Exercise 8 – Object-oriented modelling and programming

In your shared house, you like to cook together ("cooking party"), where everyone brings something. As it is rather chaotic (Who paid what? Who receives how much money back from the others?), you have decided to bring a little more order to the finances of your shared house.

The idea of a cooking party is that initially every housemate buys certain things, and at the end a complete list of all purchases is made. Afterwards, everyone settles up: each person calculates how much he/she has spent; those who initially paid less than their fair share pay into the communal fund, while the others who initially paid more receive a corresponding payout from the fund — so that each person ultimately contributes equally.

Write a programme that outputs for multiple cooking parties:

- who brought what along,
- who initially paid how much, and
- who must pay back how much / who gets paid out how much from the communal fund.

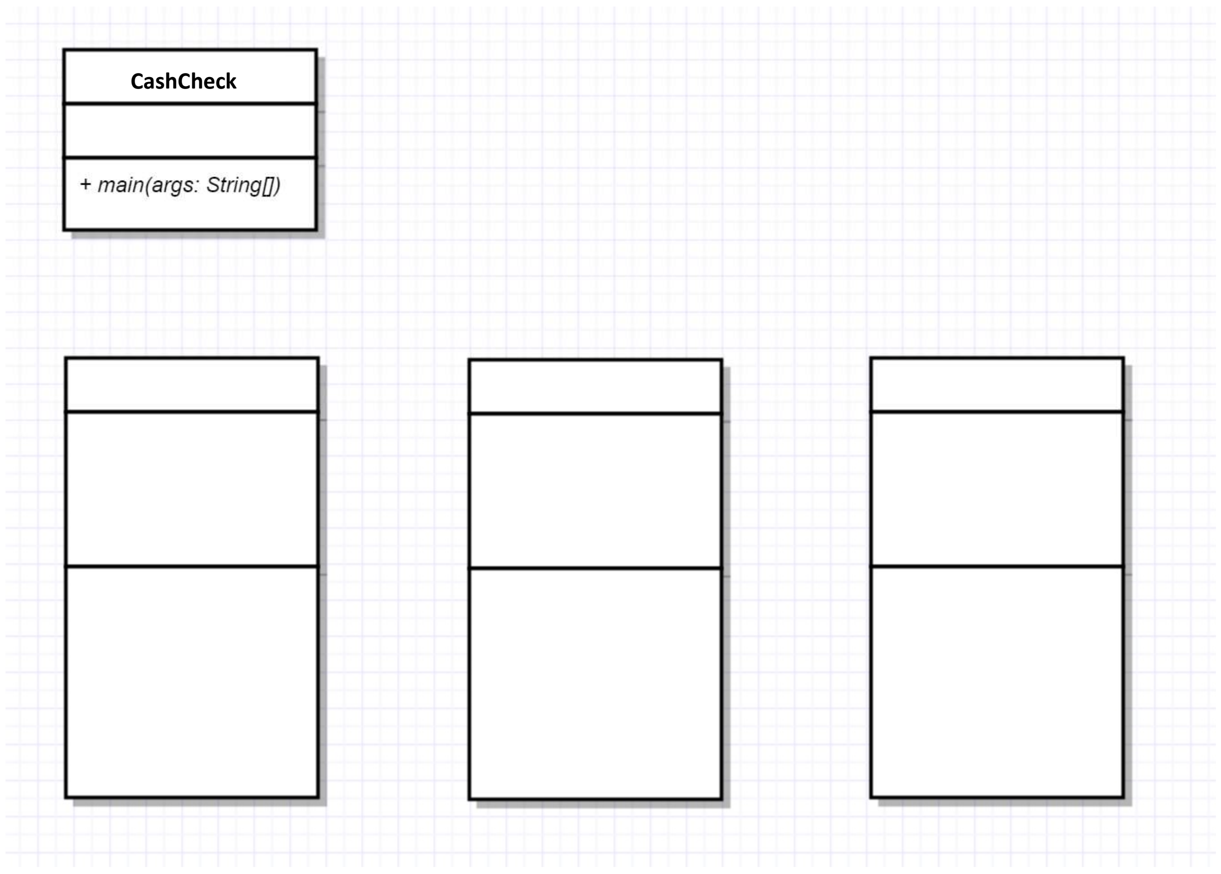
Task 1 - Modelling

Create a class diagram that appropriately abstracts the real world. What classes are there, what properties do they have, what tasks / behaviour do they have, and what relationships do they have with each other?
For example:

- At the St Nicholas' Day cooking party
  - Max brings olive oil (3.50), pasta (1.50) and tomato sugo (2.25)
  - Sarah brings minced meat (3.50), garlic (0.50) and an onion (0.50)
  - Peter brings a bottle of mulled wine (3.50)
  - Mary brings nothing, because she comes directly from a lecture.
- At the pre-Christmas cooking party
  - Harald brings biscuits from the bakery (4.50)
  - Greta brings a crate of beer (12.50)
  - Simon brings a crate of mulled wine (18.00)
  - Charly brings four packs of German lebkuchen/gingerbread (12.00 each)
  - Maria, Max, Peter and Sylvia bring nothing, because they come directly from sports.

Note:

- You should consider both the data as well as the methods and behaviour.
- Pay attention to the correct visibility for variables and for methods.
- The actual application should be implemented in a separate class called SettlingUp.
- It might be helpful to think about: who can best calculate how much he/she paid initially? Which class is the best way to calculate the fair share?
- Here, it is also worth calculating in cents again, and only converting to decimal Euros at the payout.



Task 2 – Implementing the classes

Implement your class diagram. Pay particular attention to ensuring that the encapsulation / secrecy principle (visibility, method and class responsibilities) have been implemented.

Think of suitable constructors to give the objects "everything they need".

Example output:

```
4 participants at the cooking party on 6.12.2019:
Max pays 7.25 for olive oil, pasta, tomato sugo,
Sarah pays 4.5 for minced meat, garlic, onion,
Peter pays 3.5 for mulled wine,
Total spent: 15.25, so 3.81 per person
Peter pays 0.31 into the fund
Maria pays 3.81 into the fund
Max receives 3.44 from the fund
Sarah receives 0.69 from the fund
It doesn’t add up exactly, balance of the fund: -0.01

8 participants at the cooking party on 23.12.2019:
Harald pays 4.5 for biscuits,
Greta pays 12.5 for a crate of beer,
Peter pays 18.0 for a crate of mulled wine,
Charly pays 48.0 for gingerbread, gingerbread, gingerbread, gingerbread,
Total spent: 83.0, so 10.37 per person
Harald pays 5.87 into the fund
Maria pays 10.37 into the fund
Max pays 10.37 into the fund
Peter pays 10.37 into the fund
Sylvia pays 10.37 into the fund
Greta receives 2.13 from the fund
Simon receives 7.63 from the fund
Charly receives 37.63 from the fund
It doesn’t add up exactly, balance of the fund: -0.04
```