Softwareentwicklung 1 – Praktikum

Kontrollstrukturen und numerische Datentypen

Abgabe bis 6.12.2016, 23:00 Uhr

Ziele

- Verwenden von ein und zweidimensionalen Arrays
- Verstehen und Nachprogrammieren von Algorithmen

Hinweis

Sie finden für jede der Aufgaben ein vorbereitetes Greenfoot Szenario in meinem GitHub Projekt **karatest**. Für jedes Aufgabenblatt können Sie die jeweils aktuelle Version der Aufgaben in einem Archiv herunterladen:

https://github.com/uhafner/karatest/archive/assignment4.zip

Bitte nutzen Sie immer diese vorgegebenen Szenarien zur Lösung der Aufgaben. Verwenden Sie insbesondere immer die dort bereits enthaltenen Aufgabendateien für Ihre Lösung, nur so können wir Ihre Lösungen automatisiert auswerten. Ebenso dürfen Sie die Verzeichnisstruktur nicht umändern.

Erlaubte Elemente zur Umsetzung

Zur Lösung der Aufgaben dieses Blattes dürfen alle in der Vorlesung behandelten Java Sprachmittel sowie die 10 vorgestellten Kara Methoden und Sensoren (move(), turnRight(), turnLeft(), putLeaf(), removeLeaf(), isTreeFront(), isTreeLeft(), isTreeRight(), isMushroomFront(), isOnLeaf()) genutzt werden.

Achtung: Die Nutzung von Fields (Instanzvariablen) ist nicht erlaubt.

Beliebige Welten

Beachten Sie, dass in allen Aufgaben dieses Blattes beliebige Welten (gemäß der Aufgabenstellung) zulässig sind. D.h. eine Aufgabe ist nur teilweise gelöst, wenn Ihr Programm nur für die im PDF abgedruckte Variante funktioniert. Nutzen Sie deshalb zum Testen eigene Welten (und die im Szenario und in den Tests vorgegebenen Welten).

Aufgabe 12 (ca. 6 Punkte)

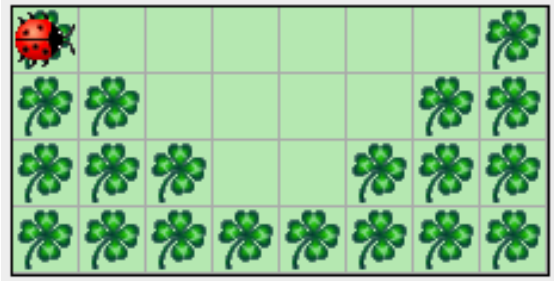
Schreiben Sie ein Programm, das Karas Welt um die angegebene Anzahl ***n*** an Spalten verschiebt (***n*** ist eine beliebig große Zahl). Die Welt von Kara ist immer 8x4 groß und enthält ein Muster aus Blättern. Bäume und Pilze sind nicht vorhanden. Kara startet dabei in der linken oberen Ecke mit Blick nach rechts. Nach dem Ende des Programms soll Kara wieder in der Ausgangsposition stehen. Die Welt soll dabei so verschoben werden, dass Spalten, die links bzw. rechts herausgeschoben werden, auf der gegenüberliegenden Seite wieder hineingeschoben werden. Bei positiven Zahlen wird die Welt nach links verschoben, bei negativen Zahlen nach rechts.

Die Anzahl der Spalten um die verschoben wird, können Sie mit folgendem Aufruf am Anfang Ihrer **act** Methode ermitteln:

```
public void act() {
    int shift = readInt("Zu verschiebende Spalten: ");

    ... Verschieben umsetzen ...
}
```

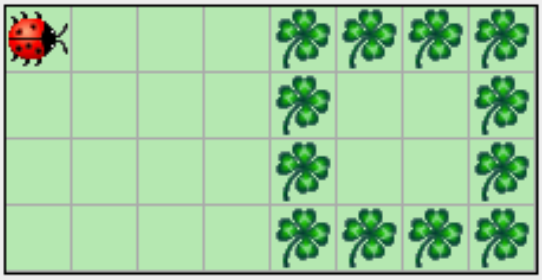
Start



Resultat (verschoben um 2, 10, -6, ...)

Start

Resultat (verschoben um 4, 12, -4, ...)



Tipp: Lesen Sie Karas Welt in ein zweidimensionales Array ein und schreiben Sie dann die Welt zurück, jeweils versetzt um den angegebenen Wert.

Aufgabe 13 (ca. 8 Punkte)

Schreiben Sie ein Programm, das die Anzahl aller Primzahlen bis zu einer vorgebbaren natürlichen Zahl ***n*** > 1 berechnet und ausgibt. Verwenden Sie zur Lösung der Aufgabe den Algorithmus „Sieb des Eratosthenes“, der hier in Kurzform beschrieben ist:

*Zuerst werden alle Zahlen 2, 3, 4, … n der Reihe nach aufgeschrieben. Alle Zahlen sind zunächst **potentielle** Primzahlen. Der Algorithmus nimmt sich nun jeweils die kleinste noch nicht gestrichene Zahl, und markiert diese als Primzahl. Im ersten Schritt ist dies die 2. Nachdem eine Primzahl gefunden wurde, werden alle Vielfachen dieser Primzahl gestrichen, denn diese sind sicher keine Primzahlen. D.h. im ersten Schritt werden die 4, 6, 8, etc. gestrichen. Danach wird wieder von vorne gestartet und die kleinste nächste nicht gestrichene Zahl als Primzahl markiert. Im zweiten Schritt ist das die 3. Gestrichen werden dann die Zahlen 9, 12, 15, etc. (Es genügt beim Streichen mit dem Quadrat der Primzahl zu beginnen, da alle kleineren Vielfachen bereits in Schritten davor gestrichen sein müssen.) Sobald das Quadrat der Primzahl größer als die Schranke n ist, sind alle Primzahlen kleiner oder gleich n bestimmt: Es sind die nicht gestrichenen Zahlen.*

Nutzen Sie das vorgegebene Greenfoot-Szenario zur Lösung dieser Aufgabe, auch wenn das nicht ganz passend ist, da wir Kara selbst nicht bewegen und die Welt auch sonst nicht verwenden. Damit die automatische Auswertung mit Maven funktioniert, müssen Sie die beiden Methoden **readInt** und **showMessage** verwenden. So können Sie diese beiden Methoden einsetzen:

```
public void act() {
    int max = readInt("Suche nach Primzahlen einschließlich bis: ");

    ... Sieb des Eratosthenes umsetzen ...

    showMessage("%d Primzahlen gefunden.", numberOfPrimes);
}
```

Aufgabe 14 (ca. 8 Punkte)

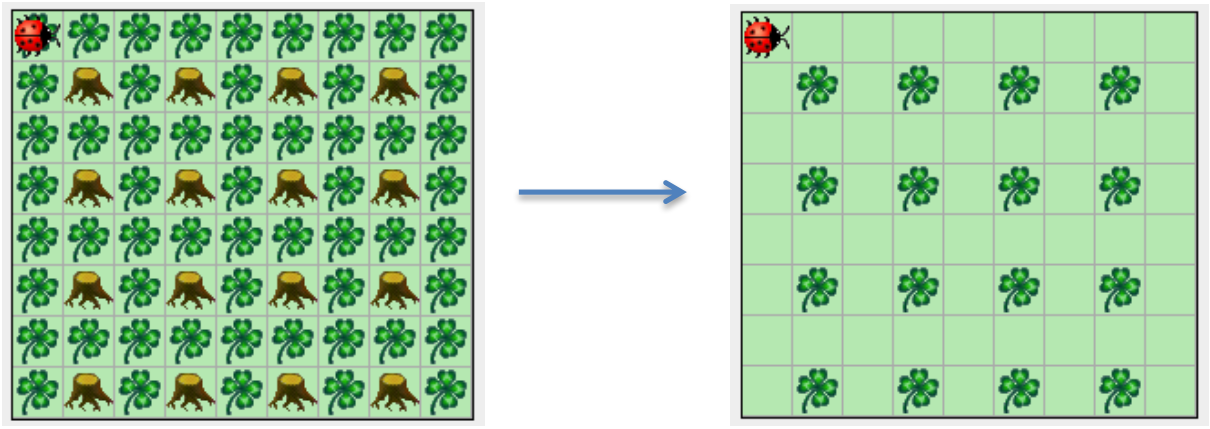
Ändern Sie Ihr Programm aus Aufgabe 6 so ab, das es die abgelaufene Welt umändert:

a) Jedes Blatt wird durch ein leeres Feld ersetzt

b) Jeder Baum wird durch ein Feld ersetzt

Haben Sie die Aufgabe 6 nicht gelöst, bzw. möchten Sie nicht mit Ihrer Lösung weiterarbeiten, dann können Sie auch die im Szenario enthaltenen Musterlösung als Ausgangsbasis nutzen.

Es gelten die gleichen Annahmen wie in Blatt 6: Die Welt von Kara ist beliebig groß und enthält ein Muster aus Blättern und Bäumen. Pilze und leere Felder sind nicht vorhanden. Kara startet in der linken oberen Ecke mit Blick nach rechts. Nach dem Ende des Programms soll Kara wieder in der Ausgangsposition stehen.



Tipp: Lesen Sie die Positionen der Bäume in ein zweidimensionales Array ein. Schreiben Sie anhand dieses Arrays dann die neue Welt. Um die Bäume zu entfernen, hat Kara die neue Methode **clear()** gelernt.