Лабораторная работа #2

Необходимо реализовать программу в соответствии с заданием. При реализации использовать паттерны проектирования, либо показать, что Ваше решение имеет преимущества. На защите продемонстрировать:

- работу программы;
- показать наиболее важные части решения с помощью диаграмм в нотации UML, IDEF и так далее (использовать достаточное количество диаграмм и степень детализации для демонстрации решения);
- быть готовым ответить на вопросы по выбранным решениям.

Для защиты задания все исполнители должны защититься.

Задание

Реализовать клиент-серверное приложение для передачи сообщений. Программа должна:

a. отображать список активных клиентов;
b. передавать сообщения различного типа между клиентами. При поступлении запроса на передачу получатель может либо принять, либо отказаться от приёма;
c. шифровать передаваемые сообщения;
d. вести журнал работы приложения как на клиентах, так и на сервере;
e. обеспечивать возможность докачки сообщения в случае разрыва соединения.

Для выполнения задания необходимо разбиться на команды по 3 человека. Задачи для каждого члена команды:

1. организовать клиент-серверную передачу сообщений. При реализации предусмотреть:

- возможность добавления новых протоколов взаимодействия (например, rest, soap, чистая передача данных). Необходимо реализовать возможность передачи с использованием двух протоколов (задаётся через настройки);
- возможность добавления новых типов передаваемых сообщений (картинки, файлы, большие данные, сообщения определённого формата). Необходимо реализовать 2 типа сообщений;
- передачу сообщений с докачкой. В случае разрыва соединения обеспечивать докачку сообщения после восстановления соединения;
- написать unit-тесты для каждого протокола передачи, для каждого типа сообщения и для докачки сообщений.

2. клиент и сервер используют **одну и ту же** **библиотеку, которая предоставляет реализацию журнала. Реализация журнала для клиента и для сервера разная. Реализовать запись информации о работе приложения в журнал на клиенте и на сервере, и саму библиотеку. При этом:**

- **реализация журнала для сервера должна сохранять сообщения в файле. Запись должна производиться мгновенно при поступлении сообщения. Для просмотра журнала использовать любой текстовый редактор;**
- **реализация журнала для клиента должна сохранять сообщения в памяти. При завершении сеанса работы приложения необходимо сбрасывать журнал на диск в файл с датой/временем начала и окончания сеанса работы в названии. Для просмотра журнала для текущего сеанса работы использовать пользовательский интерфейс. Для просмотра информации о предыдущих сеансах работы использовать любой текстовый редактор;**

- код, который создаёт и использует журнал, **должен быть одинаковым** и на клиенте, и на сервере

Пример. Обращение к журналу на клиенте и на сервере выглядит одинаково:

```
class Client {
	Logger logger; // сюда как-то должна попасть реализация для клиента
…
	void sendMessage(message) {
		// использование
		logger.log(message);
		…
	}
…
	void showHistory() {
		output(logger.getHistory());

	}
}

class Server {
	Logger logger; // сюда как-то должна попасть реализация для сервера
…
	// использование - так же, как и на клиенте
	logger.log(message);
…
})
```

Код дан только в качестве пояснения того, что требуется реализовать, и не должен быть воспринят как руководство к действию.

- в журнал записывать следующую информацию:
  - время начала и окончания сеанса работы программы;
  - информация о подключении/отключении клиента (для сервера);
  - информация о начале передачи (куда, откуда, время, тип сообщения и так далее);
  - информация о ходе процесса передачи;
  - информация об ошибках в процессе передачи - подробно;
  - информация об окончании процесса передачи.
- создать тесты для методов, которые используют журнал. Целью создания тестов является подтверждение работоспособности журнала. Помните, что в тестах не должна проверяться работа самой библиотеки журнала! Для этого обычно используется либо набор тестовых данных, либо какой-то способ, позволяющий убедиться, что необходимые методы журнала были вызваны

(*для приведённого выше примера при тестировании метода sendMessage() необходимо убедиться, что вызов logger.log() был произведён. При тестировании метода для отображения журнала, который использует logger.getHistory(), необходимо убедиться, что в историю попали все сообщения, которые были ранее записаны с помощью logger.log()*).

- **<u>запрещается:</u> создавать реализации компонента на клиенте и на сервере с помощью конструктора. Нужную реализацию запрашивать у библиотеки одним и тем же кодом** как на клиенте, так и на сервере.

3. реализовать шифрование передаваемых сообщений. Для этого использовать комбинацию синхронного и асинхронного шифрования. С помощью асинхронного шифрования передавать

пароль для синхронного алгоритма, затем производить шифрование передаваемого сообщения синхронным алгоритмом. Необходимо сделать следующее:

- выбрать библиотеки с реализацией необходимых алгоритмов шифрования - по две реализации синхронного и асинхронного алгоритмов;
- решить задачу с использованием выбранных библиотек;
- оградить код программы от выбранных библиотек для того, чтобы в будущем можно было легко заменить выбранные библиотеки на другие. Количество компонент, зависимых от библиотек, должно быть минимальным. Показать возможность замены на примере двух выбранных реализаций;
- написать тесты для процесса приёма/передачи сообщений. Предусмотреть, чтобы при смене библиотек тесты либо вообще не пришлось переписывать, либо изменения касались только инициализации тестов.