Übungen zu Softwareentwicklung 1 **WS 2000/2001**
Übung 10

Name: _____ Tutor: _____

Matrikelnummer: _____ Punkte: _____

Gruppe: _____ **Abzugeben bis: 24. 1. 2001, 12:00**

Übungsleiter: _____ Bearbeitungsdauer: _____

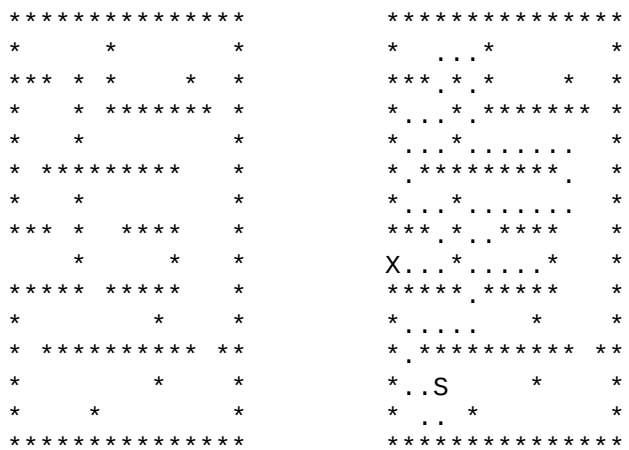
---

1 Labyrinth (16 Punkte)

Implementieren Sie eine Klasse *Maze* die ein Labyrinth darstellt. Das Feld des Labyrinths soll als 2-dimensionales *char*-Array implementiert werden. Entwickeln Sie einen **rekursiven** Algorithmus der feststellt ob das Labyrinth von einem gegebenen Startpunkt aus verlassen werden kann. Implementieren Sie das Labyrinth als Klasse mit (zumindest) folgender Schnittstelle:

```
class Maze extends Basic {
      Maze(int rows, int cols) {...}
      void init() {...}
      void print() {...}
      void startAt(int row, int col) {...}
      private boolean doExit(int row, int col) {...}
      static Maze newTestMaze(int type) {...}
}
```

Die folgende Abbildung zeigt ein Labyrinth (angegeben durch das Zeichen „*“) sowie ein Labyrinth mit dem vom Algorithmus gefundenen Weg. Die Zeichen „S“ und „X“ markieren den Startpunkt des Weges respektive den gefundenen Ausgang, das Zeichen „.“ den Weg selbst:



- Der Konstruktor ***Maze(int rows, int cols)*** legt ein uninitialisiertes Labyrinth mit *rows* Zeilen und *cols* Spalten an.
- Die Objekt-Methode ***init()*** initialisiert die Stellen des Labyrinths mit Hilfe eines Zufallsgenerators. Verwenden Sie dazu die Zufallsgenerator-Klasse *Random* aus *Basic*. Implementieren Sie den Zufallsgenerator als Klassen-Variable der Klasse *Maze* und erzeugen Sie ihn etwa wie folgt:

  ```
  myRandomGenerator = new Random(0, 1);
  ```

  Liefert der Aufruf *myRandomGenerator.readInt()* den Wert *0*, so soll die aktuelle Stelle im Labyrinth mit ' ' (Leerzeichen) als leer, ansonsten mit '*' als Mauer initialisiert werden.

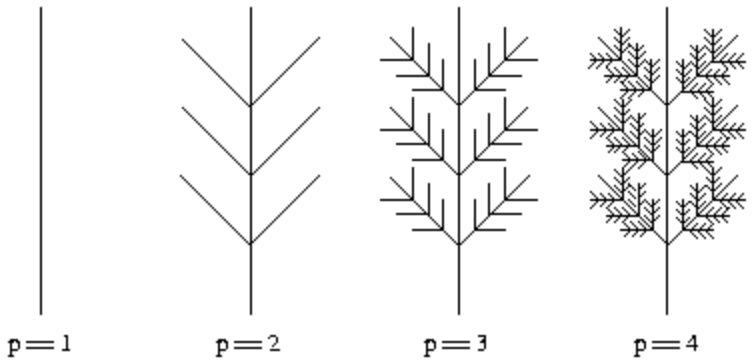
- Die Objekt-Methode ***print()*** druckt den aktuellen Zustand des Labyrinths aus.
- Wird die Objekt-Methode ***startAt(int row, int col)*** eines frisch initialisierten Labyrinths ausgeführt, so soll dieses zunächst ausgedruckt werden, dann soll versucht werden mittels der Methode *doExit(row, col)* das Labyrinth zu verlassen und schließlich soll das resultierende Labyrinth ausgegebenen werden. (Achten Sie darauf dass auch der Startpunkt mit „S“ markiert ausgegeben wird, sofern er nicht auf einer Mauer liegt!)
- Die **rekursive** Objekt-Methode ***doExit(int row, int col)*** findet und markiert, falls dies überhaupt möglich ist, für den Startpunkt S(*row*, *col*) den Weg aus dem Labyrinth. Dabei markiert sie im Labyrinth den Ausgang mit „X“ sowie den „beschrittenen“ Weg mit „.“ (siehe Abbildung). Die Methode liefert den Wert *true*, falls es bei gegebenem Startpunkt einen Weg aus dem Labyrinth gibt, andernfalls liefert sie den Wert *false*.
- Die Klassen-Methode ***newTestMaze(int type)*** liefert ein initialisiertes Labyrinth vom Typ *type* zurück. Die Methode dient vorrangig zum Testen. Ist der Typ *0*, so soll das Labyrinth dieses Angabezettels geliefert werden. Es sollen zumindest noch zwei weitere „optisch ansprechende“ Labyrinthe unterschiedlicher Größe für die Typen *1* und *2* zurückgegeben werden. Ein eindimensionales *char*-Array können Sie wie folgt schnell initialisieren:

```
char[] myRow = "*   * ******* *".toCharArray();
```

Testen Sie Ihr Programm mit einer eigenen Test-Klasse *MazeTester* mit den Test-Labyrinthen und mit einigen zufällig erzeugten Labyrinthen unterschiedlicher Größe.

2 Tannenzweig (8 Punkte)

Ein Tannenzweig kann als Fraktal gezeichnet werden, indem auf jeden Ast sechs weitere Äste aufgesetzt werden. Folgende Grafik verdeutlicht das Verfahren:



Gesucht ist eine Funktion *numberOfTwigs(p)* die je nach der gegebenen „Präzision“ *p* der Darstellung die Anzahl der Äste berechnet (*numberOfTwigs (1) == 1; numberOfTwigs (2) == 7; numberOfTwigs (3) == 43;* …).

Geben Sie an:
a) eine rekursive Definition für *numberOfTwigs(p)* für *p > 0*,
b) eine rekursive Java-Implementierung:

```
static int numberOfTwigsRec(int p);
```

c) und eine iterative Java-Implementierung:

```
static int numberOfTwigsIter(int p);
```

Testen Sie die Java-Funktionen!
Hinweis: Vergleichen Sie mit der rekursiven Berechnung von *n!* aus der Vorlesung.