Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Programmierung 1

Wintersemester 2022/23
11. Lernnotiz

Hinweis: Diese Lernnotiz enthält einen sehr sinnvollen Vorschlag um den Lehrstoff der 11. Woche der Veranstaltung zu erlernen. Er ist gegliedert in die generellen Ziele und die Arbeitsschritte. Es ist notwendig, dass Sie die in dieser Lernnotiz genannten Videos bis zum Ende der offiziellen Vorlesungszeit (Mi 16:00 und Mo 11:30) durchgearbeitet haben. Zu den Vorlesungszeiten besteht die Möglichkeit in Zoom Fragen zu stellen und weitergehende Themen zu diskutieren.

https://hs-osnabrueck.zoom.us/my/kleuker

Diese Lernnotiz bezieht sich genauer auf die Vorlesungstermine am 14.12 und 19.12.
Denken Sie daran, dass ich auch über E-Mail erreichbar bin und Fragen beantworte.

Ziele VL 19

- Verständnis und Fähigkeit zur Nutzung der Klasse Object, mit der Möglichkeit vollständige equals().-Methoden beim Überschreiben zu erstellen
- Verständnis und Fähigkeit zur Nutzung von flachen und tiefen Kopien von Objekten mit Visualisierung in Objektspeicherdiagrammen

Arbeitsschritte VL 19

- *Laden Sie sich die folgenden Videos zuerst herunter, wenn Sie die HS-Plattform nutzen* und schauen Sie sich diese an. Es ist sinnvoll die Folien danach nochmals durchzugehen.

  Einführung in die Klasse Object mit BlueJ
  http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1Object1.mp4 (38:41), auch https://youtu.be/nPTZsbTk_3Q

  Folien 477 – 491: Idee der Klasse Object und Vervollständigung der equals()-Methoden
  http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1Object2.mp4 (24:31), auch https://youtu.be/TcQtYLj_OCI

  Folien 492 – 510: Tiefe und flache Kopien
  http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1Object3.mp4 (23:21), auch https://youtu.be/8GGCN8Iu_jA

  Hinweis: Die hier vorgestellte Idee, dass jede Klasse, die in einer tiefen Kopie genutzt werden kann, eine clone()-Methode oder einen Kopierkonstruktor zur Verfügung stellt, ist generell sinnvoll und wird oft genutzt. Diese Lösung funktioniert aber nicht, wenn es in den zu clonenden Objekten Referenzen auf identische (==) Objekte gibt, die auch in der Kopie identisch bleiben sollen. Dies veranschaulicht folgender Programmcode.

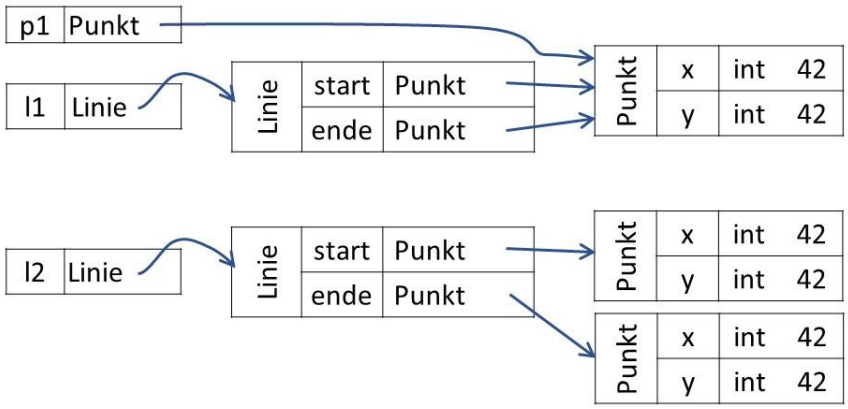
```
public class AnalyseTiefeKopie{

  public void linienspielerei(){
    EinUndAusgabe io = new EinUndAusgabe();
    Punkt p = new Punkt(42, 42);
    Linie l1 = new Linie(p, p);
    io.ausgeben("l1.start==l1.ende: "
          + (l1.getStart()==l1.getEnde()) + "\n");
    Linie l2 = l1.clone();
    io.ausgeben("l2.start==l2.ende: "
          + (l2.getStart()==l2.getEnde()) + "\n");
    io.ausgeben("l2.start eq l2.ende: "
          + (l2.getStart().equals(l2.getEnde())) + "\n");
  }
```

```
}
```

Innerhalb von l1 wird zweimal ein identisches Objekt referenziert, was für konkrete Linien natürlich kaum einen Sinn macht, aber z. B. in einer ArrayList durchaus sinnvoll sein kann. Wird dann l1 geclonet wird jeder einzelne Punkt der Linie geclonet. Das garantiert, dass in der gecloneten Linie start und ende gleich (equals) sind. Da aber jeweils neue Kopien entstehen, sind diese Punkte nicht identisch. Das zeigt auch die zugehörige Ausgabe.

```
l1.start==l1.ende: true
l2.start==l2.ende: false
l2.start eq l2.ende: true
```

Die Situation am Ende des Programmes zeigt auch folgendes Objektspeicherdiagramm.



Als Lösung kann man sich schon geclonete Objekte in einer bei der clone()-Methode mit zu übergebenden Hilfsobjekt merken und vor jedem Clonen prüfen, ob das Objekt schon geclonet wurde und ggfls. dieses als Ergebnis liefern. Diesen Ansatz können wir aktuell noch nicht umsetzen. Soll in Java ein echter Clone mit möglichen identischen Referenzen entstehen, gibt es verschiedene Lösungen. Ein Ansatz ist es die Möglichkeit von Java zu nutzen Objekte in Dateien zu schreiben (ObjectOutputWriter, XMLEncoder) und diese dann wieder zu lesen. Java garantiert dann, dass eine echte tiefe Kopie mit passenden Referenzen geliefert wird.
Diese Anmerkung ist bewusst als Hinweis markiert, da Sie das Problem verstehen sollten, sich die Lösung aber erst frühestens Ende des Semesters selbst erarbeiten können.

- Lesen Sie das zur Vorlesung gehörende Fragen-Und-Antworten-Dokument, das meist kurz nach der Vorlesung auf der Veranstaltungsseite in der Nähe dieser Lernnotiz steht.

Ziele VL 20

- Verständnis von und Fähigkeit zur Nutzung abstrakter Klassen als Grundgerüst und Vorgabe zur Entwicklung einer Vererbungshierarchie
- Verständnis von und Fähigkeit zur Nutzung und Umsetzung der dynamischen Polymorphie mit abstrakten Klassen

Arbeitsschritte VL 20

- *Laden Sie sich die folgenden Videos zuerst herunter, wenn Sie die HS-Plattform nutzen* und schauen Sie sich diese an. Es ist sinnvoll die Folien danach nochmals durchzugehen.

Einführung in abstrakte Klassen mit BlueJ
http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1AbstrakteKlassen1.mp4
(45:08), auch https://youtu.be/3kgnfxTlMZs

Folien 511 – 528: Schrittweise Einführung von abstrakten Klassen
http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1AbstrakteKlassen2.mp4
(34:21), auch https://youtu.be/uZHm_U3Atz4

Beispiel: Programmerweiterung durch Abstraktion und Verschiebung von Funktionalität
http://kleuker.iui.hs-osnabrueck.de/Videos/Prog1/Prog1AbstrakteKlassen3.mp4
(11:45), auch https://youtu.be/VVv9TyF2sTc

- Lesen Sie das zur Vorlesung gehörende Fragen-Und-Antworten-Dokument, das meist kurz nach der Vorlesung auf der Veranstaltungsseite in der Nähe dieser Lernnotiz steht.

- Bearbeiten Sie Aufgabenblatt 10. Denken Sie daran, dass ich für Fragen meist kurzfristig erreichbar bin.

- Prüfen Sie, ob Sie die angegebenen Ziele der Woche erreicht haben.