

Aufgaben zu Java-Karel -Einstiegsübungen

1.  Öffne das Projekt Roboter_Karel_Start mit BlueJ.

Erzeuge jeweils eine Welt und ändere ihre Größe.

Anschließend erzeuge Roboter mit anderen Startpositionen und Blickrichtungen in dieser Welt.

Probiere wie unten angegeben die einzelnen Methoden der Klasse ROBOTER aus.

(→ Befehlsreferenz)

Welt für alle Aufgaben: Breite 8, Länge 10 , Höhe 5 → (8,10,5)

Ziel: Lasse Karol eine Strecke (Länge=5) laufen.

Ziel: Lasse Karol eine Treppe mit Höhe 3 bauen. Gerne auch in bunten Farben.

Ziel: Lasse Karol eine Reihe laufen, vier Ziegel in verschiedenen Farben ablegen und auf dem Rückweg wieder einsammeln.

Ziel: Lasse Karol auf den Rückweg einen Schritt laufen und dann prüfen, ob vor ihm ein Ziegel ist. Ist dies der Fall, so soll er ihn aufheben und eine Markierung setzen.

Ziel: Lasse Karol die Anzahl der Ziegel zählen und ausgeben.

Tipp: Zahl in Text: Integer.toString(zahl);


Ziel: Lasse Karol irgendwo in seiner Welt starten, dann soll er sich um 90° drehen. Er soll angeben in welche Richtung er schaut und ob vor ihm eine Wand ist.


Hinweis: Karol kann nur einmal eine Meldung ausgeben, deshalb musst du beide Infos in einem String zusammenfassen.

Zusatz-Ziel: Lasse Karol und Karoline solange entgegenlaufen bis sie sich treffen und sich wie in alten Filmen ihre Liebe gestehen.


Ergänze jetzt die Klasse ROBOTER an den von mir markierten Stellen um die nachfolgenden Methoden. Gehe immer davon aus, dass sich der Roboter an seiner Startposition befindet und keine sonstigen Hindernisse vorhanden sind.

2.  Erzeuge eine Reihe mit 4 bunten Ziegeln → viererReiheLegen().


3.  Erstelle eine Methode fuenferQuadrat() die ein Quadrat der Seitenlänge 5 mit Hilfe der Methode viererReiheLegen() erstellt.


4.  Bringe dem Roboter die Methoden schrittNachRechts() und entsprechend schrittNachLinks() bei.

Hinweis: Er soll dabei einen Schritt zur Seite gegangen sein und wieder in Laufrichtung schauen.

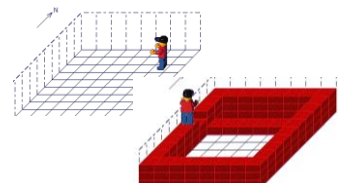
5.  Erstelle mit Hilfe der viererReiheLegen() eine Methode mauerBauen(), welche ihn eine Mauer der Breite 4 und der Höhe 3 bauen lässt.

Hinweis: Du musst die Methode springeHoch(zahl) in viererReiheLegen() einbauen.

6.  Erstelle eine Methode reiheZurWand(), die Karol von einer beliebigen Position (im Konstruktor ändern) aus eine Reihe Ziegel (Zusatz: Farbe mit Parameter eingeben) bis zur Wand legen lässt.


7.  Erstelle eine Methode hundertZiegel(), die unseren Roboter einhundert Ziegel von einer beliebigen Startposition aus am Rand seiner Welt entlang ablegen lässt.

Tipp: In der Zählschleife wird geprüft, ob er an einer Wand ist.




Wenn nicht legt er Ziegel ab. Wenn Ja, dann dreht er sich einmal nach Rechts und legt Ziegel ab.

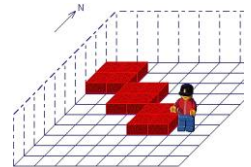
```
for (int i=0,i<100,i++){  
if(..){ } else { };  
}
```

8.  Karol soll aus jeder beliebigen Position zum Startpunkt (1,1) laufen und dort nach Süden schauen. Formuliere die Methode `zumStartpunktGehen()`.

Tipp: Karol speichert seine aktuelle Blickrichtung in der lokalen Variablen `r`.


Dann dreht er sich je nach Ergebnis, bis er nach Norden schaut und läuft bis er auf die Norwand trifft. Dann dreht er sich und läuft bis er wieder an der Wand auf (1,1) steht. Er dreht sich in Südrichtung.

9.  Erstelle eine möglichst effektive Methode `dreiQuadrate()`, die Karol zu (1,1) gehen lässt und das abgebildete Muster legen lässt.




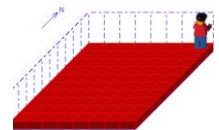
Tipp: Du malst das 1. Quadrat und erzeugst es dann mithilfe einer Zählschleife drei Mal. Verwende auch die bereits fertigen Methoden

`zumStartpunktGehen()`, `schrittnachLinks()` / `schrittnachRechts()`.


10.  Erstelle eine Methode `reiheSammeln()`, die alle Ziegel auf dem Weg zur Wand entfernt, zählt, wie viele aufgesammelt wurden und diesen Wert am Schluss am Bildschirm ausgibt. Teste deine Methode mit den Welten "reiheSammeln1.kdw" und "reiheSammeln2.kdw"

Anspruchsvollere Aufgaben für Schnelle

11.  Karol soll eine komplette Schicht Ziegel legen. Nutze die Methode `zumStartpunktGehen()`. Er legt an (1,1) den ersten Ziegel und beginnt dann von dort aus. Formuliere die Methode `fussbodenLegen()`.



Tipp: Wenn du zwei Bedingungen gleichzeitig prüfen willst geht das mit `&&` (UND) bzw. `||` (ODER)

12.  Erstelle (analog oben) eine Methode `allesSammeln()`, welche die ganze Ebene säubert und wieder am Bildschirm ausgibt, wie viele Ziegel gesammelt wurden.

Teste die Methode an zwei Beispielen: Mit der Welt "allesSammeln.kdw" und indem du zuerst `fussbodenLegen()` ausführst.

13 a.  Erstelle eine Methode `zufallsSchritt()`, die abhängig von einer zufällig gewürfelten 1 oder 0 den Roboter einen Schritt machen lässt oder nicht.


Verwende zum würfeln den Befehl `z=(int)(Math.random() * 10 + 1)` um zufällig eine Zahl zwischen 0 und 10 zu erzeugen. Ist die Zahl gerade bleibt bei der Modulo -Rechnung mit Zwei der Rest 0 $\rightarrow z \% 2 == 0$

gerade Zahl \rightarrow vor() oder ungerade Zahl \rightarrow stehen bleiben

13 b.  Erzeuge einen zweiten Roboter und erstelle eine Methode `wettRennen()`.

Lasse die beiden Roboter gegeneinander zu einem Wettrennen antreten und am Bildschirm ausgeben, wer das Rennen gewonnen hat.

Ob ein Roboter dabei einen Schritt macht oder nicht, soll durch die Methode `zufallsSchritt()` ermittelt werden.

13 c.  Verbessere dein Programm, indem du sie 10mal gegeneinander antreten lässt und mitzählst, wer wie oft gewonnen hat. Nach jedem Rennen müssen beide Roboter zurück auf ihre Startposition gehen.