Practice Script 4

Summary:

- Object-Oriented Concurrent Programming (shared object communication model).
- Intra-object synchronization with Monitors.
- Intra-object synchronization with Readers-Writer exclusion scheme.
- Shared object synchronization programming.

Exercise 4.1

Implement a non-instantiable generic queue class (`Queue`) and two implementation classes. One with internal representation with an array (limited size) and other with a linked list (unlimited size).

Implement also classes for shared queues with the two types of synchronization: monitor and readers-writer exclusion.

Use the problem 3.1 (producers-consumers) to test the developed code.

Exercise 4.2

Implement a shared module to register log information (*Logger*). This module attaches current date (accurate to the millisecond) to each record. Each record contains also a textual classification type. Types are defined by clients, and must be defined before registering log information (i.e. a log to be accepted must provide an existing classification type).

The module's interface (Abstract Data Type) should follow the Design by Contract principles and contain at least the following public services:

- add new classification types (identified as strings);
- log messages (accepted only for existing types);
- a group of fetch services that return existing registered messages ordered by date for a list of classification types;

By default, all logger activity should be done both to standard output and a predefined text file named `logger.txt`.

A possible `Logger` class interface could be:

```
public class Logger {
   public boolean typeExists(String type);
   public void registerType(String type);
   public void log(String type, String message);
   public List[] fetchMessages();
   public List[] fetchMessages(String[] typeList);
}
```

Test this module with programs from previous class registering method invocation and termination (two different classification types).

Exercise 4.3

Is intended to develop a simulation of a barber shop.

A simulation program should be developed with the following parameters (passed as program arguments, with proper default values):

- Number of barbers in barber shop (maximum number of simultaneous hair cuts);
- Number of clients;
- Interval defining the limits of the number of cuts per client.
- Interval defining the limits of the random hair cut elapsed time.
- Interval defining the limits of the random outside barber shop elapsed time.

Use module logger developed in problem 4.2 to visualize the occurrences within this simulation.

Exercise 4.4

It is intended to construct a simulation of an automobile repair shop. When a car enters the shop it is necessary to identify existing faults (randomly determined), and then sequentially fix each fault. To ease the implementation consider the following:

- a car is identified by its license plate (a string text);
- there are only the following list of faults: paint, cleaning, motor, lights;
- each repair corresponds only to a register operation (together with a random selected waiting time) of the beginning and end of the operation.
- there is only one area (shared by all cars) for each fault type repair;
- each car passes sequentially by the required areas;

- the list of faults is attached to each car and randomly selected when the car enters the shop.
- The car is ready when all fault have been repaired (its exit from the shop must be registered).

All the processes must be visible in the simulation using the module developed in problem 4.2.