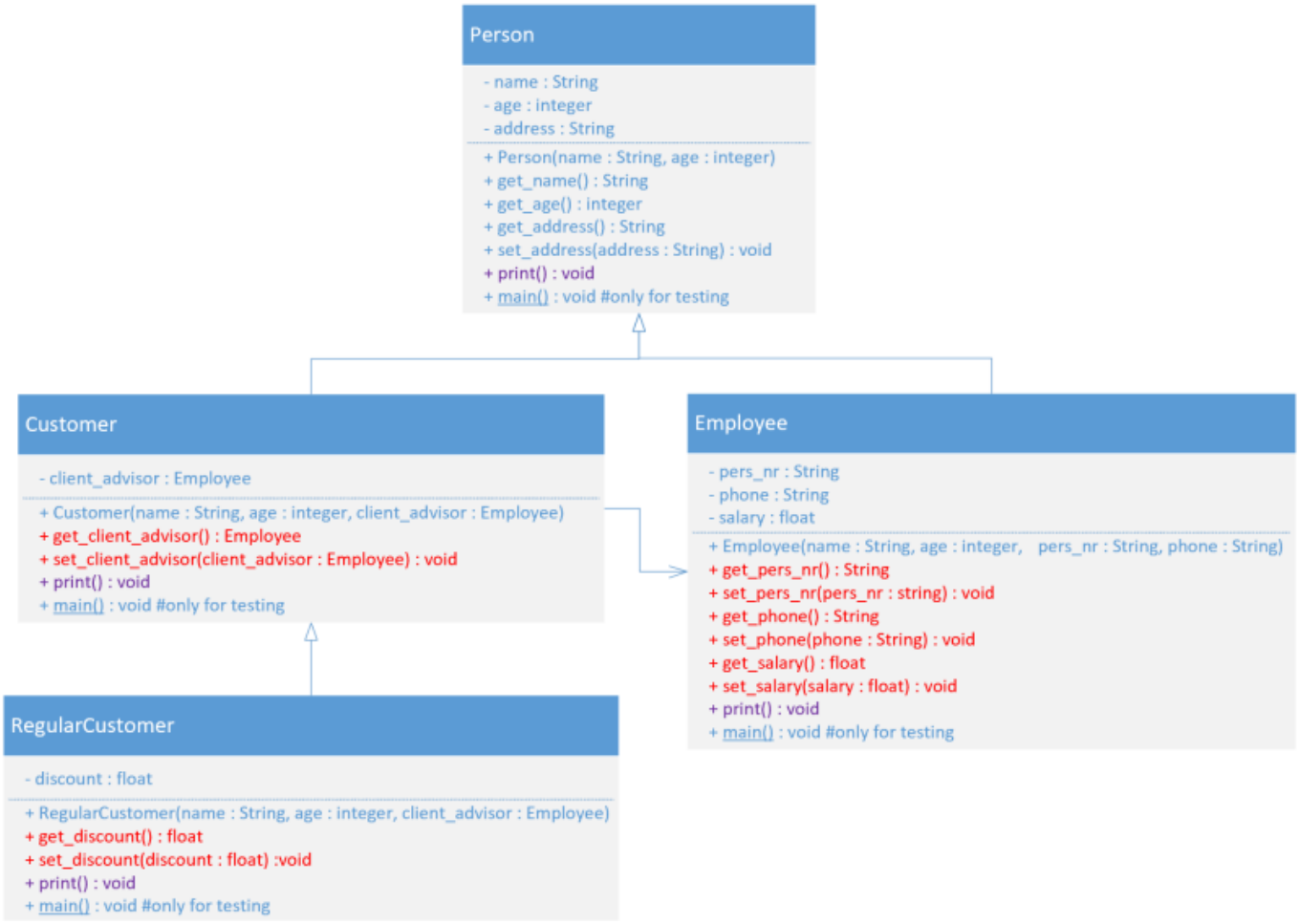
# Aufgabe 4 - Lösung

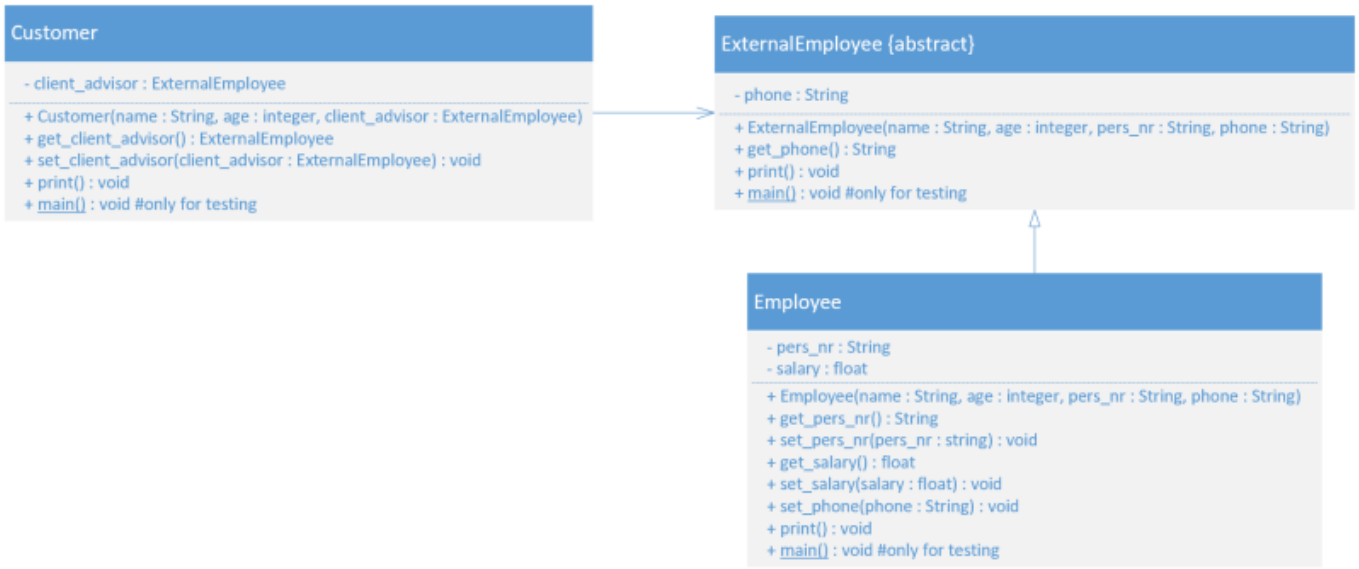
Sie finden hier die Lösungen zu den Teilaufgebn 6 und 7. Die praktischen Programmieraufgaben liegen im solution Branch auf github.

### Teilaufgabe 6




- Methoden die eine Klasse erweitern (zusätzliche Funktionalität) sind rot markiert.
- Methoden die überschrieben werden, sind violett markiert. Beim überschreiben bleibt der Name der Methode in allen abgeleiteten Klassen gleich.

### Teilaufgabe 7




Die Klasse `Employee` wird aufgeteilt. In der Klasse `ExternalEmployee` sind alle Attribute und Methoden zu finden, die bezüglich Datenzugriff unkritisch sind, hier z.B. `get_phone()`. Alle kritschen Attribute und Methoden werden in der Klasse `Employee` belassen, die nun aber von `ExternalEmployee` erbt. Die Klasse `Customer` sieht in dieser Konstellation nur ein Objekt der Klasse `ExternalEmployee` und hat somit keinen Zugriff auf die Methoden der Klasse `Employee`. Dadurch kann rein über die Struktur der Software der Zugriff verhindert werden.
Ein `Employee`-Objekt muss aber immer alle Daten umfassen! Es macht daher keinen Sinn, Objekte der Klasse `ExternalEmployee` zu instanziieren. Das wird verhindert, in dem diese Klasse als **abstrakte** Klasse deklariert wird; im UML-Diagramm am Keyword `{abstract}` zu erkennen.
Das Thema abstrakte Klassen folgt in Learningunit 7.

---