Magnus Myréen
2015-09-18

Lab 3.2: Labyrintövning med rekursion

Denna labuppgift är ny. Jag vill gärna ha kommentarer (till myreen@chalmers.se) om uppgiften. Är uppgiften för lätt eller för svår? Är beskrivningen oklar eller onödigt lång?

Översikt

Given kod:

- `Labyrinth.java` definierar ett gränssnitt för objekt som beskriver labyrinter.
- `Lab.java` implementerar gränssnittet `Labyrinth`. *Ni behöver inte läsa denna fil.*
- `LabSolver.java` är filen ni bör editera.
- `LabSolverTest.java` är en testfil som rättarna kommer att köra.

Er uppgift är att skriva en rekursiv funktion som hittar en lösning till en slumpmässigt skapad labyrint. Labyrinterna skapas av den givna koden i `Lab.java`. Er kod, som hittar lösningen, bör skrivas i `LabSolver.java` filen. När det färdiga programmet körs bör utskriften vara ungefär:

```
Created a random labyrinth:
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                          |       |  |  |     |     |  |
+--+  +  +--+--+--+--+  +  +--+  +--+--+  +  +  +--+  +--+  +
|     |             |  |          |        |        |     |
+  +--+--+  +--+--+--+--+  +--+--+--+  +--+  +  +--+--+  +--+
|  |       |     |  |  |       |  |     |        |  |  |  |
+  +  +  +  +  +--+  +  +  +--+--+  +  +  +--+  +  +  +  +  +
|  |  |  |            |            |     |  |          |
+--+  +--+--+  +  +--+  +--+--+--+--+--+--+--+  +  +--+--+  +
|     |       |  |          |          |  |  |     |  |
+  +--+  +  +  +  +  +  +--+--+  +  +--+--+--+  +--+  +--+--+
|  |  |  |  |  |  |       |       |  |  |          |  |
+  +  +  +--+  +  +  +--+--+--+--+--+--+  +  +--+--+--+  +  +
|  |    |     |  |               |  |       |     |     |
+--+--+  +  +  +  +  +--+--+  +  +--+  +--+  +--+  +  +  +--+
|       |  |  |  |       |  |  |    |          |  |     |
+  +--+--+--+  +--+  +  +--+--+--+  +--+  +--+  +  +--+  +  +
|  |             |  |                    |  |  |     |  |
+--+  +  +  +  +--+--+--+--+  +--+--+  +  +  +  +--+--+--+--+
|     |  |  |             |       |  |  |  |             |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

```
Solution found:
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|XX XX XX                  |       |  |  |     |     |  |
+--+  +  +--+--+--+--+  +  +--+  +--+--+  +  +  +--+  +--+  +
|     |XX XX        |  |          |        |        |     |
+  +--+--+  +--+--+--+--+  +--+--+--+  +--+  +  +--+--+  +--+
|  |     XX|     |  |  |       |  |     |        |  |  |  |
+  +  +  +  +  +--+  +  +  +--+--+  +  +  +--+  +  +  +  +  +
|  |  |  |XX XX XX XX XX|             |     |  |          |
+--+  +--+--+  +  +--+  +--+--+--+--+--+--+--+  +  +--+--+  +
|     |       |  |XX XX        |          |  |  |     |  |
+  +--+  +  +  +  +  +  +--+--+  +  +--+--+--+  +--+  +--+--+
|  |  |  |  |  |  |XX|       |       |  |  |          |  |
+  +  +  +--+  +  +  +--+--+--+--+--+--+  +  +--+--+--+  +  +
|  |    |     |  |XX              |  |       |     |     |
+--+--+  +  +  +  +  +--+--+  +  +--+  +--+  +--+  +  +  +--+
|       |  |  |  |XX XX   |  |  |    |XX XX XX   |  |     |
+  +--+--+--+  +--+  +  +--+--+--+  +--+  +--+  +  +--+  +  +
|  |             |  |XX XX XX XX XX XX XX   |XX|  |     |  |
+--+  +  +  +  +--+--+--+--+  +--+--+  +  +  +  +--+--+--+--+
|     |  |  |             |       |  |  |  |XX XX XX XX XX|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Labyrint gränssnittet (`Labyrinth.java`)

`Labyrinth.java` definierar ett gränssnitt för labyrinter. En labyrint har koordinater i vanligt x-y koordinat system. Koordinaterna `(0,0)` är det övre vänstra hörnet. Koorinaterna för det nedre högra hörnet är `(lab.getWidth()-1, lab.getHeight()-1)`.

```
public interface Labyrinth {

    public enum Direction { RIGHT, LEFT, UP, DOWN }

    // den viktigaste funktionen

    public boolean canMove(Direction dir, int x, int y);

    // getters

    public int getWidth();
    public int getHeight();

    // man kan spara en boolean vid varje koordinat

    public void setMark(int x, int y, boolean b);
    public boolean getMark(int x, int y);

    // man kan skriva ut tillståndet till sträng

    public String toString();

}
```

Med `canMove` kan man fråga ett labyrinth-objekt om det är möjligt att ta ett steg i någon av riktiningarna `Labyrinth.Direction.RIGHT, Labyrinth.Direction.LEFT, Labyrinth.Direction.UP, Labyrinth.Direction.DOWN` från givna `x` och `y`. Man kan också spara en boolean vid varje koordinat i labyrinten med `getMark` och `setMark`.

Implementationen av labyrint gränssnittet (`Lab.java`)

Ni behöver inte förstå koden i `Lab.java`, men ni får gärna läsa den. Koden har en konstruktor som skapar en slumpmässig labyrint med den givna storleken. Labyrinten är skapad på ett sådant sätt att det endast finns ett sätt att komma från en plats i labyrinten till en annan plats.

Att hitta lösning till en labyrint (`LabSover.java`)

Er uppgift är att skriva koden för `findPath(x0,y0,x1,y1,l)` metoden i `LabSolver.java` (nedan). Idén med metoden `findPath` är att ändra på tillståndet i det givna labyrinth-objektet `l` så att `l.getMark(x,y)` returnerar sant ifall `(x,y)` är på vägen från `(x0,y0)` till `(x1,y1)`. När man skriver ut labyrint-objektet med `System.out.println(l)` då skriver den ut `XX` där `l.getMark(x,y)` är sant.

Hur ska man tänka när man löser denna uppgift? Svar: försök hitta en lösning rekursivt. Om `(x0,y0)` och `(x1,y1)` är samma koordinat, då är vi färdiga. Ifall `(x0,y0)` och `(x1,y1)` koordinaterna inte är samma, då markerar vi den nuvarande `(x0,y0)` koordinaten med `setMark` och går i någon av riktningarna som är möjliga, och försöker hitta en lösning från de nya koordinaterna. Exempel: ifall det är möjligt att ta ett steg till höger, då försöker vi hitta en lösning från koordinaten `(x0+1,y0)` till `(x1,y1)`. Om det inte lyckas, då måste vi gå tillbaka, dvs ta bort `setMark`.

Koden behöver inte vara lång. Min implementation av `findPath` är endast 25 rader lång (om man inte räknar kommentarer).

Tips. Innan ni skriver kod lönar det sig att fundera nogrannt hur ni tänker att koden skall fungera.

Obs. Det är osannolikt att er kod fungerar första gången den körs. Det lönar sig att sätta flera `System.out.println(l)`-utskrifter i koden så att ni kan se vad som händer och hur labyrinten ser ut.

Lös uppgiften med att ersätta kommentarerna "`// ...`" med er egen kod.

```
public class LabSolver {

    public static void main(String[] args) {
        int width = 20;
        int height = 10;
        if (args.length > 1) {
            width = Integer.parseInt(args[0]);
            height = Integer.parseInt(args[1]);
        }
        Labyrinth l = new Lab(width,height);
        System.out.println("\nCreated a random labyrinth:");
        System.out.println(l);
        boolean success = findPath(0,0,width-1,height-1,l);
        if (success) {
            System.out.println("Solution found:");
            System.out.println(l);
        } else {
            System.out.println("Failed to find a solution. (Bug in LabSolver.java)");
        }
    }

    public static boolean findPath(int x0, int y0, int x1, int y1, Labyrinth l) {
        // ...
        if ((x0 == x1) && (y0 == y1)) {
            return true;
        }
        // ...
        return false;
    }

}
```

När ovanstående kod körs, kan utskriften se ut som nedan.

```
$ java LabSolver

Created a labyrinth:
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|     |  |  |                       |              |     |
+  +--+  +  +  +--+  +--+--+  +--+--+--+  +--+--+--+--+  +--+
|              |  |        |     |              |  |  |  |  |
+  +--+  +  +  +  +  +--+  +  +  +  +--+--+--+--+  +  +  +  +
|     |  |  |     |     |  |  |        |        |  |  |     |
+  +--+  +  +--+--+--+--+  +--+--+  +  +  +--+--+  +  +  +--+
|  |     |             |        |  |           |  |  |     |
+  +--+  +--+  +  +--+  +  +--+--+--+  +  +--+--+  +  +  +--+
|  |        |  |  |     |           |  |        |  |  |  |  |
+--+--+  +--+  +  +--+--+  +--+--+--+--+--+--+  +  +  +  +
|        |     |     |               |  |  |              |
+--+--+  +--+  +  +--+--+  +--+  +--+--+  +  +  +--+  +--+--+
|        |     |     |     |  |                 |        |
+--+  +  +--+--+--+  +--+  +  +  +--+  +--+--+  +--+  +  +--+
|     |          |  |     |     |          |  |  |  |  |  |
+--+--+  +--+  +--+--+  +  +--+  +--+  +--+  +--+  +--+--+  +
|  |        |        |  |  |        |     |                 |
+  +--+  +--+  +  +--+  +--+--+  +--+  +--+--+  +  +  +--+  +
|           |  |     |        |  |           |  |  |     |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

Failed to find a solution. (Bug in LabSolver.java)
```

Testning (`LabSoverTest.java`)

Rättarna kommer att köra `java LabSoverTest` när de testar om er lösning fungerar. Se till att er lösning kommer igenom detta test.

Inlämning

Skicka in er *välkommenterade* version av `LabSolver.java` och en README-fil där ni beskriver hur ni kom fram till er lösning. Skicka in dessa filer som en zip-fil.

Frivilliga extra uppgifter:

1. Skapa en ny version av `Lab.java`, som konstruerar labyrinter där det kan finnas flera lösningar. Kalla den nya filen `LabExtra.java`. (***Tips:*** kopiera filen och skriv kod som tar bort några väggar efter att den fullständiga labyrinten har skapats.)

2. Skapa en ny version av `LabSolver.java`. Kalla den nya filen `LabSolverExtra.java`. Den nya bör hitta den kortaste lösningen. (***Tips:*** läs om Iterative deepening depth-first search på wikipedia, https://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search)

3. Implementera i en ny fil `LabSolverAll.java` kod som skriver ut alla lösningar till en given labyrinth från del 1. Sortera resultatet innan ni skriver ut det så att den kortaste lösningen är först och den längsta är sist.

Ni kan lämna in `LabExtra.java`, `LabSolverExtra.java` och `LabSolverAll.java` om ni vill ha kommentarer på dem också, men man får inga extra poäng för dessa frivilliga uppgifter.