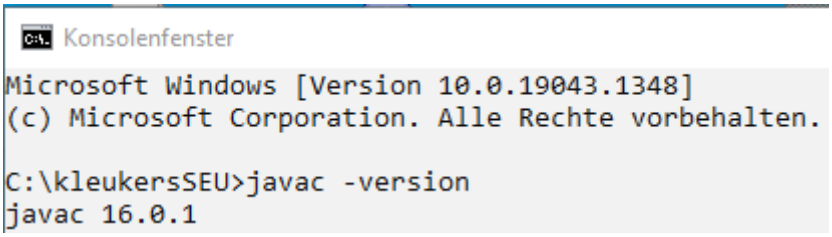
Fragen, Antworten, Kommentare zur aktuellen Vorlesung

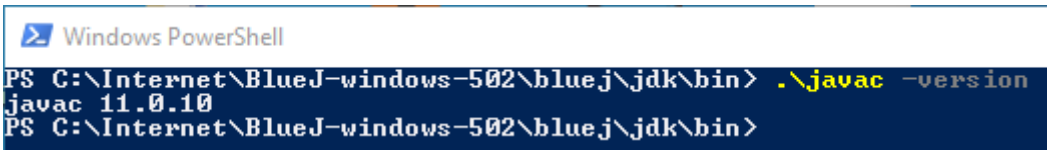
Hinweis: Die Diskussion einer Beispiellösung zur Aufgabe 17 mit der Einarbeitung der Überprüfung der graphischen Objekte finden Sie unter https://youtu.be/Aj0kpm7gUiQ (44:13).

Frage: Ich habe bei der Nutzung der Klasse EinUndAusgabe oder Interaktionsbrett eine Fehlermeldung, woran liegt das?

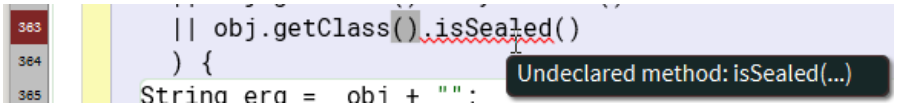
Dies ist bisher bei Personen aufgetreten, die nicht die vorgeschlagene KleukersSEU nutzen (ok, aber auf eigene Gefahr). Wenn Sie die Umgebung selbst zusammenbauen müssen Sie darauf achten möglichst die gleichen Versionen von BlueJ (5.0.3) und Java (18.01) wie ich zu nutzen. Auch neuere Versionen können theoretisch zu Fehlern führen, sollte aber hier nicht der Fall sein. Bei bisherigen Untersuchungen zeigte sich, dass eine ältere Version von Java genutzt wurde. Zur Überprüfung der genutzten Java-Version wird im jeweiligen Betriebssystem ein Terminal-Fernster geöffnet und folgender Befehl eingegeben.



Nutzen Sie das BlueJ, das zusammen mit einem JDK (Java Development Kit) heruntergeladen wird, nutzen Sie automatisch eine Java 11-Version.



Dies führt bei der Klasse EinUndAusgabe.java zu folgender Fehlermeldung. Ein kleiner schmuddeliger Workaround ist es, diese Zeile auszukommentieren.



Für echte Projekte gilt nebenbei, dass am Anfang eine Umgebung mit den kritischen Werkzeugen vorgegeben und diese nur nach intensiver Analyse in laufenden Projekten eventuell geändert wird.

Sollten weitere Fehler existieren, bitte bei mir, möglichst mit Beschreibung und Screenshot, melden.

Hinweis: Bei den Aufgaben zur Überprüfung der Geo-Objekte kann sehr gut die Kompaktschreibweise für Methoden mit Boolescher Rückgabe geübt werden. In der Klasse Punkt ist es für Erstsemester absolut in Ordnung, wenn die folgende Lösung gefunden wird.

```
public boolean istOk() {
  boolean ergebnis = true;
```

```
    if (this.x <= 0 || this.x >= 360) {
      ergebnis = false;
    }
    if (this.y <= 0 || this.y >= 440) {
      ergebnis = false;
    }
    return ergebnis;
  }
```

Generell ist der Ansatz am Anfang eine Hilfsvariable für das Ergebnis zu definieren gerade für Personen am Anfang der Programmierausbildung sehr sinnvoll. Im nächsten Schritt kann aber überlegt werden, dass keine besonderen Berechnungen mit der lokalen Variable `ergebnis` passieren, so dass diese weggelassen werden kann.

```
  public boolean istOk() {
    if (this.x <= 0 || this.x >= 360) {
      return false;
    }
    if (this.y <= 0 || this.y >= 440) {
      return false;
    }
    return true;
  }
```

Da beide if im positiven Fall zum gleichen Ergebnis führen, können diese einfach mit Oder verknüpft werden.

```
  public boolean istOk4() {
    if (this.x <= 0 || this.x >= 360 || this.y <= 0 || this.y >= 440) {
      return false;
    }
    return true;
  }
```

Bei return true und return false gerade in Kombination kann man sich überlegen, dass der Rückgabewert entweder genau der gennutzen Booleschen Bedingung entspricht, oder wie hier bei der Rückgabe von false der Negation davon.

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

Mit einer einfachen Booleschen Äquivalenzumformung kann die äußere Negation in diesem Fall vermieden werden, das finale Ergebnis sieht wie folgt aus.

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

Natürlich ist es auch ok, gleich auf das Ergebnis zu kommen.

Soll im nächsten Schritt überprüft werden, ob ein Kreis ok ist, ist die Prüfung vom Punkt zu nutzen, da sie dort schon implementiert wurde. Schlecht ist deshalb folgende funktionierende Lösung.

```
public boolean istOkNichtObjektorientiert() {
    return this.ausgangspunkt.istOk()
        && this.ausgangspunkt.getX() + 2 * this.radius < 360
        && this.ausgangspunkt.getY() + 2 * this.radius < 440;
}
```

Bei der Korrektur hilft eine kleine Überlegung, dass neben dem Ausgangspunkt, also der „linken oberen Ecke“ des Kreises nur noch die „rechte untere Ecke“ geprüft werden muss. Dazu wird ein Hilfspunkt konstruiert und geprüft.

```
public boolean istOk() {
```

```
    Punkt rechtsUnten = new Punkt(this.ausgangspunkt.getX() + 2 * this.radius
                                , this.ausgangspunkt.getY() + 2 * this.radius);
    return this.ausgangspunkt.istOk() && rechtsUnten.istOk();
}
```

Die obige Lösung wäre für die Praktikumsaufgabe ok, wenn bei der Eingabe des Kreises zuerst der eingegebene Punkt getrennt überprüft und bei nicht ok null für den Kreis zurückgegeben wird, da sonst ein Kreis mit einem Ausgangspunkt null das Ergebnis sein könnte, was schnell zu einer NullPointerException führen kann. Vollständig korrekt wird diese Prüfung, auch zusätzlich erst, wenn der Ausgangspunkt geprüft wird, da unklar ist, wer die Klassen und Methoden später nutzt.

```
public boolean istOkRichtig() {
    if(this.ausgangspunkt == null) {
      return false;
    }
    Punkt rechtsUnten = new Punkt(this.ausgangspunkt.getX() + 2 * this.radius
                                , this.ausgangspunkt.getY() + 2 * this.radius);
    return this.ausgangspunkt.istOk() && rechtsUnten.istOk();
}
```

Dies führt zu einer weiteren wichtigen Überlegung, wie Sie Ihr Programm testen. Natürlich ist zumindest einmal der positive Fall zu prüfen, erscheint ein ordentlicher Kreis am gewünschten Ort. Dann sind die Problemfälle einzeln zu prüfen, dabei reicht oft ein Problemfall nicht aus. Im Fall eines Kreises sind dabei folgende Fälle zu beachten:

- illegaler x- oder y-Wert beim Aufhängepunkt
- Radius negativ (fehlte in der Aufgabenstellung!)
- Radius zu groß in Richtung x-Achse
- Radius zu groß in Richtung y-Achse (Aufhängepunkt muss recht weit unten sein)