Programmieren 1 – Schmiedecke

Übung X3 – Etch-A-Sketch

Lernziele: Expressions, Statements, Funktionen, Kontrollstrukture, Umsetzen mathematischer Funktionen in Java-Ausdrücke.

Vorbereitung:

Lesen Sie die Aufgabe vor Beginn des Labors vollständig durch und beantworten Se die Vorbereitungsfragen!

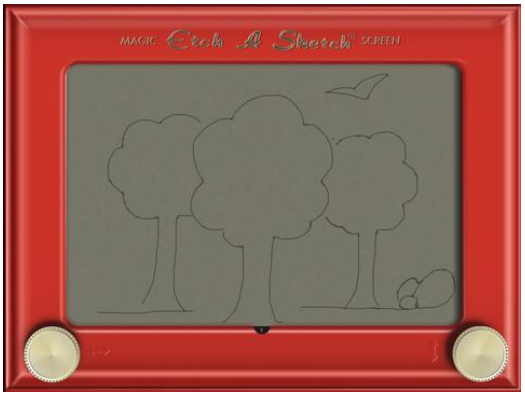
Nachbereitung: Laborprotokoll

Fügen Sie bitte diesmal immer ihre gesamte main-Methode direkt ins Protokoll ein (bei Screenshots auf Lesbarkeit achten!). **Und laden Sie bitte auch Ihr eclipse-Projekt hoch!**

Wichtig: Halten Sie auf jeden Fall fest, welche Verhalten Sie aufgrund Ihrer Ausdrücke *erwartet* haben und ob das *beobachtete* Verhalten davon abweicht (und möglichst: warum?). Es ist nicht peinlich, das Falsche erwartet zu haben!!!

Einführung: Der "Etch-A-Sketch®-Apparat"

"Etch-A-Sketch", ein beliebtes Kinderspielzeug, ist eine Zeichentafel, auf der man mit Hilfe eines Malpunktes zeichnen kann. Der Malpunkt wird durch zwei Drehknöpfe bewegt. Ein Knopf bewegt ihn in horizontaler Richtung, einer in vertikaler. Beide Knöpfe können gleichzeitig gedreht werden. Die Zeichnung entsteht, indem der Malpunkt eine Spur hinterlässt. Geschickte Zeichner zeichnen Schrägen, gleichmäßige Bögen, Kreise und ganze Bilder.... Inzwischen gibt es das Spielzeug auch als Retro-App, hier ein Screenshot der iPad-App:



In dieser Übung werden Sie einen **virtuellen Etch-A-Sketch-Apparat** bedienen. Anstelle der beiden unabhängigen Knöpfe sollen Sie den Malpunkt mithilfe einer Methode direkt zur nächsten x/y-Position lenken. Wo diese Position sein soll, errechnen Sie innerhalb der Methode, wobei Sie Konstanten, die aktuelle Position und die Grenzen des Malfelds verwenden können. Durch wiederholten Aufruf der Methode ergibt sich sich ein Linienzug.

Aufgrund einer eingebauten Verzögerung können sie sogar beobachten, wie die Linie im Laufe der Programmausführung wächst.

Der Zeichenapparat wird durch die Klasse NovoSketch realsiert. NovoSketch bietet Ihnen folgende Methoden an, mit denen Sie Ihre Zeichnung gestalten können:

```
public static void setTitle(String title); // Fenstertitel
public static void setStartingPoint(int x, int y);
public static void drawLineTo(int x, int y, boolean visible);
      // Linienzug unterbrechen durch unsichtbare Linie, visible=false
public static void drawInBlue();
public static void drawInRed();
public static void drawInGreen();
public static int getMaxX(); // Malfeldgrenze in X-Richtung
public static int getMaxY(); // Malfedlgrenze in Y-Richtung
```

Die Klasse ist nicht sehr kompliziert, Sie können den Quellcode weitestgehend verstehen, müssen es aber nicht, Sie können die Klasse auch einfach so verwenden. Wichtig ist zu wissen, dass ein ***zentriertes kartesisches Koordinatensystem*** verwendet wird, der Punkt (0,0) also in der Mitte der Malfläche liegt. D.h. die X-Werte gehen von –MaxX bis +MaxX.

Wie zeichnen Sie mit NovoSketch? Schreiben Sie eine Klasse mit einer main-Methode. Darin setzen Sie den Fenstertitel und den Anfangspunkt. Dann rufen Sie in einer Schleife drawLineTo(x, y, true), wobei es Ihre Aufgabe ist, für die Parameter x und y immer neue Werte zu errechnen.

Für jede Zeichnung schreiben Sie eine neue Klasse (im selben Eclipse-Projekt).

Beispiel: waagerechte gerade Linie vom Nullpunkt aus

```
public class HorizontalFromZero {
      static int x=0, y=0;

      public static void main(String[] args) {
            NovoSketch.setTitle("Horizontale Linie");
            NovoSketch.setStartingPoint(0, 0);
            while (true) {
                  NovoSketch.drawLineTo(x, y, true);
                  x++;
            }
      }
}
```

Vorbereitungsfragen:

1. Wie platziert man den Malpunkt in der rechten unteren Ecke?
2. Wie malt man eine vertikale Linie? Was, denken Sie, passiert am Rand der Malfläche?
3. Was passiert, wenn man neben x auch y jedesmal erhöht?

Laborexperimente

Jetzt sollen Sie mit dem virtuellen Etch-A-Sketch-Apparat programmatisch einige Zeichenaufgaben lösen. Denken Sie daran: Um zu erkennen, ob ein Programm sich "richtig" verhält, muss man zuerst theoretisch das richtige Verhalten ermitteln, also nachdenken ;). "Hacken" Sie die Programme also nicht einfach ein, sondern planen Sie sie vorher auf Papier und überzeugen Sie sich theoretisch davon, dass sie funktionieren (müssten). Wo Sie sie korrigieren müssen, tun Sie dies geplant - und schreiben Sie es im Laborbericht mit.

1. Statische Positionierung:

a) Schreiben Sie ein Programm, das den Malpunkt auf (10,20) setzt und nicht bewegt, und probieren Sie sie aus.

2. Linien

a) Zeichnen Sie eine waagerecht, eine senkrechte und eine diagonale Linie.

b) versuchen Sie, Ihre Linie über die ganze Malfläche zu ziehen.

c) Zeichnen Sie eine gestrichelte Diagonale.

d) Verändern Sie c) so, dass die Lücken kürzer sind als die Striche.

e) optional. Zeichnen Sie eine gestrichelte Diagonale mit wechselnden Farben.

3. Muster

a) Zeichnen Sie eine waagerechte Zickzacklinie.

b) Erzeugen Sie eine Linie im "Wickelmodus" (Wrap around), d.h. dass bei Erreichen eines Randes scheinbar auf der Rückseite der Malfläche weitergemalt wird, bis der nächste Rand erreicht wird und die Linie wieder sichtbar wird.

c) Implementieren Sie statt des Wraparound eine (Billard-artige) Reflexion am Rand des Malfensters. Wechseln sie bei jeder Reflexion die Strichfarbe.

d) optional: Malen Sie eine diagonale Schraffur über die gesamte Malfläche.

4. Kurven

a) Implementieren Sie eine einfache Beschleunigung in der y-Richtung, d.h. bei jedem Schritt in x-Richtung vergrößert sich der Schritt in y-Richtung. Schreiben Sie das Programm so, dass Sie die Anfangsgeschwindigkeit und die Beschleunigung leicht verändern können. Überlegen Sie unbedingt theoretisch, wie der Graph aussehen wird, ehe Sie es ausprobieren! (Falls Sie nichts sehen: Debugger nutzen!)

Bis hierher sollten Sie in der Übung mindestens kommen. Wenn Ihnen das nicht gelingt, ist das kein Beinbruch, aber sprechen Sie Ihren Dozenten darauf an!
Wenn möglich, versuchen Sie trotzdem aus dem bisher Geschafften eine einfache eigene Aufgabe zu finden, z.B. eine Schraffur mit ungleichen Abständen, und als 5.

einzubringen!

b) optional: Versuchen Sie, eine **ganze** Parabel zu zeichnen. (Formel nicht mehr klar? --> Googlen Sie sich dorthin!)

4. optional: Mathematische Experimente

a) optional: Implementieren Sie Funktions-Plotter für folgende Funktionen:

- $y = x^2$;
- y = sin(x);
- y = 1/x;

(Die nötigen Java-Funktionen finden Sie in der Bibliotheksklasse java.lang.Math. Dokumentation der Klasse: http://docs.oracle.com/javase/8/docs/api/ → Math)

b) optional: Zeichnen Sie einen Kreis! (Auch hier googlen Sie sich ggf. zur Formel... Englisch: Circle Equation)

5. Eigene Zeichnung – die "Kür"

Stellen Sie sich selbst eine Aufgabe, z.B.

- eine Spirale
- Ihre Initialen
- Das Haus vom Nikolaus
- Ein Stern
- Die Silhouette des Brandeburger Tors
- ...

Und nun fröhliches Programmieren!

Vergessen Sie nicht, Screenshots für Ihr Protokoll zu machen.....

Hinweis zum Copyright

Viele der Übungen zu diesem Kurs wurden ursprünglich von Prof. Lynn Andrea Stein für Ihren Kurs "IPIJ - Interactive Programming in Java" im Rahmen des Projekts"Rethinking CS101"

Diese Aufgabe wurde vollkommen verändert für den Kurs Programmieren 1 im WS 14/15 an der Beuth-Hochschule Berlin.

© der deutschen Version: Ilse Schmiedecke 2014- Fragen und Anregungen an schmiedecke@bht-berlin.de