



Die Online-Befragung zur genutzten alternativen Veranstaltungsform und zur Lehrevaluation ist online. Bitte ausfüllen: <https://forms.gle/fZTBWpMUhK4ogLHw5>. Sie werden eventuell aufgefordert sich bei Google anzumelden, das ist nur notwendig, wenn Sie in der Bearbeitung eine Pause machen wollen und das Teilergebnis zwischenspeichern wollen. Die Befragung endet am 19.12., die Ergebnisse stehen in einem nachfolgenden Fragen&Antworten-Dokument auf der Webseite der Veranstaltung.

0.10. Aufgabe (1 Punkt)

Geben Sie die Lösungsworte der Quizze aus der Lernnotiz an.

33. Aufgabe (2 Punkte, Polymorphie-Verständnis, Achtung Klausurähnlichkeit, VL 18)

Gegeben seien die folgenden drei Klassen.

```
public class Y1{
    protected EinUndAusgabe io
        = new EinUndAusgabe();

    public void test(){
        io.ausgeben("1\n");
    }
}
```

```
public class Y2 extends Y1{
    @Override
    public void test(){
        io.ausgeben("2\n");
    }

    public void do1(){
        io.ausgeben("y2:do1 ");
        test();
    }

    public void do2(){
        io.ausgeben("y2:do2 ");
        super.test();
    }

    public void do3(){
        io.ausgeben("y2:do3 ");
        do4();
    }

    public void do4(){
        io.ausgeben("3\n");
    }
}
```

```
public class Y3 extends Y2{
    @Override
    public void do1(){
        io.ausgeben("y3:do1 ");
        test();
    }

    @Override
    public void do2(){
        io.ausgeben("y3:do2 ");
        super.do3();
    }

    @Override
    public void do3(){
        io.ausgeben("y3:do3 ");
        super.do2();
    }

    @Override
    public void do4(){
        io.ausgeben("y3:do4 ");
        do2();
    }
}
```

In einem anderen Programm werden folgende drei Objekte erzeugt.

```
Y1 x1 = new Y1();
Y2 x2 = new Y2();
Y3 x3 = new Y3();
```

Welche Ausgabe liefern die folgenden einzelnen Zeilen (falls sie keine Ausgabe angeben können, beschreiben Sie, was passiert)?



Welche Ausgabe liefern die folgenden einzelnen Zeilen (falls sie keine Ausgabe angeben können, beschreiben Sie, was passiert)?

- | | | |
|--------------|--------------|-------------|
| a) x1.test() | e) x2.do3() | i) x3.do2() |
| b) x2.test() | f) x2.do4() | j) x3.do3() |
| c) x2.do1() | g) x3.test() | k) x3.do4() |
| d) x2.do2() | h) x3.do1() | |

34. Aufgabe (4 Punkte, geschachtelte Schleifen, Fortsetzung letztes Aufgabenblatt, VL 15)

Nutzen Sie die Klasse Messreihe, die als Objektvariablen den Messort und eine Sammlung von ganzzahligen Messwerten enthält, vom vorherigen Aufgabenblatt. Es gelten die gleichen Randbedingungen. Auf der Veranstaltungsseite finden Sie ein Projekt mit Testklassen, stellen Sie am Ende sicher, dass alle Tests laufen.

Hinweise: Natürlich dürfen Sie weitere Methoden zum Strukturieren Ihres Programms oder bei Code-Wiederholungen schreiben. Wieder ist es sinnvoll, die Lösung erst auf Papier zu entwickeln.

- Schreiben Sie eine Methode paarSumme(), die ein neues Messreihen-Objekt zurückgibt, die die Summe der ersten beiden, des dritten und vierten, also jedes Wertepaares enthält. Ist die Anzahl der Werte ungerade, wird die letzte Zahl so übernommen, aus {4,3,2,4,3} soll z. B. {7,6,3} werden (4+3 = 7, 2+4 = 6, 3 alleine). Existiert die Liste nicht oder ist sie leer, enthält Ergebnis eine einelementige Liste mit dem Wert 0.
- Schreiben Sie eine Methode gesamtsumme(), die die Summe aller Messwerte zurück gibt und dabei zum Rechnen ausschließlich die Methode paarSumme() aus a) nutzt.
- Schreiben Sie eine Methode dreier(), die einen int-Wert übergeben bekommt und genau dann true liefert, wenn es drei Elemente an verschiedenen Positionen in der Liste gibt, die addiert diesen int-Wert ergeben. Z. B., für den Parameter-Wert 10 und die Liste {4,3,2,4,3} soll z. B. das Ergebnis true (4 + 2 + „andere 4“ = 10; auch 4 3 3 möglich) und die Liste {5,5,1,1,3} das Ergebnis false sein.
- Schreiben Sie eine Methode werteNachHinten(), die einen int-Wert w übergeben bekommt und alle Vorkommen von w nach hinten in die Messwerte schiebt, die restliche Reihenfolge bleibt unverändert. Z. B., für den Parameter-Wert 4 und die Liste {4,3,2,4,3} soll z. B. das Ergebnis die Liste {3,2,3,4,4}, für den Wert 2 die Liste {4,3,4,3,2} und für den Wert 5 die unveränderte Liste enthalten. (Erinnerung: Die Methode remove() darf nicht genutzt werden, set() schon.)

35. Aufgabe (2 Punkte, equals und clone, VL 19)

Von der Veranstaltungswebseite ist ein Projekt mit den Klassen Mitarbeit und Pairprogramming erhältlich, deren Anfang wie folgt aussieht.

```
public class Mitarbeit {
    private int minr;
    private String name;
}

public class Pairprogramming {
    private Mitarbeit mitglied1;
    private Mitarbeit mitglied2;
}
```

Die Klassen enthalten jeweils einen Konstruktor, get- und set-Methoden sowie leider falsche equals()- und clone()-Methoden. Korrigieren Sie diese Methoden mit den in der Veranstaltung vorgestellten Ansätzen und prüfen Sie ihre Ergebnisse mit den gegebenen Tests. Beachten Sie, dass alle Objektvariablen, deren Typ von Object erbt (also name, mitglied1, mitglied2), null-Werte enthalten dürfen. Zwei Pairprogramming-Objekte sollen nur gleich sein, wenn jeweils mitglied1 und mitglied2 übereinstimmen. Um bei Strings sicherzustellen, dass man eine Kopie mit gleichem Inhalt erhält, kann der Konstruktor genutzt werden, z. B. new String(this.name).



36. Aufgabe (9 Punkte, Entwicklung und Nutzung einer Vererbungshierarchie, VL 19)

Laden Sie das Ausgangsprojekt AufgabePartyverwaltung für diese Aufgabe von der Veranstaltungsseite, das Ihnen die Erstellung des Nutzungsdialoges abnimmt.

Implementiert werden soll die Abrechnung eines Party-Betriebs, die Grundlage der Abrechnungen sind besondere Produkte, die durch ihre Produktnummer, ihren Namen und ihren Preis beschrieben sind (Preise sollen in dieser Aufgabe als int-Werte, also Cent-Beträge modelliert werden, um die Rechengenauigkeit zu garantieren). Ein Produkt kann z. B. „Nr. 1 kitschiger Eisschwan 25000“ sein.

- a) Schreiben Sie eine Klasse Produkt mit den erwähnten Objektvariablen, sinnvollem Konstruktor, einer get-Methode getPreis() für den Preis und einer toString()-Methode für eine ordentliche Ausgabe.

Weiterhin bietet der Betrieb Materialien an, die für Partys gemietet werden. Die Miete, also der Preis für den Kunden, richtet sich nach der Anzahl der gemieteten Materialien. Die Miete für einen Stuhl beträgt z. B. 300, so dass später auf der Rechnung dann „Nr.42 50*Stuhl 15000“, für 50 gemietete Stühle stehen kann.

- b) Schreiben Sie eine Klasse Material, die von der Klasse Produkt erbt, einen sinnvollen Konstruktor hat und die die Anzahl als weitere Objektvariable enthält. Weiterhin müssen für die Berechnung die Methoden getPreis() und toString() überschrieben werden.

Weiterhin bietet der Betrieb Dienstleistungen für die Partys an. Dabei bestimmt die Anzahl der Personen und die Dauer ihrer Anwesenheit den Preis. Der Stundenlohn soll mit 2500 festgesetzt sein. Auf der späteren Rechnung kann dann z. B. „Nr. 43 2 * 6h Tuersteher 30000“, für zwei Türsteher, die sechs Stunden arbeiten sollen, stehen.

- c) Schreiben Sie eine Klasse Dienstleistung, die von der Klasse Produkt oder von Material erbt, einen sinnvollen Konstruktor hat und die die Personenanzahl und Stundenanzahl verwalten kann. Weiterhin müssen für die Berechnung die Methoden getPreis() und toString() überschrieben werden.

- d) Realisieren Sie den in der Klasse Partyverwaltung beschriebenen Nutzungsdialog mit *genau einer Sammlung von Produkten*, der die Eingabe unterschiedlicher Produkte, das Löschen eines eingegebenen Produkts, eine Gesamtübersicht mit Gesamtpreis und eine Zählung der unterschiedlichen Produktarten ermöglicht.

Achten Sie darauf, möglichst keine Programmzeilen, z. B. zur Eingabe, doppelt zu programmieren und Methoden zur Eingabe von Methoden zur Berechnung zu trennen. Bei der Ausgabe der Produktarten, sollen echte Produkte, Material und Dienstleistungen gezählt werden, dabei sollen Sie ohne if, ohne instanceof, ohne casten und ohne getClass(), also vollständig mit dynamischer Polymorphie auskommen [ist hier mit „Kanonen auf Spatzen schießen“, soll aber eingeübt werden]. Es kann sinnvoll sein, z. B. get und set-Methoden, weitere Methoden sowie parameterlose Konstruktoren zu ergänzen. Ein Nutzungsdialog kann wie folgt aussehen, Eingaben sind umrandet.

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Produktnummer:
Produktname:
Preis:

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Produktnummer:
Produktname:
Preis:



Programmierung 1

Wintersemester 2025/26

Aufgabenblatt 10

Anzahl:

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Produktnummer:

Produktname:

Anzahl:

Stunden:

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Nr. 1 kitschiger Eisschwan 25000

Nr. 42 50 * Stuhl 15000

Nr. 43 2 * 6h Tuersteher 30000

Gesamtsumme: 70000

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Produkte: 1

Material: 1

Dienstleistungen: 1

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen

- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Nr. 1 kitschiger Eisschwan 25000

Nr. 42 50 * Stuhl 15000

Nr. 43 2 * 6h Tuersteher 30000

Produktnummer:

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Nr. 1 kitschiger Eisschwan 25000

Nr. 43 2 * 6h Tuersteher 30000

Gesamtsumme: 55000

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

Produkte: 1

Material: 0

Dienstleistungen: 1

- (0) Dialog beenden
- (1) echtes Produkt hinzufuegen
- (2) Ausleihmaterial hinzufuegen
- (3) Dienstleistung hinzufuegen
- (4) Rechnungsposten loeschen
- (5) Gesamtuebersicht mit Preis
- (6) Anzahl von Produktarten

- e) Freiwillig: Sichern Sie das Programm gegen unsinnige Eingaben, wie negative Werte und doppelte Produktnummern ab, bzw. überlegen Sie, wie man sinnvoll z. B. zwei kitschige Eisschwäne kaufen (nicht mieten) kann.