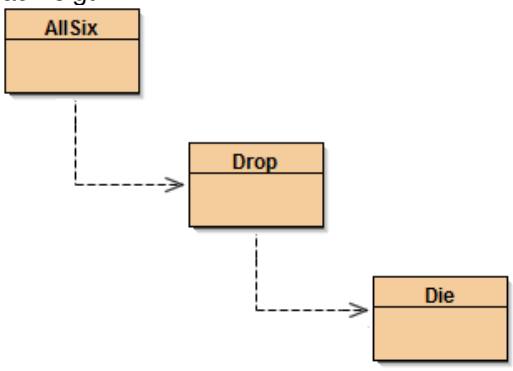
Programmieren 1
Zusatzaufgabe Semesterwoche 7

Aufgabe
Implementieren Sie eine Klasse AllSix, analog zu PlayDice. Die gewürfelten Werte sollen jedoch für eine spätere Analyse gespeichert werden.
1. Zeichnen Sie ein Klassendiagramm und ein Sequenzdiagramm, welches den Ablauf beim Würfeln aufzeigt.



2. Implementieren Sie eine Methode `public int diceAllSix(int nbrOfDies)`, welche mit der übergebenen Anzahl Würfeln solange würfelt, bis zum ersten Mal alles Sechser gewürfelt wurden. Die Augenzahlen aller Würfe werden in einer ArrayList abgespeichert (Hinweis: Implementieren Sie dazu eine Hilfsklasse Drop, welche alle Würfelwerte eines Wurfes speichert).

3. Implementieren Sie eine Methode `printTries()`, welche die gespeicherten Würfe ausgibt.

4. Zur Analyse soll folgende Methode implementiert werden:
`public int searchNbrOfAllEqual(int count)`. Ein Aufruf von z.B. `searchNbrOfAllEqual(3)` untersucht die gespeicherten Werte des letzten Spiels daraufhin, wie oft in einem Wurf nur 3er gefallen sind. Für `searchNbrOfAllEqual()` muss dann natürlich 1 resultieren!

Lösungsbeispiel

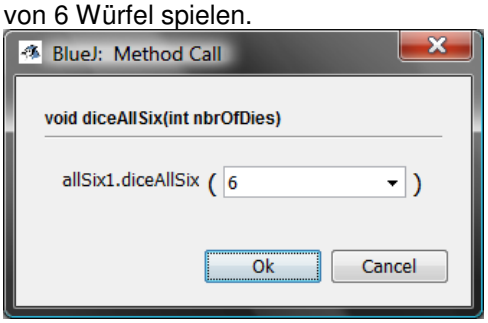
```
/* Klasse AllSix ************************************************************/

import java.util.*;
public class AllSix
{
    private ArrayList<Drop> aDrop;
    public AllSix()
    {

    }

    public void diceAllSix(int nbrOfDies)
    {
        aDrop = new ArrayList<Drop>();
        Drop drop;

        do
        {
            drop = new Drop(nbrOfDies);
            aDrop.add(drop);
        }while(!drop.checkAllDies(6));
    }

    public int searchNbrOfAllEqual(int Number)
    {
        int n = aDrop.size();
        int nbrOfMatches = 0;

        for(int i=0; i < n; i++)
        {
            Drop drop = aDrop.get(i);
            if(drop.checkAllDies(Number))
            {
                nbrOfMatches++;
            }
        }
        return nbrOfMatches;

    }

    public void printTries()
    {
        int n = aDrop.size();
        for(int x=0; x < n; x++)
        {
            Drop d = aDrop.get(x);
            System.out.print("Try "+x+" : ");

            for(int y=0; y < d.getN(); y++)
            {
                System.out.print(d.get(y).getValue());

            }
            System.out.println("");
        }
    }
}
```

```
/* Klasse Drop ************************************************************/

import java.util.*;
public class Drop
{
    private int n;
    private ArrayList<Die> aDie;

    public Drop(int nbrOfDies)
    {
        n = nbrOfDies;
        drop(n);
    }

    public void drop(int nbrOfDies)
    {
        aDie = new ArrayList<Die>();

        for(int i=0; i<nbrOfDies; i++)
        {
            Die d = new Die();
            aDie.add(d);
        }

    }

    public Die get(int index)
    {
        return aDie.get(index);
    }

    public int getN()
    {
        return n;
    }

    public boolean checkAllDies(int value)
    {
        int nbrOfMatches=0;
        boolean returnvalue = false;

        for(int i=0; i < n; i++)
        {
            if(aDie.get(i).getValue()==value)
            {
                nbrOfMatches++;
            }
        }

        if(nbrOfMatches==n)
        {
            returnvalue = true;
        }
        else
        {
            returnvalue = false;
        }
        return returnvalue;
    }

}
```

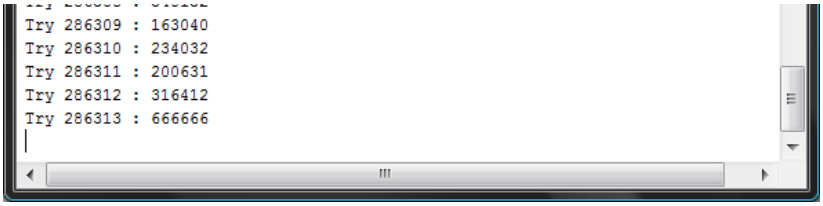
```
/* Klasse Die ********************************************************/

public class Die
{
    private int value;
    public Die()
    {
        value = getRandomValue();
    }

    public int getRandomValue()
    {
        return (int) (Math.random() * 7);
    }

    public int getValue()
    {
        return value;
    }

}
```

Funktionstest

- Methode „diceAllSix(int NbrOfDies)“ starten.
- Hier kann die Anzahl zu verwendender Würfel festgelegt werden. Wir wollen mit einem Set von 6 Würfel spielen.



- Bereits nach einigen Millisekunden Ausführungszeit (abhängig von der CPU Rechenleistung) hat der Computer genügend Würfelversuche gemacht, sodass alle 6 Würfel die Zahl 6 zeigen.
- Mit der Methode „printTries()“ werden alle Würfelversuche ausgegeben. Dieser Vorgang nimmt u.U. einige Zeit in Anspruch. Hier waren 286313 Würfelversuche nötig, um alle Würfel auf die Zahl 6 zu würfeln:



Tuning

Werden 7 oder mehr Würfel gewählt, kommt es höchstwahrscheinlich zu einer „Java OutOfMemoryException“. Der Heap Speicher der Java Virtual Machine ist standardmässig auf 128MB eingestellt. Um diesen Wert für unsere speicherintensive Anwendung zu erhöhen muss in der Datei „bluej.defs“ folgende Zeile eingetragen werden:

```
bluej.vm.args=-Xms512m -Xmx1024m
```

So kann BlueJ zwischen 512 und 1024MB Arbeitsspeicher im Heap beanspruchen. Würfelversuche mit 8 Würfel sind nun möglich. Aber bereits hier kommt die Anwendung wieder an ihre Speichergrenzen: Ein Würfelversuch mit 9 Würfel wäre bei 1024MB Heap Speicher bereits unmöglich…