Übungseinheit 6. - 10. November 2017

Was Sie lernen sollen:

- Anwendung und Wiederholung des bisher erlernten Stoffes.
- Zufallszahlen; Umgang mit der Klasse `Random`.

Aufgabe 1

Schreiben Sie ein Klasse `RandomBot` mit dem Konstruktor

```
public RandomBot(RobotSE robi)
```

der einen zu steuernden Roboter in ein Attribut speichert.

Die Methode

```
public void zufallsSchritt()
```

soll den Roboter einen Schritt in eine zufällig gewählte, nicht durch eine Mauer blockierte Himmelsrichtung machen lassen.

Schreiben Sie in einer eigenen Testklasse eine statische Methode, die einen geschlossenen `Raum` mit zwei Instanzen von `RandomBot` erzeugt und diese beiden Roboter abwechselnd so lange einen zufälligen Schritt machen lässt, bis sich die zwei Roboter an einer Kreuzung treffen!

Aufgabe 2

Schreiben Sie eine Klasse `Goldsucher` mit dem Konstruktor

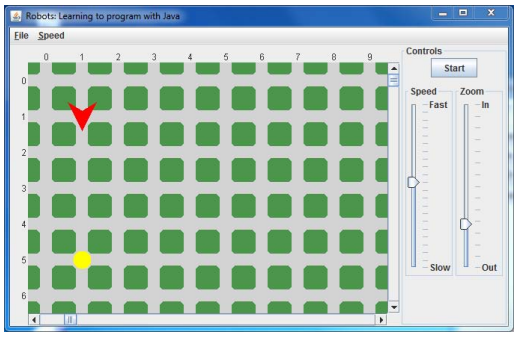
```
public Goldsucher(RobotSE robi)
```

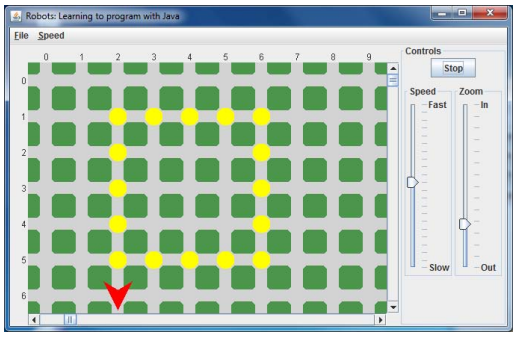
der einen zu steuernden Roboter in ein Attribut speichert.

Ein Goldsucher soll über eine Methode

```
public void steckeClaimAb()
```

verfügen, die den Goldsucher zum Pfostenlager gehen und die dort befindlichen Pfosten abzählen lässt (siehe linkes Bild). Sodann berechnet er die Seitenlänge des flächengrößten Quadrats, das er mit dieser Anzahl an Pfosten abstecken kann. (Verwenden Sie eine eigene Hilfsmethode zur Berechnung!). Abschließend bewegt er sich eine Spalte nach Osten und steckt das entsprechende Quadrat ab (siehe rechtes Bild).





Schreiben Sie eine Testklasse `TestGoldsucher` mit einer statischen Methode, die eine Stadt mit Goldsucher erzeugt und die Methode `steckeClaimAb()` testet. Der Roboter `robi` soll sich, nach Süden blickend, an der Position $(1, 1)$ befinden, an der Position $(5, 1)$ ist ein Pfostenlager anzulegen, an dem eine zufällige Anzahl von 0 bis 25 Dingen liegt (siehe linkes Bild).

Aufgabe 3

Schreiben Sie eine Klasse `FolgeSpurBot` mit dem Konstruktor

```
public FolgeSpurBot(RobotSE robi)
```

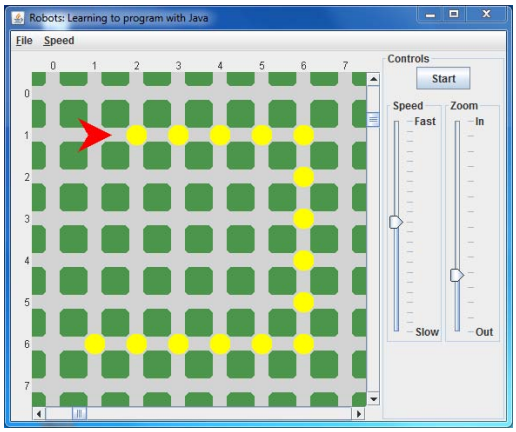
der einen Roboter in ein Attribut speichert.

Die Methode

```
public void folgeSpur()
```

soll den Roboter einer vor ihm beginnenden Spur von Dingen folgen und sämtliche Dinge aufheben lassen, bis die Spur endet. Dabei soll die Spur so sein, dass es immer genau eine Möglichkeit gibt, ihr zu folgen.

Schreiben Sie eine Testklasse `TestFolgeSpurBot` mit einer statischen Methode, die eine Stadt mit einer Spur erzeugt und die Methode `folgeSpur()` testet. Der Roboter soll nach Osten blicken und an den Koordinaten $(1, 1)$ stehen. Versuchen Sie, eine Spur wie im Bild dargestellt zufällig zu erzeugen, indem Sie dreimal hintereinander eine zufällige Anzahl von mindestens drei Dingen in eine passende zufällige Richtung auslegen.



Aufgabe 4

Schreiben Sie eine Klasse `Kreis` mit dem Konstruktor

```
public Kreis(double mittelpunktX, double mittelpunktY, double radius)
```

der einen Kreis mit Mittelpunkt und Radius speichert.

Die Methode

```
public boolean istImBereich(double x, double y)
```

soll `true` zurückgeben, falls der übergebenen Punkt (x, y) im Kreis liegt, anderenfalls `false`. Hierzu können Sie die Kreisformel $(x - x_{Kreis})^2 + (y - y_{Kreis})^2 \leq r^2$ verwenden.

Schreiben Sie eine Klasse `FlaechenSchaetzung`, die folgende statische Methode enthält:

```
public static double gibNaeherung(Kreis f, int anzahlVersuche),
```

die die Schnittfläche von Kreis und Quadrat mit den Eckpunkten $(0,0),(1,0),(1,1)$ und $(0,1)$ näherungsweise berechnet.

Die Methode `gibNaeherung` soll folgendes Verfahren implementieren:
In der Klasse `Random` liefert die Methode `nextDouble()` eine Zufallszahl (Gleichverteilung) aus dem Intervall $[0,1)$ zurück. Jedes Paar (x,y), wobei x und y für Werte stehen die man durch die Aufrufe von `nextDouble()` erhält, repräsentiert also einen Punkt im Quadrat mit den Eckpunkten $(0,0),(1,0),(1,1)$ und $(0,1)$. Wenn Sie viele Punkte (x,y) bestimmen, so liegen alle Punkte im beschriebenen Quadrat und einige im Kreis. Das Verhälnis der im Kreis liegenden Punkte zu allen erzeugten Punkte soll zurückgegeben werden. Über den Parameter `anzahlVersuche` steuern Sie, wie viele Punkte in diese Näherung einfließen.

Für den Kreis mit dem Mittelpunkt $(0.5,0.5)$ und Radius 0.5 ergibt die Näherung ungefähr $\pi/4$.

Aufgabe 5

Schreiben Sie eine Klasse `TestePrim`, die folgende statische Methode enthält:

```
public static boolean testeObPrim(int zahl)
```

soll testen, ob eine eingegebene `zahl`≥ 1 eine Primzahl ist. `zahl` ist keine Primzahl wenn sie durch mindestens einen der Werte w, $2 \leq w \leq \sqrt{zahl}$, teilbar ist. Ist `zahl` eine Primzahl, so soll `true` zurückgegeben werden, anderenfalls `false`.

Aufgabe 6

Schreiben Sie eine Klasse `EinfacheFunktionen`, die folgende statische Methode enthält:

```
public static double eHochX(double x, double epsilon)
```

zur näherungsweisen Berechnung von e^x. Die Funktion e^x kann durch die Reihe $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$ angenähert werden. In den dritten Übungen haben wir einfach diese Reihe bis zum neunten Glied berechnet. Die Qualität der Näherung war damit eher mangelhaft. Da ist es natürlich besser, wenn wir die Reihe so lange auswerten, bis ein Summand betragsmäßig kleiner als `epsilon` wird, wobei wir `epsilon` entsprechend klein wählen (z.B. `epsilon`$= 0.0000001$). Außerdem haben wir in den dritten Übungen die Summanden berechnet, indem wir zuerst $i!$ und x^i bestimmt haben. Das ist aber viel zu aufwändig! Wir sehen ja sofort, dass wir den $(i+1)$-ten Summanden s_{i+1} erhalten, wenn wir s_i mit dem Ausdruck $\frac{x}{(i+1)}$ multiplizieren. Implementieren Sie auch diese Vereinfachung in der Methode `eHochX(double x, double epsilon)`.