Universität Stuttgart

FIUS

Übungsblatt Extra

Java Vorkurs(2024)

Hinweis: Alle Aufgaben auf diesem Blatt sind Optional und dementsprechend schwerer als die vorherigen Aufgaben.

Aufgabe 1 - Totoro im Labyrinth

Totoro hat sich in den Büschen verlaufen und will nach Hause zu seinem Baum. Er erinnert sich, dass man aus jedem Labyrinth entkommen kann, wenn man immer an der rechten Wand entlangläuft.

Implementiere die Operation `checkSideBush()` der Klasse `MySmartTotoro`. Sie soll `true` zurückgeben, wenn sich auf Totoros rechter Seite ein Busch befindet. Ansonsten soll sie `false` zurückgeben.

Implementiere die Operation `moveSmart()` der Klasse `MySmartTotoro`. In dieser soll Totoro zuerst überprüfen, ob sich zu seiner rechten ein Busch befindet. Ist dies nicht der Fall, soll er sich nach rechts drehen und einen Schritt in diese Richtung gehen. Anderenfalls soll er überprüfen, ob er geradeaus laufen kann. Ist dies der Fall, soll er einen Schritt nach vorne machen. Sollte sich sowohl rechts, als auch geradeaus ein Busch befinden, soll Totoro sich nach links drehen und stehen bleiben.

Die Klasse `OptionalLabyrinthTask` enthält ein Attribut vom Typ SmartTotoro. Weise diesem Attribut in der prepare-Operation ein neues MySmartTotoro-Objekt zu und platziere es auf Position *[0, 0]*. Vervollständige weiterhin die solve-Operation mithilfe der Operationen, die du gerade geschrieben hast, sodass Totoro seinen Baum erreicht und darauf stehen bleibt.

Aufgabe 2 - Neo im Labyrinth Teil 2

Nachdem Totoro aus dem Labyrinth entkommen ist, fragt er sich, ob er nicht schneller gewesen wäre, wenn er der linken Wand gefolgt wäre.

Implementiere die Operationen `checkSideBush()` und `moveSmart()` in einer neuen Klasse `MyLefthandedSmartToto` ↪ entsprechend. Wie musst du die `OptionalLabyrinthTask` Klasse verändern, damit Totoro an der linken Wand entlang läuft?

Aufgabe 3 - Rekusion

Rekursion

Ein Kommando welches sich selbst aufruft wird als rekusiv bezeichnet. Die Fibbonacci Reihe zum beispiel (1,1,2,3,5,8,13,21,...) kann beschrieben werden durch $F(0) = 1$ $F(1) = 1$ $F(n) = F(n-2)+F(n-1)$. wobei F(n) die n-te Fibbonacci Zahl ist. bei $F(n) = F(n-2)+F(n-1)$ wird erneut mit einem kleineren Argument verwendet. Dadurch wird das Argument in allen Schritten kleiner bis es bei $F(0)$ oder $F(1)$ ankommt welche einen Festen Wert haben.

a) versuche (rekursiv) ein Fibbonacci Kommando zu implementieren welches einen Integer n erhällt und dann $F(n)$ Nüsse ablegt.

b) lasse Totoro die ersten Sechs Elemente der Fibbonacci Folge nacheinander auf das Spielfeld legen.

c) (bonus) implementiere das Fibbonacci Kommando aus a) itterativ (ohne Rekursion).

Aufgabe 4 - Rekusion 2 - Füllen.

a) Implementiere das `moveTo(Position position)` Kommando in der Klasse totoro, welches totro zu der übergebenen Position laufen lässt. Um das Kommando zu testen, lasse totoro zu (2,4) laufen (Tipp: In dieser Aufgabe gibt es keine Büsche auf dem Spielfeld. Du musst daher keinen komplexen Wegfindungsalgorithmus implementieren.)

b) Totro ist umgeben von Nüssen. Wir wollen auch nun das innere dieser Form mit Nüssen füllen. Dafür verwenden wir einen beliebten (rekursiven) Füllalgorithmus der funktioniert wie folgt:
Fill(Position pos(x,y)):
-lege eine Nuss auf pos
-falls (x-1,y) leer ist: Fill(x-1,y)
-falls (x+1,y) leer ist: Fill(x+1,y)
-falls (x,y-1) leer ist: Fill(x,y-1)
-falls (x,y+1) leer ist: Fill(x,y+1)

implementieren diesen Füllalgorithmus, und lasse totoro die Form mit Nüssen füllen