Algorithmen und Datenstrukturen

Rekursion

Aufgabe 1: Türme von Hanoi [2 Punkte]

Sie haben in den Unterlagen (Folien) einen Lösungsansatz für die Problemstellung der *Türme von Hanoi*. Implementieren Sie einen *HanoiServer* mit der rekursiven Methode *moveDisk*. Geben Sie einen Text aus, der einer Anleitung für die Lösung entspricht:

- `move A to C`
- `move C to B`
- `...`

Die Anzahl der aufsteigend sortierten Scheiben (kleinste oben, grösste unten) auf Stab A soll als Parameter mitgegeben werden. Das Programm muss nur die Schritte ausgeben (von -> nach), keine Stacks der Stäbe oder ähnliches führen. D.h. der Rückgabewert des HanoiServers ist ein String mit den Schritten.

Aufgabe 2: Koch'sche Schneeflocke [4 Punkte]

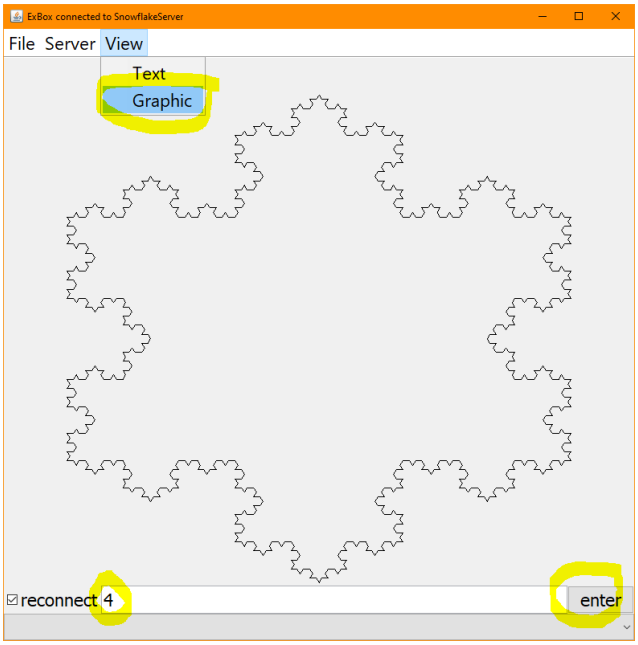
Es soll nicht direkt in der ExBox gezeichnet werden, sondern es soll ein CommandInterpreter `SnowflakeServer` entwickelt werden, der die Zeichnung mit Hilfe der Klasse Turtle erstellt und die fertige Graphik in `getTrace()` als XML String übergibt.

Hinweise:

- Instanzieren Sie in Ihrer `execute`-Methode eine Turtle Klasse, mit der Sie die Schneeflocke nach der Vorlage aus dem Skript zeichnen. Die `execute`-Methode des SnowflakeServer nimmt eine Ganzzahl (als String) entgegen, der die Anzahl Stufen angibt.
- Die Klasse Turtle ist bereits vollständig implementiert und muss nicht erweitert werden. Der SnowflakeServer soll die Turtle nutzen um die Schneeflocke zu zeichnen und als Rückgabe getTrace() der Turtle zurückgeben
- Mittels `getTrace()` kann der zurückgelegte Weg der `Turtle` als String abgefragt werden. Die Klasse `Turtle` liefert eine Spur des zurückgelegten Weges in der oben beschriebenen Form.

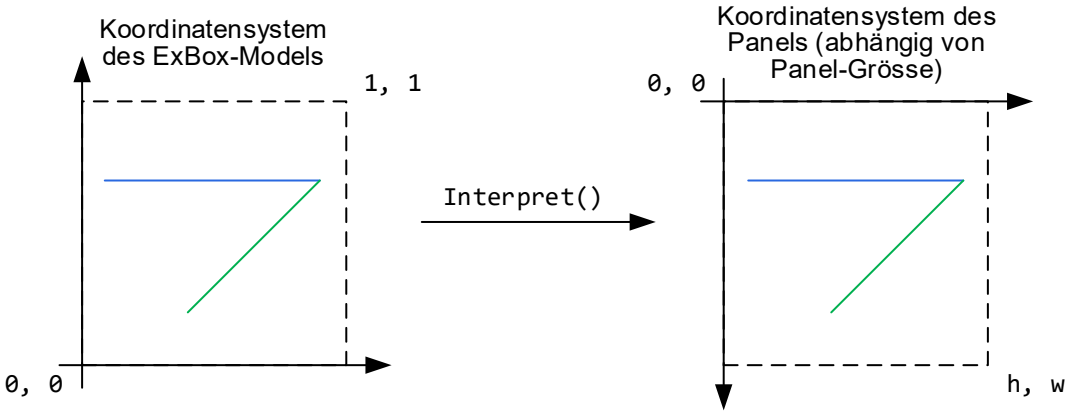
Hinweise zur Implementation der grafischen Darstellung:

Die nachfolgende Beschreibung der grafischen Implementation ist nur zur Info und muss nicht implementiert werden, diese steht Ihnen bereits zur Verfügung. Um Grafiken zu zeichnen wurde die ExBox um die Möglichkeit der graphischen Ausgabe erweitert. Es existiert ein `GraphicPanel`, mit dessen Hilfe Zeichnungen dargestellt werden können. Das Menu ist entsprechend um den Eintrag `View` mit den Untereinträgen `Text` und `Graphic` erweitert, damit kann zwischen graphischer und textueller Darstellung umgeschaltet werden.



Die Graphik wird als XML-String erzeugt, wobei jeweils eine Gruppe von vier Werten startX, startY, endX, endY), denen das Wort "<line" vorangestellt wird, eine Linie definieren (siehe unten). In der Behandlung der `interpret`-Methode in der Experiementierbox wird das Resultat mittels `setFigure()`dem `GraphicPanel` übergeben und dargestellt (falls die graphische Ansicht aktiviert ist).

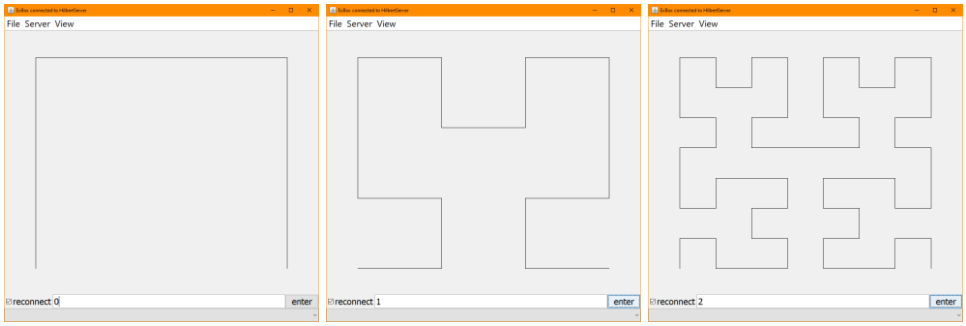
```
turtle.getTrace()
    <line x1="0.1" y1="0.7" x2="0.9", y2 ="0.7"/>
    <line x1="0.9" y1="0.7" x2="0.5", y2 = "0.2"/>
```



Dabei sind die Koordinaten der Modell-Zeichenfläche so normiert, dass die linke untere Ecke bei 0.0,0.0 liegt und die rechte obere Ecke bei 1.0,1.0. Die Normierung hat übrigens den Vorteil, dass für die Berechnung keine Annahmen über das Koordinaten-System der ExBox (z.B. aktuelle Grösse des Fensters) gemacht werden müssen. Das Einheitsquadrat wird in maximaler Grösse dargestellt.

Aufgabe 3: Hilbertkurve [4 Punkte]

In der Mathematik ist die Hilbert-Kurve eine stetige Kurve, die – durch Wiederholung ihres Konstruktionsverfahrens – jedem beliebigen Punkt einer quadratischen Fläche beliebig nahekommt und die Fläche vollständig ausfüllt. Die Hilbert-Kurve ist eine sogenannte raumfüllende oder FASS-Kurve. Sie wurde 1891 von dem deutschen Mathematiker David Hilbert entdeckt. Die Möglichkeit, mit einer stetigen eindimensionalen Kurve ein zweidimensionales Gebiet komplett abdecken zu können, war den Mathematikern des neunzehnten Jahrhunderts neu. Die Bilder zeigen die Hilbertkurve mit der Rekursionsstufe 0, 1 und 2.



Hinweise:

Die Hilbert-Kurve Stufe 0 sieht wie folgt aus (es fehlen noch die rekursiven Auftrufe für die weiteren Rekursionsstufen):

```
private void hilbert(int depth, double dist, double angle) {
   turtle.turn(-angle);
   // draw recursive
   turtle.move(dist);
   turtle.turn(angle);
   // draw recursive
   turtle.move(dist);
   // draw recursive
   turtle.turn(angle);
   turtle.move(dist);
   // draw recursive
   turtle.turn(angle);
}
```

```
int depth = Integer.parseInt(command);
double dist = 0.8 / (Math.pow(2,depth+1)-1);
turtle = new Turtle(0.1, 0.1);
hilbert(depth, dist, -90);
```

Überlegen Sie sich, wie Sie mittels rekursiven Aufrufen, von der ersten Kurve zur zweiten Kurve kommen. Dabei kann mit `angle` jeweils die Ausrichtung der U-förmigen Kurve festgelegt werden (alternierend 90 und -90 Grad).