Aufgabe 1:
Es ist ein Spiel zu erstellen, in welchem jeder Spieler im Wechsel aus einem Gefäß (Topf) abwechselnd Murmeln nimmt.
Die Anzahl der Spieler wird vor dem Spielbeginn festgelegt und daraus resultiert die initiale Anzahl der sich im Topf befindlichen Murmeln. Die zu Beginn des Spieles im Topf enthaltenen Murmeln errechnen sich durch die Anzahl der Spieler multipliziert mit 3. Bedenken Sie das eine einfache Änderung des Multiplikationsfaktors später gewährleistet sein soll.
Das Spiel ist rundenbasiert und jeder Spieler soll jeweils im Wechsel pro Runde zwischen einer und drei Murmeln aus dem Topf entfernen. Wer die letzte Murmel nimmt verliert das Spiel.
Es versteht sich das nicht mehr Murmeln aus dem Topf entnommen werden können, als darin enthalten sind. Daher soll nach jedem Spielzug durch eine Funktion die maximale Anzahl der Murmeln welche der Spieler entnehmen kann berechnet werden. Bei einer ungültigen Eingabe soll die Abfrage nach den zu entnehmenden Murmeln so lange wiederholt werden bis eine gültige Eingabe erfolgt.

Im Nachfolgenden sehen Sie eine beispielhafte Ausgabe des Programms mit 3 Spielern:

```
Anzahl der Spieler? 3

Es befinden sich 9 Murmeln im Topf.
Spieler 1: Wieviel Murmeln möchten Sie entnehmen (1-3)? 3
Spieler 2: Wieviel Murmeln möchten Sie entnehmen (1-3)? 3
Spieler 3: Wieviel Murmeln möchten Sie entnehmen (1-3)? 2
Spieler 1: Wieviele Murmeln möchten Sie entnehmen (1)? 1
Spieler 1 hat das Spiel verloren
```

Aufgabe 2:
Erweitert Sie das in Aufgabe 1 beschriebene Spiel so, dass anstatt gegen eine Person auch gegen den Computer als Gegner gespielt werden kann. In diesem Fall ist die Anzahl der Spieler fest auf zwei eingestellt. Der Computer soll die Rolle des zweiten Spielers einnehmen. Die Berechnung der Anzahl der Murmeln die vom Computer entfernt wird soll zufallsgesteuert sein. Verwenden Sie hierzu die Methode *Math.random()* aus der Math Biliothek. Beachten Sie die in Aufgabe 1 angegebenen Grenzen.

Beispiel für die Verwendung der Math.random() Funkion

```
// Generiere eine Zufallszahl r
static Random r = new Random();
// Für 256 setzen Sie den maximalen Wert ein den Ihre Zahl
annehmen darf
static int c = r.nextInt(256);
```

Aufgabe 3:
Ersetzen Sie den in Aufgabe 1 verwendeten festen Faktor 3 in eine Zufallszahl welche die Zahlenwerte zwischen 2 und 4 annehmen kann. Die Zufallszahl soll durch die Methode *Math.random()* aus der Math Bibliothek generiert werden.

Aufgabe 4:
Erstellen Sie ein Programm welches eine Person beim Lernen des 10-Finger-Systems auf der Tastatur unterstützt.Dazu sollen in einem Array die einzugebenden Texte in Form eines Strings vordefiniert werden. Es sind nur Alphanumerische Zeichen (A-Z,a-z und 0-9) und das Leerzeichen als Eingabe erlaubt. Die Texte werden nacheinander aus dem Array expandiert und auf der Konsole mit einem Zeilenumbruch ausgegeben. Der Benutzer muss nun versuchen die dargestellte Textzeile, Zeichen für Zeichen, korrekt abzutippen.
Sowohl korrekte als auch falsche Eingaben sollen am Ende jeder Zeile dargestellt werden. Nach der Eingabe aller im Array enthaltener Texte soll eine Ausgabe über alle aufgetretenen Fehler und korrekten Eingaben der Zwischenschritte ausgegeben werden. Die Ausgabe soll auch einen prozentualen Wert über die fehlerhaften Eingaben enthalten wie in dem Listing als Beispiel dargestellt wird.

```
jjfff ff jfj jff jjfj ffj fff jff fjfj jff jjff fjff
jjfff ff jfj jff jjfj ffj fkf jff fjfj jff jjff fjfk
Sie haben bei 53 Eingaben 2 Fehler gemacht
dk kdd kkk ddkdk dkd kdkk kkkkd kdd ddkk dddkd ddk
dk ked klk djkdk dgd kdkk kkkkd kdd ddkk dddkd ddk
Sie haben bei 51 Eingaben 5 Fehler gemacht
Bei insgesamt 104 Zeichen haben Sie 7 Fehler gemacht.
Ihre Fehlerquote liegt bei 6%
```

Aufgabe 5:
Stoppen Sie die Zeit von Beginn der Eingabe bis zum Ende eines Durchlaufs. Verwenden Sie hierfür die Funktionen initTimer() getDurationInMillis() aus der makeItSimple. Die Ausgabe soll am Ende des Programms in Sekunden ausgegeben werden.

```
Für die Eingabe haben Sie 10 Sekunden benötigt.
```