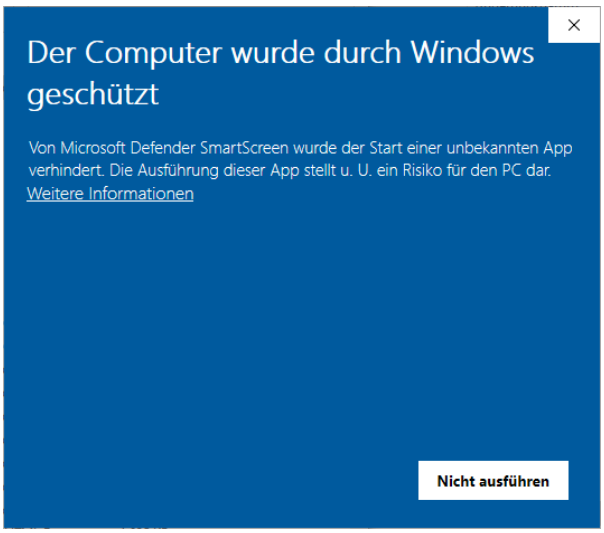
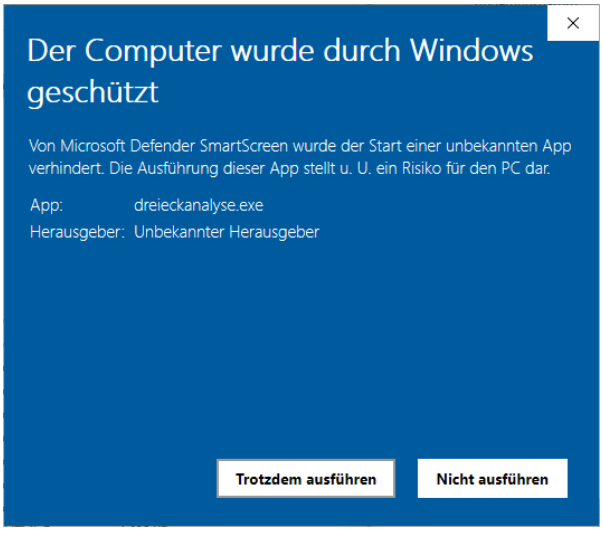
Fragen, Antworten, Kommentare





Hinweis: Bei den von der Veranstaltungsseite herunterladbaren exe-Programme mit Beispiellösungen kommt beim Starten mit einem Doppelklick typischerweise die links-oben gezeigte Meldung. Dies ist aus Sicherheitsgründen sinnvoll. Damit Sie das Programm trotzdem starten können, klicken Sie auf „Weitere Informationen“ und dann auf „Trotzdem ausführen“.

Frage: In 17 a) in der Klasse Punkt funktioniert meine folgende Methode nicht:

```
boolean istOk(int x, int y) {
    if (0 < this.x < 360 && 0 > this.y  > 440) {
      return true;
    } else {
      return false;
    }
}
```

Nicht Teil des Problems, aber deutlich wichtiger ist die Parameterliste, die keinen Sinn ergibt. Sie sollen einen Punkt überprüfen, d. h. es geht um das Punktobjekt selbst und seine Eigenschaften. Diese Eigenschaften stehen in this.x und this.y, also sind die Parameter hier überflüssig; genauer ist dies objektorientiert völlig falsch gedacht.

Das andere Problem ist, dass die mathematisch korrekte Schreibweise `0 < this.x < 360` so nicht in Programmen funktioniert, der Ausdruck muss einfach in zwei mit Und verknüpfte Ausdrücke zerlegt werden. [Als Randnotiz, man erhält einen Typfehler, da 0 < this.x erstmal nach true oder false ausgewertet wird und dann für diesen Booleschen Wert < 360 versucht wird auszuwerten.]

Nebenbei, die kürzere, aber erst später (ab 2. Semester) immer zu nutzende Schreibweise ist

```
public boolean istOk() {
    return this.x > 0 && this.x < 360 && this.y > 0 && this.y <440;
}
```

Kommentar: Da es vereinzelt Unklarheiten bei den Begriffen gab, hier nochmal eine Teilübersicht auf Basis der Aufgabe 14. Es wird davon ausgegangen, dass die Klasse Punkt korrekt existiert. Soll jetzt die Klasse Dreieck geschrieben werden, nutzt diese die Klasse Punkt. Zur Angabe eines Dreiecks reichen die Eckpunkte aus, so dass die Klasse wie folgt anfängt.

```
public class Dreieck {
    private Punkt ecke1; // Namen frei wählbar
    private Punkt ecke2;
    private Punkt ecke3;

    public Dreieck (Punkt ecke1, Punkt ecke2, Punkt a){
        this.ecke1 = ecke1;
        this.ecke2 = ecke2;
        this.ecke3 = a;  // andeuten, dass andere Namen moeglich sind
    }
```

Wichtig ist, dass hier keine einzelnen int-Variablen stehen. Objekte nutzten Objekte, so wird es vermieden viel Code mehrfach zu schreiben. Man erinnere sich, dass jedes Dreieck eigene Punkte hat. Dass diese Punkte zum Dreieck gehören, wird durch das `this` deutlich, mit dem auf Objektvariablen (und Objektmethoden) zugegriffen wird. Die drei Parameter werden zu lokale Variablen im Konstruktor, sie gehören nicht zum Objekt, ein Zugriff mit `this.a` ist nicht möglich. Natürlich dürfen die Parameter auch andere Namen haben.

Soll jetzt die Methode darstellen geschrieben werden, wird ein Objekt, genauer die Referenz auf ein Objekt, vom Typ Interaktionsbrett übergeben, deshalb ist die Parameterliste eindeutig. Man könnte auch in der Methode ein Interaktionsbrett erzeugen, dies wäre dann aber ein anderes, dass so nicht in verschiedenen Methoden genutzt werden kann. Da diese Methode nicht die Aufgabe hat ein Interaktionsbrett zu erzeugen, wäre der Ansatz hier also komplett falsch. Nebenbei sei bemerkt, dass mehrere Interaktionsbretter einfach übereinander angezeigt werden und es so nicht besonders auffällt (in der Fußzeile des Bildschirms schon), dass mehrere Objekte angelegt wurden.

Danach wird die passende Methode des Interaktionsbretts herausgesucht. Da es keine Methode für ein Dreieck gibt, müssen die drei Linien gemalt werden. Dazu gibt es eine Methode, die allerdings 4 int-Werte als Parameter hat. Diese int-Werte stecken in unseren Punkten in den Objektvariablen. Da diese get-Methoden haben, ist der Zugriff kein Problem.

Es gibt zwei Varianten, wie auf die einzelnen Punkte zugegriffen werden kann, diese sind beide nutzbar. Wenn man selbst die Methode schreibt, sollte man aber nur eine davon nutzen. Gegen diese Regel wird in der folgenden Methode bewusst verstoßen um beide Varianten zu zeigen.

```
public void darstellen(Interaktionsbrett ib){
    ib.neueLinie(this.ecke1.getX(), this.ecke1.getY()
                , this.ecke2.getX(), this.ecke2.getY());
    ib.neueLinie(this.ecke1.getX(), this.ecke1.getY()
                , this.ecke3.getX(), this.ecke3.getY());
    ib.neueLinie(this.ecke3.getX(), this.ecke3.getY()
                , this.getEcke2().getX(), this.getEcke2().getY());
}
```

Soll ein Punkt eingegeben werden, erfolgt dies in einer Methode, die nichts übergeben bekommt, also keine Parameter hat und einen Punkt zurückgibt. Zur Eingabe haben wir die Klasse EinUndAusgabe. Hier gibt es zwei Möglichkeiten so ein Objekt sinnvoll zu erhalten. Einmal kann eine lokale Variable in der Methode vom Typ EinUndAusgabe genutzt werden, der dann ein neues Objekt zugewiesen wird. Dies ist generell ok, etwas schöner ist der zweite Ansatz, dies Hilfsobjekt sich in einer Objektvariablen zu merken und das zugehörige Objekt z.B. im Konstruktor zu erzeugen.

Alternativ kann das Objekt direkt bei der Initialisierung zugewiesen werden, wie folgender Code zeigt. Die Eingabe sieht dann wie folgt aus.

```
public class GeoobjektEingabe{
  private EinUndAusgabe io = new EinUndAusgabe();

  Punkt punktEingeben() {
    this.io.ausgeben("X-Wert eingeben: ");
    int x = io.leseInteger();
    this.io.ausgeben("Y-Wert eingeben: ");
    int y = io.leseInteger();
    return new Punkt(x,y);
  }
```

Zu den Dingen, die schlichtweg völlig falsch sind, wäre es, wenn die Klasse GeoobjektEingabe eine Objektvariable vom Typ Punkt hätte. Dies wäre dann sinnvoll, wenn hier ein Punkt-Objekt verwaltet werden sollte, was nicht der Fall ist.

Für die Klasse Geoobjektausgabe gibt es in der ursprünglichen Aufgabenstellung einige akzeptable Varianten. Es werden Objekte der Klasse GeoobjektEingabe und Interaktionsbrett benötigt. Der flexibelste Ansatz ist, dass diese Objekte von außen übergeben und dann genutzt werden, was der immer wiederkehrende Ansatz der Objektorientierung ist. Dies war bei der Einstiegsaufgabe in das Thema aber nicht gefordert.

```
public class GeoobjektAusgabe{
  private Interaktionsbrett ib = new Interaktionsbrett();
  private GeoobjektEingabe geoin = new GeoobjektEingabe();

  public GeoobjektAusgabe(Interaktionsbrett ib
                        , Geoobjekteingabe geoin){
    this. ib = ib;
    this.geoin = geoin;
}
```

Im nächsten Schritt soll ein Punkt eingelesen und ausgegeben werden. Die einzig korrekte Methode dazu sieht wie folgt aus. Als Variante kann man die Anweisungskette natürlich in Teilschritte aufteilen.

```
  public void punktAusgeben(){
    this.geoin.punktEingeben().darstellen(this.ib);
    }
  }
```

Wiederum objektorientiert völlig falsch wäre ein Ansatz mit den folgenden Zeilen.

```
  public void punktAusgebenSchlecht(){
    Punkt tmp = this.geoin.punktEingeben();
    this.ib.neuerPunkt(tmp.getX(), tmp.getY()); // au weia
  }
```

Die Klasse Punkt hat die Aufgabe übernommen sich auf einem Interaktionsbrett zu zeichnen. Das ist sinnvoll, da fast alle für die Aktion wesentlichen Informationen, der x- und der y-Wert, in dieser Klasse vorliegen. Sollte dann ein Punkt dargestellt werden sollen, muss diese Methode genutzt werden.