HTWK Leipzig, FIMN
Dipl.-Math. Dörte König

Anwendungsorientierte Programmierung II
Hinweise zur Übung2

1. Programm-Eingaben

Wenn nicht explizit etwas anderes angegeben wurde, benutzen Sie bitte für alle Nutzereingaben in den Übungsaufgaben die **Methoden der Klasse `Scanner`**.

2. Eingabeprüfung

Es ist zu prüfen, ob die **eingegebene Zahl** tatsächlich **im zulässigen Wertebereich** (0,4000) liegt. Im Fehlerfall ist dem Nutzer durch eine geeignete Schleife zu ermöglichen, die Eingabe so lange zu wiederholen, bis ein zulässiger Wert eingegeben wurde.

Erweiterte Eingabeprüfung (optional)

Weitere Eingabefehler können auftreten, wenn die Nutzereingabe nicht zum erwarteten Datenformat passt. Z.B. die Eingabe eines Buchstabens anstelle des erwarteten `int`-Wertes, würde unmittelbar zum Programmabbruch führen.

Durch eine **try-catch-Anweisung mit InputMismatchException** lassen sich solche Eingabefehler absichern. Im Programmblock der Exception muss - neben einer entsprechenden Fehlerausgabe für den Nutzer - auch der fehlerhafte Inhalt im Scanner-Objekt mit Hilfe der `reset()`-Methode entfernt werden:

```
Scanner input;
int dZahl = 0;
do
{
       input=new Scanner(System.in);
       try
       {
              ...
              dZahl= input.nextInt();
              ...

       }catch(InputMismatchException e)
              {
                     //Fehlermeldung ...
                     input.reset();
              }
}while(...);
```

Da durch die `reset()`-Methode auch alle Statusinformationen des Scannerobjekts (z.B. der Eingabestrom `System.in`) zurückgesetzt werden, muss vor der nächste Eingabe immer ein neues Scannerobjekt erzeugt werden.

3. Konvertierung

Rechenbeispiel (mit Anwendung der Subtraktionsregel):

```
3974 = 3*1000+1*900+0*500+0*400+0*100+0*90+1*50+0*40+2*10+0*9+0*5+1*4+0*1
     =  MMM  + CM                          + L       + XX         + IV
     =  MMMCMLXXIV
```

Im Programm erreicht man die Umrechnung der arabischen Zahl, in dem man mit der größten römischen Ziffer (`M`-für das vorgegebene Intervall) beginnend, deren Wert (1000) so häufig wie möglich

von der umzurechnenden Zahl abzieht und die römischen Ziffern dabei speichert. Analog verfährt man im nächsten Schritt mit der zweitgrößten römischen Ziffer etc.

Pseudocode:
```
solange arabZahl>=1000
       arabZahl-=1000
       roemZahl+="M"

...
```

Hinweis: Bei allen Werten, die keine Zehnerpotenz sind, genügt auch die einmalige Prüfung, da sie nur ein Mal subtrahiert werden können.

4. Programmwiederholung

Laut Aufgabenstellung soll die Konvertierung von Zahlen mehrfach durchgeführt werden können, ohne das Programm neu zu starten. Dies erreicht man durch eine endgeprüfte Schleife, die die Antwort des Nutzers auf die Frage nach Wiederholung (J/N) auswertet. Das Einlesen der Nutzereingabe könnte dann (unter Verwendung der Instanz `input` der Klasse `Scanner`) so aussehen:

```
char wdh=input.next().toUpperCase().charAt(0);
```

`next()`	- liest den eingegebenen Token
`toUpperCase()`	- wandelt Klein- in Großbuchstaben um
`charAt(0)`	- holt das erste Zeichen (vom Index 0) aus dem String

5. Kurze Lösung (optional)

Die Konvertierung nach der Methode aus 3. enthält im Quellcode zahlreiche Wiederholungen, da für jeden Wert die gleiche Prüfung erfolgen muss.
Das lässt sich wesentlich abkürzen, wenn die arabischen und römischen Werte in Feldern gespeichert werden, wobei einander zugeordnete Werte auf dem jeweils gleichen Arrayindex stehen müssen:

```
int[] aWerte        = {1000,900,...};
String[] rWerte     = {"M","CM",...};
```

Bei der Konvertierung ist dann nur noch eine Schleife erforderlich, in der die Umrechnung für die zugeordneten Werte auf jedem Array-Index nach dem Pseudocode aus 3. erfolgen muss.