**Чат (1)**

Сегодня мы напишем чат. Набор программ с помощью которого можно будет  
обмениваться текстовыми сообщения. Набор будет состоять из одного сервера и  
нескольких клиентов, по одному для каждого участника чата.

Начнем с сервера. Нам понадобятся такие классы:  
1) Server – основной класс сервера.  
2) MessageType – enum, который отвечает за тип сообщений пересылаемых между  
клиентом и сервером.  
3) Message – класс, отвечающий за пересылаемые сообщения.  
4) Connection – класс соединения между клиентом и сервером.  
5) ConsoleHelper – вспомогательный класс, для чтения или записи в консоль.

Создай эти классы и добавь метод main в класс Server.

Требования:  
1. Класс Server должен быть публичным и создан в отдельном файле.  
2. Перечисление MessageType должно быть публичным и создано в отдельном файле.  
3. Класс Message должен быть публичным и создан в отдельном файле.  
4. Класс Connection должен быть публичным и создан в отдельном файле.  
5. Класс ConsoleHelper должен быть публичным и создан в отдельном файле.  
6. В классе Server должен быть создан метод public static void main(String[] args).

### Чат (2)

Первым делом, для удобства работы с консолью реализуем класс **ConsoleHelper**. В  
дальнейшем, вся работа с консолью должна происходить через этот класс.

**Добавь в него:**  
1. Статическое поле типа BufferedReader, инициализированное с помощью System.in.  
2. Статический метод writeMessage(String **message**), который должен выводить  
сообщение message в консоль.  
3. Статический метод String readString(), который должен считывать строку с  
консоли. Если во время чтения произошло исключение, вывести пользователю  
сообщение «***Произошла ошибка при попытке ввода текста. Попробуйте еще раз.***» И  
повторить ввод. Метод не должен пробрасывать исключения IOException наружу.  
Другие исключения не должны обрабатываться.  
4. Статический метод int readInt(). Он должен возвращать введенное число и  
использовать метод readString(). Внутри метода обработать исключение  
NumberFormatException. Если оно произошло вывести сообщение «***Произошла ошибка***  
***при попытке ввода числа. Попробуйте еще раз.»*** И повторить ввод числа.

В этой задаче и далее, если не указано дополнительно другого, то все поля класса должны  
быть приватными, а методы публичными.

Требования:  
1. В классе ConsoleHelper должно быть создано и инициализировано приватное статическое поле типа BufferedReader.  
2. В классе ConsoleHelper должен быть реализован статический метод writeMessage(String message), выводящий сообщение на консоль.  
3. В классе ConsoleHelper должен быть реализован статический метод readString, возвращающий строку считанную с консоли.  
4. В классе ConsoleHelper должен быть реализован статический метод readInt, возвращающий число считанное с консоли.  
5. Метод readInt должен использовать метод readString для чтения с консоли.  
6. Метод readString должен перехватывать IOException, выводить сообщение о некорректном вводе и повторять считывание с консоли.  
7. Метод readInt должен перехватывать NumberFormatException, выводить сообщение о некорректном вводе и повторять считывание с консоли.

### Чат (3)

Прежде, чем двигаться дальше, нужно разработать протокол общения клиента и сервера.

**Сформулируем основные моменты протокола:**  
— Когда новый клиент хочет подсоединиться к серверу, сервер должен запросить имя  
клиента.  
— Когда клиент получает запрос имени от сервера он должен отправить свое имя серверу.  
— Когда сервер получает имя клиента он должен принять это имя или запросить новое.  
— Когда новый клиент добавился к чату, сервер должен сообщить остальным участникам о  
новом клиенте.  
— Когда клиент покидает чат, сервер должен сообщить остальным участникам об этом.  
— Когда сервер получает текстовое сообщение от клиента, он должен переслать его всем  
остальным участникам чата.

Добавь для каждого пункта вышеописанного протокола соответствующее значение в **enum**  
**MessageType:**  
1) NAME\_REQUEST – запрос имени.  
2) USER\_NAME – имя пользователя.  
3) NAME\_ACCEPTED – имя принято.  
4) TEXT – текстовое сообщение.  
5) USER\_ADDED – пользователь добавлен.  
6) USER\_REMOVED – пользователь удален.

Требования:  
1. В перечислении MessageType должен присутствовать элемент NAME\_REQUEST.  
2. В перечислении MessageType должен присутствовать элемент USER\_NAME.  
3. В перечислении MessageType должен присутствовать элемент NAME\_ACCEPTED.  
4. В перечислении MessageType должен присутствовать элемент TEXT.  
5. В перечислении MessageType должен присутствовать элемент USER\_ADDED.  
6. В перечислении MessageType должен присутствовать элемент USER\_REMOVED.