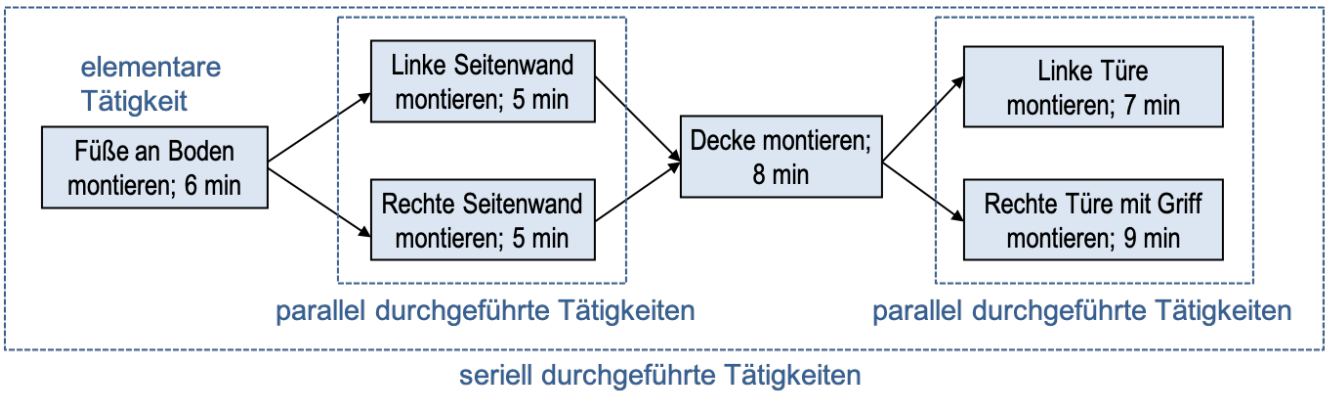
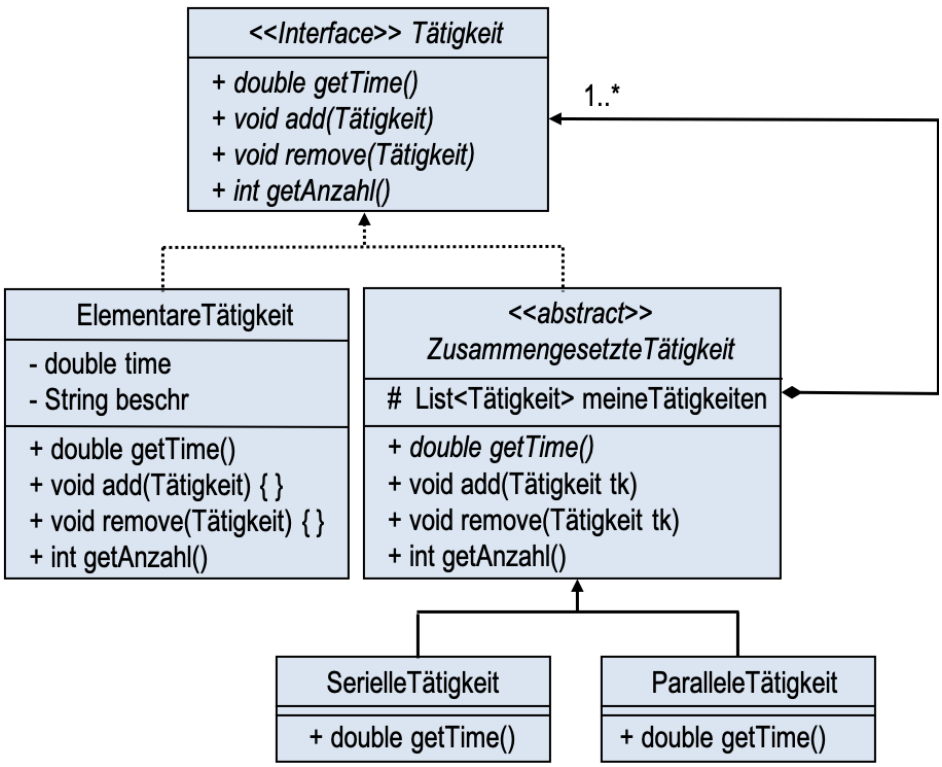
Aufgabenblatt 6

Teil 1: Entwurfsmuster Kompositum

In der Vorlesung haben Sie das Kompositum-Muster kennengelernt (Foliensatz 15, Entwurfsmuster). In dieser Aufgabe sollen Sie dieses Muster zur Modellierung von Tätigkeiten verwenden, die entweder elementar und mit einer benötigten Zeit versehen sind, oder die ist aus mehreren Tätigkeiten zusammengesetzt sind. Das Zusammensetzen von Tätigkeiten kann parallel oder seriell (hintereinander) erfolgen. Das Beispiel unten zeigt eine Schrankmontage.

Wir interessieren uns für die Zeit, die wenigstens benötigt wird, um alle Tätigkeiten abzuarbeiten. Bei parallel durchgeführten Tätigkeiten wird das Maximum und bei seriell durchgeführten Tätigkeiten wird die Summe der einzelnen Zeiten berechnet. Im Beispiel wird für die Schrankmontage wenigstens 28 min benötigt. Rechnen Sie bitte nach!





Eine Tätigkeit ist also entweder eine elementare Tätigkeit oder eine zusammengesetzte Tätigkeit, die seriell oder parallel sein kann. Eine zusammengesetzte Tätigkeit wird mit Hilfe von `add` bzw. `remove` zusammengebaut bzw. abgebaut. Mit `getAnzahl` lässt sich die Anzahl der elementaren Tätigkeiten bestimmen. Die Methode `getTime` liefert die benötigte Zeit zurück. Sehen Sie auch eine geeignete `toString`-Methode vor.

Die oben abgebildete Schrankmontage soll sich damit wie folgt umsetzen setzen. Dieser Testcode ist in der Klasse `TaetigkeitenTest` bereits vorgegeben.

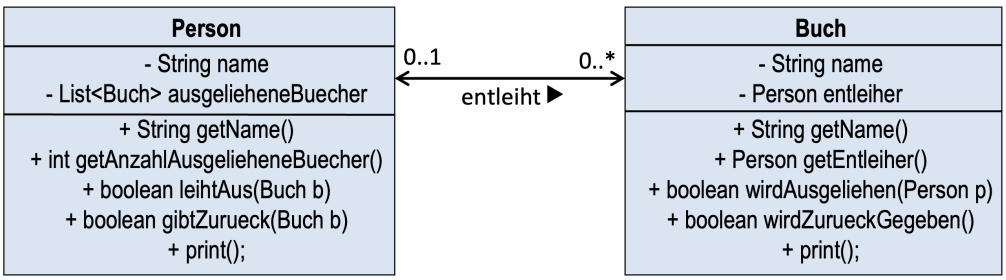
```
Tätigkeit tk1 = new ParalleleTätigkeit();
tk1.add(new ElementareTätigkeit("Linke Seitenwand montieren", 5.0));
tk1.add(new ElementareTätigkeit("Rechte Seitenwand montieren", 5.0));

Tätigkeit tk2 = new ParalleleTätigkeit();
tk2.add(new ElementareTätigkeit("Linke Türe montieren", 7.0));
tk2.add(new ElementareTätigkeit("Rechte Türe mit Griff montieren", 9.0));

Tätigkeit schrankMontage = new SerielleTätigkeit();
schrankMontage.add(new ElementareTätigkeit("Füße an Boden montieren", 6.0));
schrankMontage.add(tk1);
schrankMontage.add(new ElementareTätigkeit("Decke montieren", 8.0));
schrankMontage.add(tk2);
System.out.println(schrankMontage.getTime() + " min");   // 28.0 min
System.out.println(schrankMontage.getAnzahl());   // 6
System.out.println(schrankMontage);   // Test von toString
```

Teil 2: Bidirektionale 1-n-Assoziation

Für ein Bibliotheksverwaltungsprogramm ist folgende bidirektionale 1-n-Assoziation zu realisieren:



Implementieren Sie die beiden Klassen `Person` und `Buch`. Die booleschen Rückgabewerte geben an, ob das Entleihen bzw. die Rückgabe eines Buches erfolgreich war. Die `print`-Methode der Klasse `Person` gibt den Namen der Person und die Namen ihrer entliehenen Bücher aus. Die `print`-Methode der Klasse `Buch` gibt den Namen des Buchs und den Entleiher aus.

Zu Testzwecken finden Sie auf der Web-Seite die Klasse `Bibliothek`.

Hinweis: Schauen Sie sich auf dem Foliensatz 15, Entwurfsmuster nochmals den Abschnitt über bidirektionale 1-n-Assoziationen an.