

Exercise 7 – Transfer of the procedural solution approach to object orientation

The scenario is the same as in Exercise 6 – Task 1, but the task formulation for implementation is now slightly different.

Reminder:

The travel insurance policies of three insurance companies should be compared for a travel insurance portal. The premiums of the three insurance policies are calculated as follows:

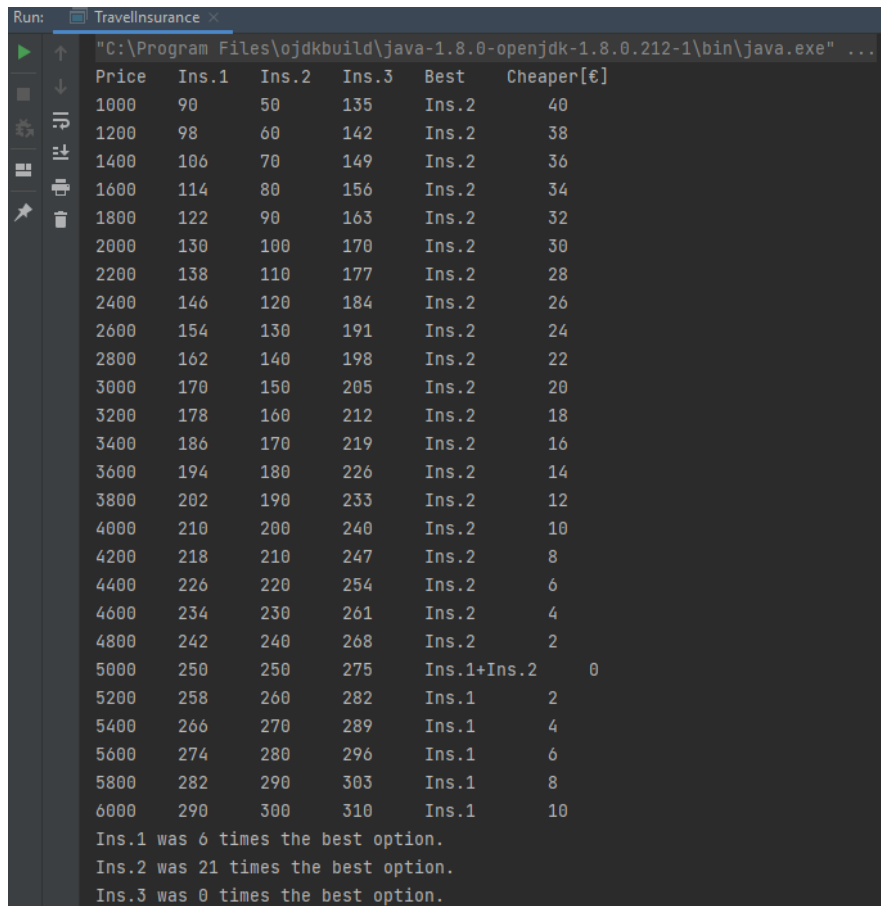
- Insurance policy 1: Premium = € 50 + 4% of the travel price best from 5k to 10k
- Insurance policy 2: Premium = 5% of the travel price best till 5k
- Insurance policy 3: Premium = €100 + 3.5% of the travel price best from 10k

The premiums for travel prices from €1,000 to €6,000 should be calculated in increments of €200.

For each travel price, it should be indicated which insurance policy is the cheapest. The price difference between the cheapest premium and the second cheapest premium must also be indicated. The premiums must be rounded off to whole number (integer) amounts.

In addition, the number of cases in which each insurance policy is the cheapest must be indicated.

The programme should output the results of the comparison in a table as follows:



Price	Ins.1	Ins.2	Ins.3	Best	Cheaper[€]
1000	90	50	135	Ins.2	40
1200	98	60	142	Ins.2	38
1400	106	70	149	Ins.2	36
1600	114	80	156	Ins.2	34
1800	122	90	163	Ins.2	32
2000	130	100	170	Ins.2	30
2200	138	110	177	Ins.2	28
2400	146	120	184	Ins.2	26
2600	154	130	191	Ins.2	24
2800	162	140	198	Ins.2	22
3000	170	150	205	Ins.2	20
3200	178	160	212	Ins.2	18
3400	186	170	219	Ins.2	16
3600	194	180	226	Ins.2	14
3800	202	190	233	Ins.2	12
4000	210	200	240	Ins.2	10
4200	218	210	247	Ins.2	8
4400	226	220	254	Ins.2	6
4600	234	230	261	Ins.2	4
4800	242	240	268	Ins.2	2
5000	250	250	275	Ins.1+Ins.2	0
5200	258	260	282	Ins.1	2
5400	266	270	289	Ins.1	4
5600	274	280	296	Ins.1	6
5800	282	290	303	Ins.1	8
6000	290	300	310	Ins.1	10

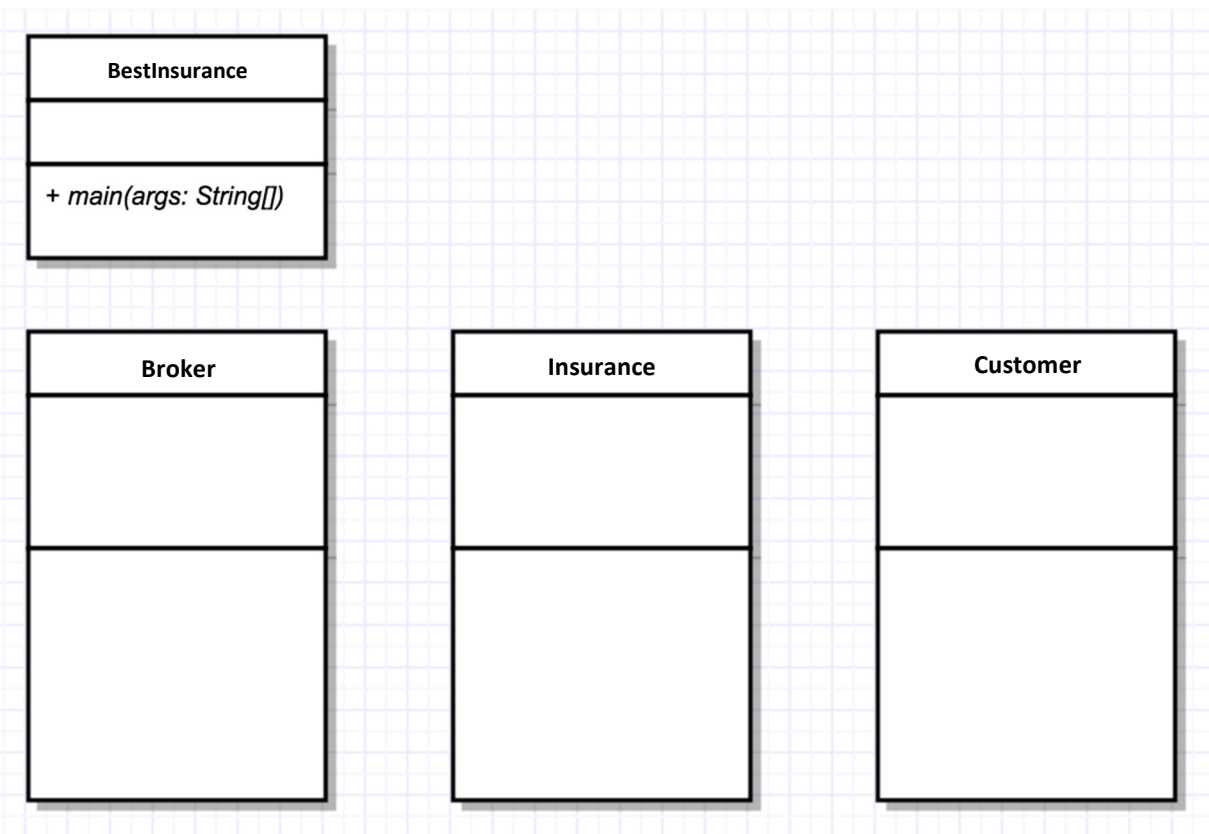
Ins.1 was 6 times the best option.
Ins.2 was 21 times the best option.
Ins.3 was 0 times the best option.

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?

Note:

- The connection between the classes is established through object attributes. Which classes are connected to each other, and how could the attributes be correctly defined here?
- The actual application should be implemented in a separate class called `BestInsurance`.
- You should consider both the data as well as the methods and behaviour.
- Pay attention to the correct visibility for variables and for methods.
- It might be helpful to think about: What is a number in the expense table? What is a row in the expense table? How is the entire table structured?



Task 2 – Implementing the classes

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

Also implement the `BestInsurance` class using the `main` method. If you have modelled properly, then

- only a few lines are necessary in the `main()`,
- no class file has more than around 50 lines,
- there is a dependency relationship between your classes, and
- the same result is produced as the last time.