Übung 4

Diagramme auf extra Blättern!

Aufgabe 13

b) Man könnte die Eigenschaft „Projekt-“ oder „Lehrmitarbeiter“ (oder auch andere Erb-Beziehungen) durch Attribute (boolean) ausdrücken. Dadurch hätte man nur eine Klasse Angestellter, die zwar dadurch einfacher zu modellieren ist, aber später deutlich komplexer bei der Wartung.

Aufgabe 15

b) Der Zusatz <T> bei Container-Klassen teilt dem Container mit, welche Klassenobjekte man in ihm speichern will. Dadurch spart man sich Typecasts bei jedem Zugriff auf Objekte des Containers, da der Compiler schon weiß, um welche Objekte es sich handelt.

* 1. A15 (main)

import java.util.Date;

import java.util.Iterator;

class A15 {

public static void main(String[] args) {

HighscoreContainer cont = HighscoreContainer.instance();

Date d1 = new Date();

Highscore h1 = new Highscore("Peter", 15, d1);

d1 = new Date();

Highscore h2 = new Highscore("Hans", 20, d1);

d1 = new Date();

Highscore h3 = new Highscore("Max", 13, d1);

cont.addHighscore(h1);

cont.addHighscore(h2);

cont.addHighscore(h3);

Iterator<Highscore> it = cont.iterator();

while (it.hasNext()) {

System.out.println(it.next());

}

// oder, hier zum Testen mit Ausgabe der Score:

for (Highscore h: cont) {

System.out.println(h.getScore());

}

System.out.println("Hoechste Highscore: " + cont.getHighscoreAt(0).getScore());

cont.removeHighscoreAt(0);

it = cont.iterator();

System.out.println("Hoechste Highscore nach Loeschvorgang: " + it.next().getScore());

cont.removeAllHighscore();

cont.getHighscoreAt(1);

}

}

* 1. Highscore

import java.util.Date;

class Highscore {

private String name;

private int score;

private Date datum;

Highscore(String name, int score, Date datum) {

this.name = name;

this.score = score;

this.datum = datum;

}

public String getName() {

return name;

}

public int getScore() {

return score;

}

public Date getDate() {

return datum;

}

}

* 1. HighscoreContainer

import java.util.Iterator;

import java.util.Vector;

public class HighscoreContainer implements Iterable<Highscore> {

private Vector<Highscore> highscore\_liste;

private static HighscoreContainer unique;

private HighscoreContainer() {

highscore\_liste = new Vector<Highscore>();

}

public static HighscoreContainer instance() {

if (unique == null)

unique = new HighscoreContainer();

return unique;

}

/\* Fuegt den neuen Highscore so in den Vektor ein, dass die Highscores der Groesse nach

\* geordnet sind, angefangen mit dem Hoechsten \*/

public void addHighscore(Highscore h) {

int size = highscore\_liste.size();

boolean found = false;

if (size == 0)

highscore\_liste.add(0, h);

else {

for (int i = 0; i < size; i++) {

if (highscore\_liste.elementAt(i).getScore() < h.getScore()) {

highscore\_liste.add(i, h);

found = true;

break;

}

}

if (!found)

highscore\_liste.add(size, h);

}

}

public void removeAllHighscore() {

highscore\_liste.removeAll(highscore\_liste);

}

public Highscore getHighscoreAt(int i) {

try {

return highscore\_liste.elementAt(i);

}

catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Index nicht vorhanden!");

return null;

}

}

public void removeHighscoreAt(int i) {

if (i >= 0 && highscore\_liste.size() > i)

highscore\_liste.removeElementAt(i);

else

System.out.println("Element nicht vorhanden!");

}

public Iterator<Highscore> iterator() {

return highscore\_liste.iterator();

}

}

Aufgabe 16

1. brauchtPalette als private, da Zugriff in dieser Aufgabenstellung nur durch klasseninterne „toString“-Methode erforderlich ist.  
   toString als public, da von außerhalb der Klasse zugegriffen werden muss.  
   Attribute von „Paket“ und „checkPaketdaten“ müssen auf „public“ gesetzt werden, da geerbte Klassen keinen Zugriff auf private Methoden und Attribute der Oberklasse haben.
   1. Hier die Attribute und Methoden als private, vollständiger Code dann bei c)

…..

private int laenge;

private int breite;

private int hoehe;

private int gewicht;

public Paket(int laenge, int breite, int hoehe, int gewicht) {

if (this.checkPaketdaten(laenge, breite, hoehe, gewicht)) {

this.laenge = laenge;

this.breite = breite;

this.hoehe = hoehe;

this.gewicht = gewicht;

} else

System.out.println("Fehler - Paketattribute entsprechen nicht den Vorgaben!");

}

private boolean checkPaketdaten(int laenge, int breite, int hoehe, int gewicht) {

return (laenge > 0 && breite > 0 && hoehe > 0 && gewicht > 0);

}

….

* 1. Vollständiger Code mit Vorgaben von d)
  2. Main

class Main {

public static void main(String[] args) {

Paket p1 = new Paket(15, 58, 13, 24);

System.out.println(p1);

Paket p2 = new Paket(17, 20, 13, 18);

System.out.println(p2);

if (p2.equals(p1))

System.out.println("Gleiche Paketklasse!");

else

System.out.println("Unterschiedliche Paketklasse!");

Sperrgut s1 = new Sperrgut(2,2,2,3);

Sperrgut s2 = new Sperrgut(20,20,30,5);

System.out.println(s1);

System.out.println(s2);

}

}

* 1. Paket

class Paket {

public int laenge;

public int breite;

public int hoehe;

public int gewicht;

public Paket(int laenge, int breite, int hoehe, int gewicht) {

if (this.checkPaketdaten(laenge, breite, hoehe, gewicht)) {

this.laenge = laenge;

this.breite = breite;

this.hoehe = hoehe;

this.gewicht = gewicht;

} else

System.out.println("Fehler - Paketattribute entsprechen nicht den Vorgaben!");

}

public boolean checkPaketdaten(int laenge, int breite, int hoehe, int gewicht) {

return (laenge > 0 && breite > 0 && hoehe > 0 && gewicht > 0);

}

private int calcPaketklasse() {

if (gewicht >= 25)

return 0;

int temp = Math.min(laenge, breite);

int min = Math.min(temp, hoehe);

temp = Math.max(laenge, breite);

int max = Math.max(temp, hoehe);

int sum = min + max;

if (sum < 50)

return 1;

else if (sum < 100)

return 2;

else if (sum < 150)

return 3;

else

return 4;

}

public boolean equals(Object o) {

return ((o instanceof Paket) && (((Paket) o).calcPaketklasse() == this.calcPaketklasse()));

}

public String toString() {

int klasse = calcPaketklasse();

if (klasse == 0)

return "Sperrgut: Gewicht zu hoch";

else if (klasse == 4)

return "Sperrgut: Paket zu gross";

else

return "Paketklasse: " + klasse;

}

}

* 1. Sperrgut

class Sperrgut extends Paket {

private boolean brauchtPalette = true;

public Sperrgut(int laenge, int breite, int hoehe, int gewicht) {

super(laenge, breite, hoehe, gewicht);

}

public String toString() {

if (this.checkPaketdaten(laenge, breite, hoehe, gewicht) && (brauchtPalette == true))

return ("Sperrgut (Laenge, Breite, Hoehe, Gewicht): " +

laenge + ", " + breite + ", " + hoehe + ", " + gewicht + " auf Palette");

else

return "Fehler - Paketattribute entsprechen nicht den Vorgaben!";

}

}