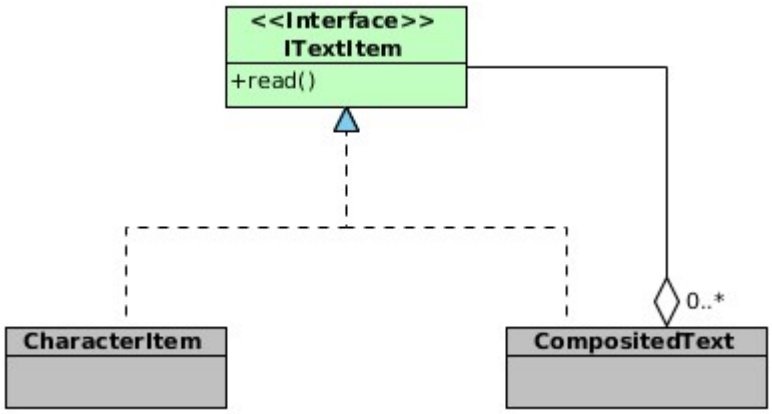
Software design patterns: 2010 – 2011
(exercise 3 – Adapter, Bridge, Composite & Decorator)

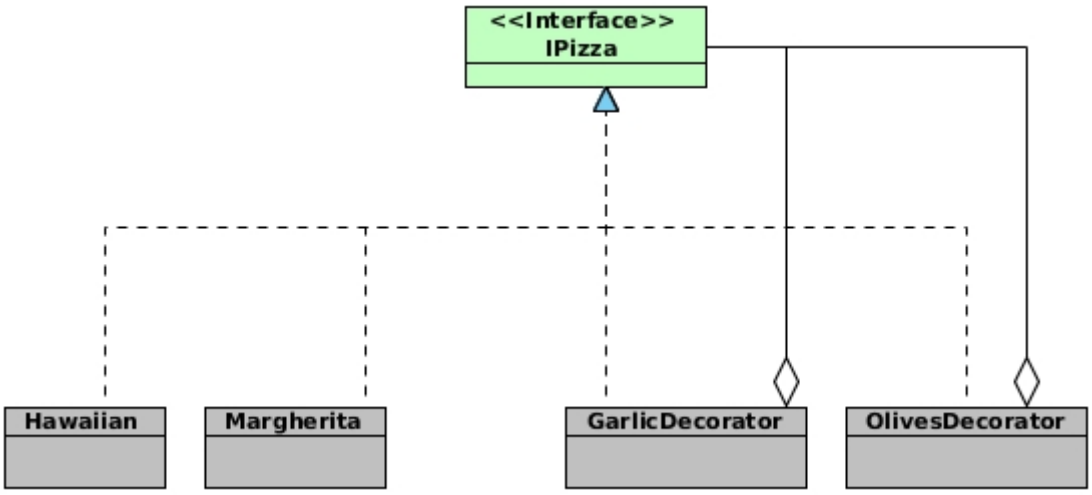
Упражнение 1.: Разгледайте следната UML диаграма съответна на предоставения сорс код, отразявайки Composite шаблона за конструиране на текст:



В сорс кода единствения довършен клас е **CharacterItem**, който представлява листото на йерархията (най-малката единица на композицията – буквата).
Нека довършим системата, като й дадем възможност и да записва на конзолата и композиция от букви: изречение, абзац (композиция от изречения), текст (композиция от абзаци) и т.н.
За това:

1.1. Имплементираме класа **CompositedText**.
1.2. Тествайте така кйонструираната програма в тестовия клас: **CompositeConsoleTest**

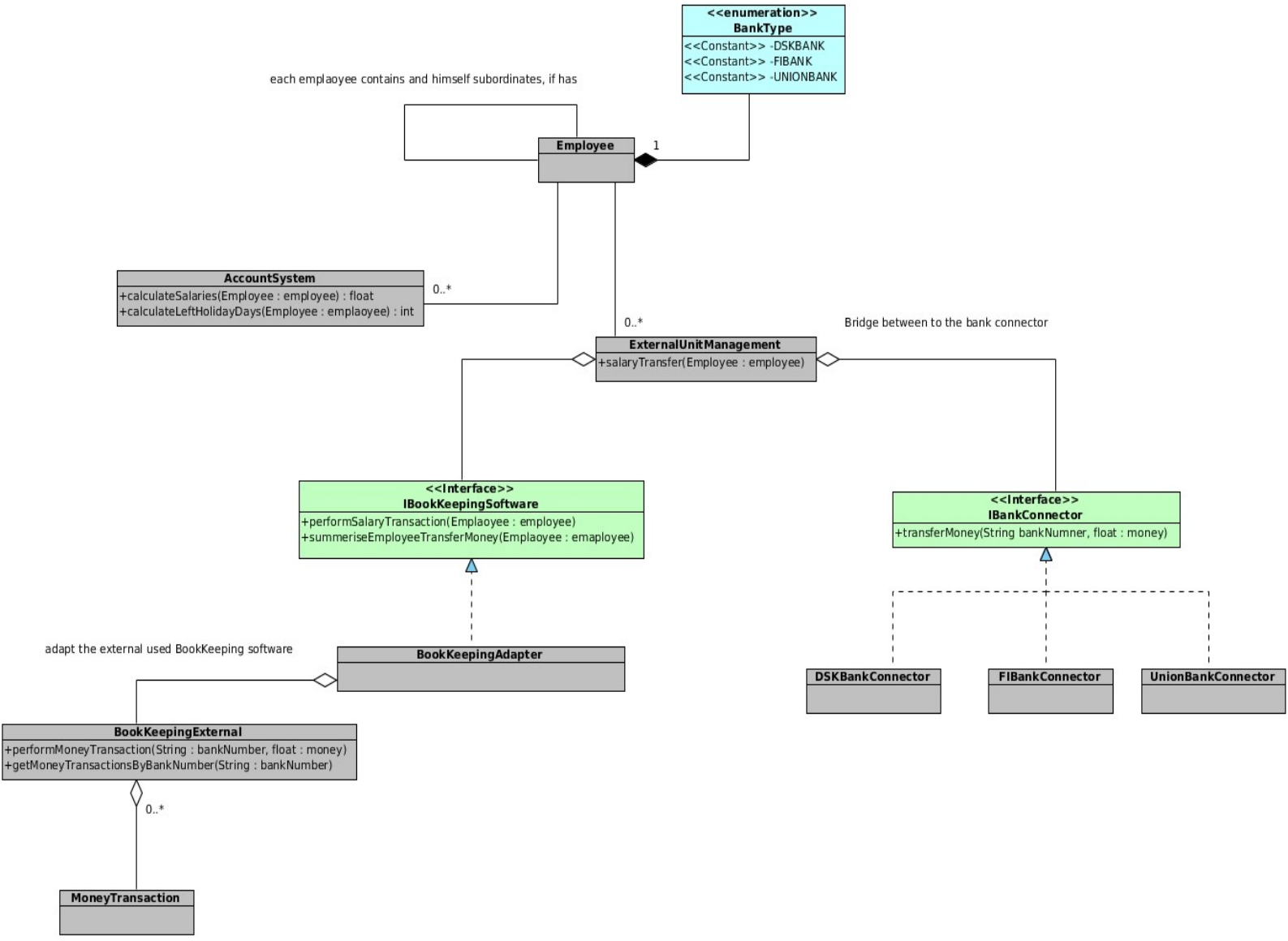
Упражнение 2. : Следната UML диаграма отразява система за направата на пици с възможността за всяка пица да се слага допълнително чесън, маслини или и двете едновременно към коя да е от пиците.



Системата е реализирана с помощта на Decorator шаблона, за който основните продута са: **Hawaiian** и **Margherita**, а добавката на екстрите се реаилиза с помощта на класовете **GarlicDecorator** и **OlivesDecorator**, които допълват с исканите продукти вече приготвените пици. Нека разпирим и тестваме системата като:

2.1. Добавете **Vegetarian** като основна пица
2.2. Добавете и допълнителните докорации (Decorators класове) **SpinachDecorator** и **AvocadoDecorator**, с които се дава възможност да се слагат иу допълнително спанак и авокадо.
2.3. Пригответе (изпойзвайки тестовия клас **DecoratorTest**) Margherita с авокадо, маслини, чесън и спанак.

Упражнение 3.: СледнатаUML диаграма :



представлява една малко по – сложна система за управление на хора в една компания, включвайки следните основни компонента:

- Служители в съответната йерархия. Разбира се, директорът е на най-високо ниво в йерархията
- Вътрешна счетоводна система за калкулиране на заплатите и броя останали дни отпуск за отдел (конкретна под-йерархия на служителите или за цялата компания)
- Конектори с външни банки, чрез които се правят преводи на заплатите на служителите. Всеки служител си има банка и банкова сметка, в която му се превежда заплатата
- Интегриран софтуер на т.н. BookKeeping система (примерно чрез Web Service), с която се пазят всички транзакции (банкови трансфери на заплати). Както и вътрешен BookKeeping модул, използващ външния такъв, като го адаптира, за да задоволи нуждите на копанията.

Имплементацията на тази система, която вия ще доръшете стъпка по стъпка, ще използва:

→ ***Composition design pattern***
→ ***Adapter design pattern***
→ ***Bridge design pattern***
→ ***Methods factory design pattern***
→ ***Singelton design pattern***

Ако стартирате тестовия клас: **EmployeeManagementTest** ще забележите, че системата не работи : невъзможност да се правят банкови преводи, нулеви резутати за заплатите на служителите и т.н. за това:

3.0. Направете **`ExternalUnitManagement`** `Singelton`

3.1. В **AccountSystem** класа имплементирайте методите:
-> **calculateSalaries(Employee employee)**,
калкулиращ заплатите на employee и неговите подчинени (йерархията под него) и
-> **calculateLeftHolidayDays(Employee employee)**,
калкулиращ свободните дни отпуска на employee и подчиненните му,
като използвате композитната йерархия на **Employee** класа.

3.2. **В ExternalUnitManagement** имплементирайте:

public IBankConnector getBankConnectorMethodFactory(BankType bankType)

метода, използвайки знанията си за метод фактори шаблона като с подадения параметър ни връща инстанция на конектор към съответната банка (**DSKBankConnector**, **FIBankConnector**, **UnionBankConnector**)

3.3. в **ExternalUnitManagement** в метода:

salaryTransfer(Employee employee) ,
имплементирайте мост (Bridge design pattern) между **ExternalUnitManagement** и **IBankConnector**, като :

3.3.1. Вземете конкретна имплементация на IbankConnector с помощта на вече имплементирания метод фактори (типа банка я взимата от emplaoyee)
3.3.2. Извикате метода transferMoney на инстанцията на IbankConnector със съответните му аргументи

3.4. Последната ни задача е да декларираме този паричен превод в KookKeeping системата. За това имплементирайте двата метода:

performSalaryTransaction(Employee employee) и
float summeriseEmployeeTransferMoney(Employee employee)

на **BookKeepingAdapter**, който с помощта на Adapter design pattern адаптира интерфейса на системата ни (**IBookKeepingSoftware**) към изпозлвалния софтуер на външната BookKeeping система: **BookKeepingExternal**