SAMPLE EXAM #3

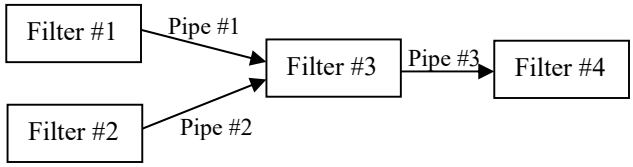
PROBLEM #1

Consider the problem of designing a system using the Pipes and Filters architecture. The system should provide the following functionality:

- Read student's test answers together with student's id.
- Read correct answers for the test.
- Compute test scores.
- Report test's scores.

It was decided to use the following Pipe and Filter architecture using the existing filters. The following existing filters are available:

Filter #1: this filter reads student's test answers together with student's id.
Filter #2: this filter reads correct answers for the test.
Filter #3: this filter computes test scores.
Filter #4: this filter prints test scores in the order as they are read from an input pipe.



1. The following are characteristics of filters and pipes:
   a. Filter #1 and Filter #2 are **active** filters.
   b. Filter #3 and Filter #4 are **passive** filters with **push** pipes.
   c. Pipe #1 and Pipe #2 are **buffered** pipes.
   d. Pipe #3 is a **un-buffered** pipe.
2. Use an object-oriented design to refine the design, where each filter is represented by a class. Provide a class diagram for your design. For each class identify operations supported by the class and its attributes. Describe each operation in pseudo-code.
3. Provide a sequence diagram for a typical execution of the system based on the class diagram of Part 2.

PROBLEM #2

There exist two servers **S1** and **S2**. Both servers support the following services:

Services supported by **server-S1**:
void Service1(string, int, int)
void Service2(string, int, int)
int Service3(string)
float Service4(string)

Services supported by **server-S2**:
void Service1(string, int)
void Service2(string, int)
int Service3(string)
float Service4(string)

There exist two client processes *ProcessA()* and *ProcessB()* and they request the following services:

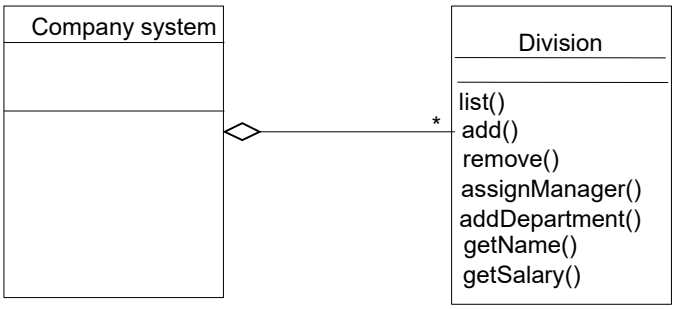
Client-A	**Client-B**
void Service1(string, int, int)	void Service1(string, int)
void Service2(string, int)	void Service2(string, int, int)
int Service3(string)	int Service3(string)
float Service4(string)	float Service4(string)

The client processes do not know the location (pointer) to servers that may provide these services. Devise a software architecture using a **Client-Dispatcher-Server** architecture for this problem. In this design the client processes are not aware of the location of servers providing these services.

- Provide a class diagram for the proposed architecture. Describe each component (class) of your design and operations supported by each class using the **pseudo-code**. In your design all components should be **decoupled** as much as possible. You do not have to specify operations of *Server-1* and *Server-2*. For *ClientA* only operation "*ProcessA()*" must be specified using pseudo-code showing how *ProcessA()* gets "*int Service3(string)"* service.

- Provide a sequence diagram to show how *ProcessA()* of *Client-A* gets "*int Service3(string)"* service.

PROBLEM #3
A company system consists of divisions as shown below:



Each division contains employees and departments. A department consists of a manager, developers and administrative assistants. In addition, a department may contain other departments. There are three types of employees: a developer, a department manager, and an administrative assistant. In addition, the system supports the following operations:

list() – a list of all employees of the current department is displayed
add(employee) – an employee is added to the department
remove(employee) – an employee is removed from the department
assignManager(employee) – assigns a manager to the department
addDepartment(department) – a department is added to the current department
getName() - returns a name of an employee
getSalary() - returns the salary of an employee

Develop a class diagram for the company system using the **Whole-Part design pattern**. Identify operations for each class. You do not have to specify operations using the pseudo-code.

It is expected that in the future new types of employees may be introduced into the system. Identify necessary changes to your design when new types of employees are incorporated into the system. Notice that required changes should be **minimal**.

PROBLEM #4

Suppose that we would like to use a fault-tolerant architecture for the *searching* component that is supposed to identify the number of elements in list *L* of *n* integers which values are between *x* and *y*, where $x \leq y$. The *search()* operation of this component accepts as an input an integer parameter *n*, an integer array *L* and integer parameters *x* and *y*. The *search()* component returns the number of elements in *L* which values are in the range between *x* and *y* (including values of *x* and *y*). An interface of the *search()* operation is as follows:

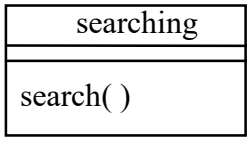
int search (int n, int L[], int x, int y)

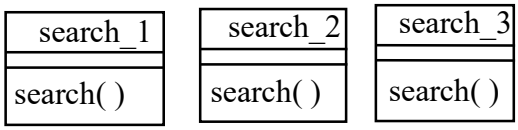
L is an array of integers.
n is the number of items in list *L*.
x is the lower bound
y is the upper bound

For example, for the following input:
n=10; L=(3, 2, 0, 10, 9, 5, -1, 8, 6, 7); x=2; y=7

The *search()* operation returns the value: 5

Suppose that three versions of the *searching* component have been implemented using different algorithms. Different versions are represented by classes: *search_1*, *search_2, search_3*.





Provide the design for the *searching* component using the **Recovery-Block** fault-tolerant software architecture. In your solution/design provide a class diagram. For each class identify operations supported by the class and its attributes. Specify in detail each operation using pseudo-code (you do not need to specify operations *search()* of the *search_i* classes). Notice that only new operations and the *search()* operation of the *searching* class need to be specified. In your solution/design provide a sequence diagram representing a typical execution of the *searching* component.