkeit davon, ob Ihre Einsendung genügend mit der Aufgabe zu tun hat ;-)

- Halten Sie sich bei der Benennung von Klassen, Funktionen und Dateien immer an die Vorgaben und achten Sie darauf, dass Sie keine Packages verwenden (s.o.). Alle Abweichungen von diesen Regeln führen zu zusätzlichem Aufwand bei der Bewertung und daher auch zu Punktabzug.
- Die vom implementierten Programm erzeugten Ausgaben müssen das geforderte Format exakt einhalten (incl. Zeilenstruktur und Leerzeichen). Soll und Ist werden automatisch verglichen. Sofern Abweichungen festgestellt werden, gibt es Punktabzug, über dessen Höhe dann die Tutoren entscheiden! Die einzig zulässige Abweichung (eine die man bei der Textansicht des Programms und seiner Ausgaben u.U. gar nicht feststellen kann) besteht in der Codierung der Zeilenenden, die unter Windows mit zwei Zeichen (carriage return [/x0d] + line feed [/x0a]) unter Unix dagegen nur mit einem Zeichen (newline [/x0a]) dargestellt werden.