

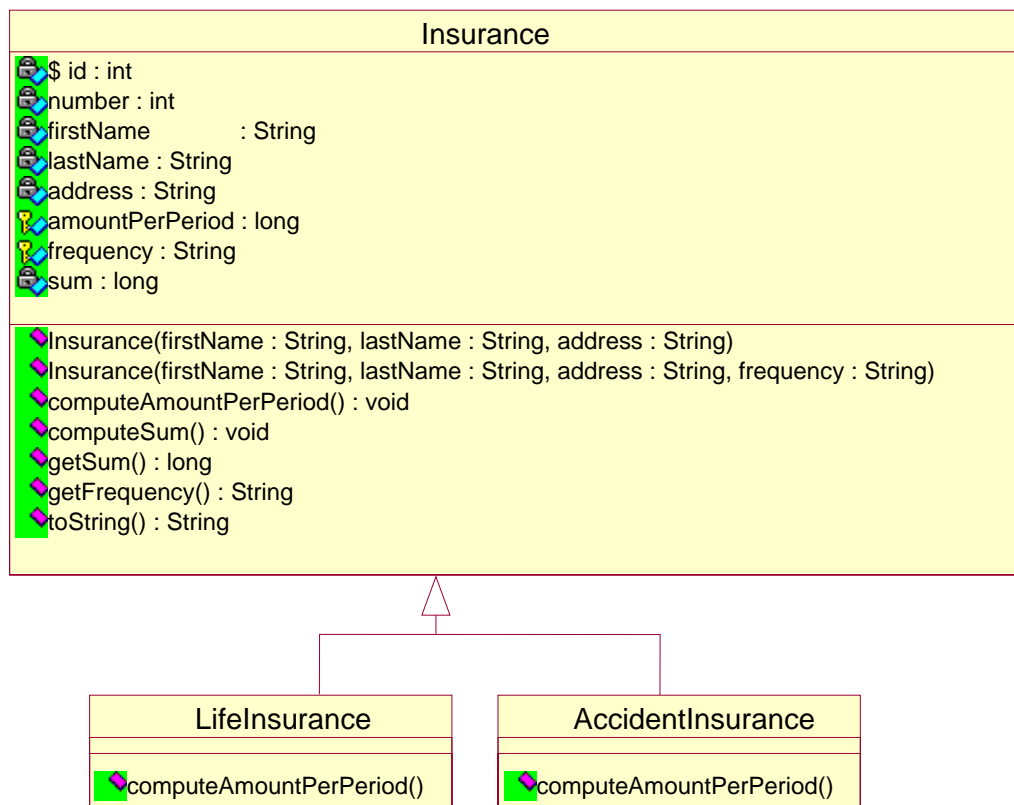
Lab 13. Recapitulative Exercises (II)

An insurance agency offers two kinds of insurance policies: life and accident policies. An insurance policy has a unique number, personal data about the insured person (first and last names, address) and other data like the amount paid per period by the insured person, payment frequency and the insured sum. The policy number will be automatically generated by the program.

The implicit payment frequency is monthly and the implicit amount per period is 13 €. But the insured person may choose another frequency function of insurance type. If the person has a life insurance policy, he/she may choose a quarterly payment, but the agency would add a 2% commission to the amount. If the person has an accident insurance, that person may choose a half-yearly payment, but the agency would add a 5% commission.

The insured sum is equal with the sum of all the amounts paid by the insured person and it is computed every time when the person pays the money per period. The program will know that an amount was paid by a person when the method `computeSum()` is called.

The model of the program is presented in the following:



Requirements:

1. Implement the classes from the model.
2. In order to test these classes, write the class `InsuranceTest` with a method `main` that carries out the following task:
 - a. Create one life insurance and one accident insurance (meaning, one object belongs to the class `LifeInsurance` and the other one is from the class `AccidentInsurance`). We assume that the second insured person has chosen to pay half-yearly.
 - b. Apply the methods `computeAmountPerPeriod()` and `computeSum()`.
 - c. Print all relevant information in the two insurances.
3. Improve the program such that all the objects created in the program are memorized in an objects dynamic collection. Before the program exiting, the collection content is memorized in a file "insurances.txt".
4. Write a method that separate the collection of `Insurance` objects in to collections, one for each `Insurance` subtype.
5. To solve 4, use a generic method that, given a collection of objects belonging to a superclass, creates a set of collections of objects, one for each subtype of the superclass. The result should associate at each subclass name the collection of objects of the subclass that are contained in the given collection.