Principles of Object Oriented Programming
Spring 2021
Assignment 2

Published: 1/5/2021 **Due:** 16/5/2021

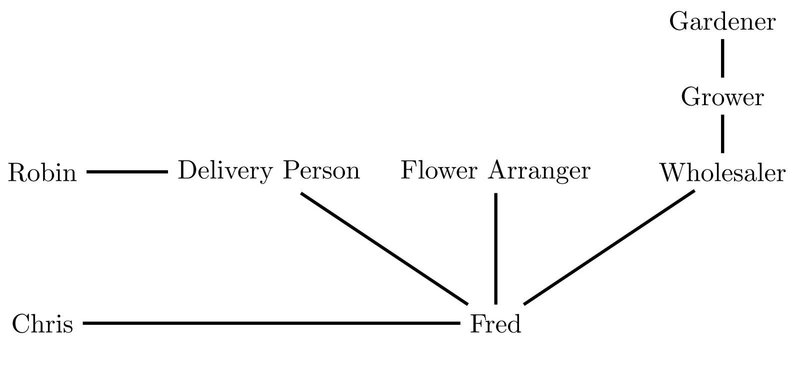
The assignment should be submitted in groups of two students.

1 General Description

In this assignment you will practice the following concepts:

- Messages
- Instances and Initialization
- C++

You will implement a simulator of the florist example given in Lecture #1, in which Chris orders flowers for Robin.
Specifically, your goal is to develop a system consisting of the following agents:



The links in the above figure represent connections between objects that should exchange messages (i.e., invoke methods) during the simulation. For example,

the link between Chris and Fred means that one of these objects should invoke a method of the other one.
You should perform the following steps:

- Design an appropriate **class diagram**. Think about the similarities and differences of the various classes you have to implement. The hierarchy structure in your class diagram should be reasonable, according to the principles learned in the lectures and sessions. The description should contain the methods provided by each class.
- Implement the simulation according to that design in C++ code.

2 Implementation Requirements

You are required to implement the following classes:

- Person
  - $name : std :: string$
  - $orderFlowers(Florist^*, Person^*, std :: vector < std :: string >) : void$
  - $acceptFlowers(FlowersBouquet^*) : void$
- Florist
  - $wholesaler : Wholesaler^*$
  - $flowerArranger : FlowerArranger^*$
  - $deliveryPerson : DeliveryPerson^*$
  - $acceptOrder(Person^*, std :: vector < std :: string >) : void$
- Wholesaler
  - $grower : Grower^*$
  - $acceptOrder(std :: vector < std :: string >) : FlowerBouquet^*$
- Grower
  - $gardener : Gardener^*$
  - $prepareOrder(std :: vector < std :: string >) : FlowersBouquet^*$
- Gardener
  - $prepareBouquet(std :: vector < std :: string >) : FlowersBouquet^*$
- Flower Arranger
  - $arrangeFlowers(FlowersBouquet^*) : void$

- Delivery Person
  - $deliver(Person^*, FlowersBouquet^*) : void$
- FlowersBouquet
  - $bouquet : std :: vector < std :: string >$
  - $is\_arranged : bool$
  - $arrange() : void$

2.1 General Notes

1. The program should start from Chris's object that request a flowers order with a string specifying the desired flowers.
2. The message is propagated through the network until it reaches the Gardener, who prepares an appropriated flowers bouquet: he creates a FlowersBouquet object, initializes it with the string that represents the flowers names, and sets the arranged field to false.
3. The gardener returns the bouquet, which eventually reaches the florist, who forwards it to the flower arranger. The flower arranger arranges the bouquet (sets the arranged field to true) and returns the bouquet to the florist. The florist then forwards the bouquet for delivery until eventually the bouquet reaches Robin.
4. To make the message-passing possible, each client object should hold a reference to its server object (or several server objects).
5. The object names should not be hard-coded in the message, but obtained using appropriate **getters** and **overriding**.
6. Remember to free the allocated memory using **delete** keyword.

Your program should produce the following output:

```
Chris orders flowers to Robin from Florist Fred: Roses, Violets, Gladiolus.
Florist Fred forwards request to Wholesaler Watson.
Wholesaler Watson forwards the request to Grower Gray.
Grower Gray forwards the request to Gardener Garett.
Gardener Garett prepares flowers.
Gardener Garett returns flowers to Grower Gray.
Grower Gray returns flowers to Wholesaler Watson.
Wholesaler Watson returns flowers to Florist Fred.
Florist Fred request flowers arrangement from Flower Arranger Flora.
Flower Arranger Flora arranges flowers.
Flower Arranger Flora returns arranged flowers to Florist Fred.
```

Florist Fred forwards flowers to Delivery Person Dylan.
Delivery Person Dylan delivers flowers Robin.
Robin accepts the flowers: Roses, Violets, Gladiolus.

3 Submission Instructions

Submit to the CS submission system a single archive file (.zip or .tar.gz) in the format [student1ID]-[student2ID].zip with the following contents:

1. hw2.pdf - should contains your class diagram modeling the code.
2. hw2 - your Visual Studio solution directory.