УНИВЕРСИТЕТ ИТМО
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Отчёт по лабораторной работе №7
Вариант 401

Выполнил Ощепков Артём,
группа p3102
Преподаватель Письмак А. Е.

Санкт-Петербург 2017

Текст задания

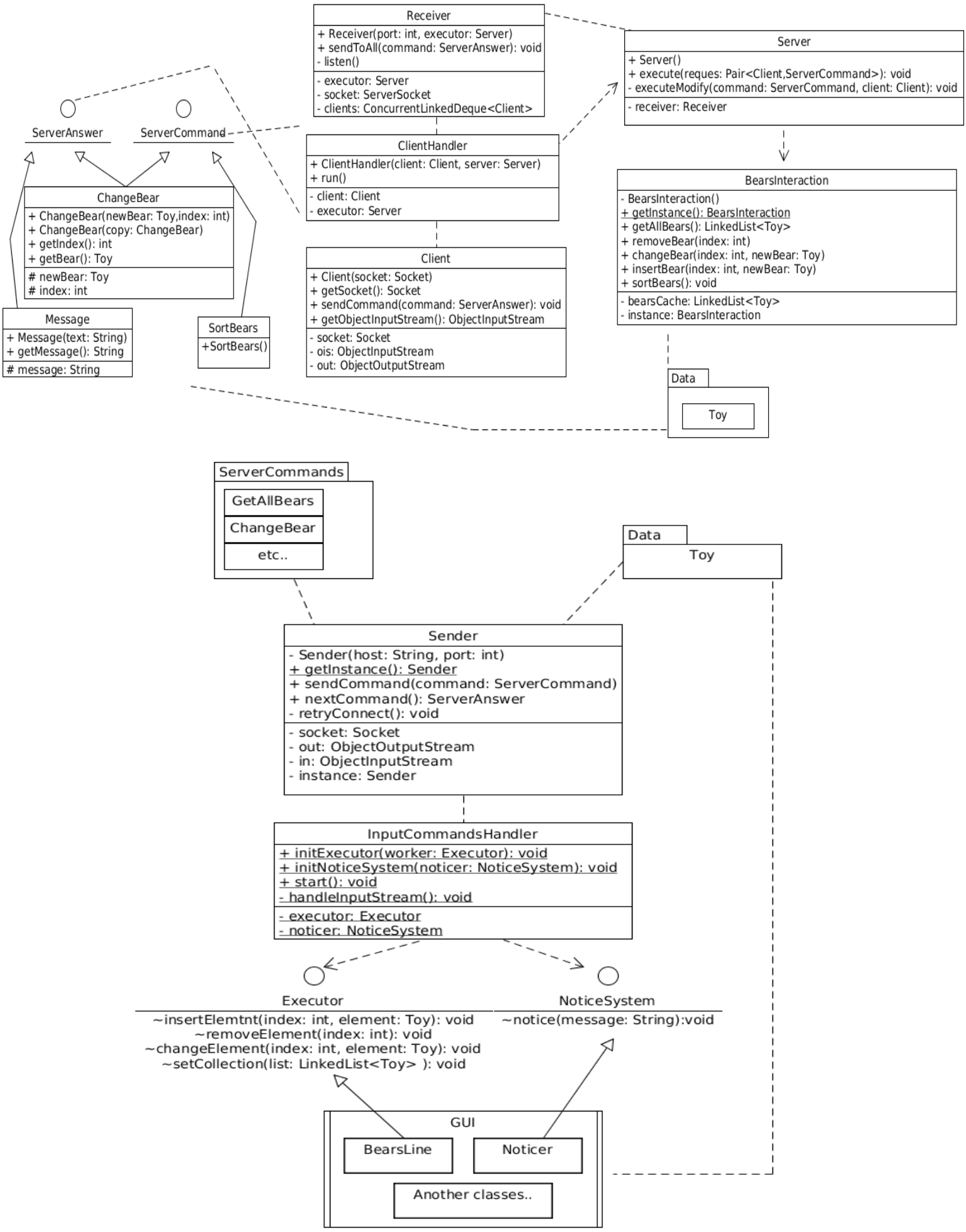
Разделить программу из лабораторной работы №6 на клиентский и серверный модули. Клиентский модуль должен обеспечивать взаимодействие с пользователем с помощью графического интерфейса. Серверный модуль должен реализовывать все действия с коллекцией, возвращая клиенту данные для отображения. Данные для коллекции должны храниться в базе данных PostgreSQL. Объекты должны передаваться в сериализованном виде.

Сервер должен поддерживать работу с несколькими клиентами, блокируя одновременные запросы на изменение данных. В случае, если клиент пытается выполнить операцию с неактуальными данными, сервер должен извещать об этом клиента. При этом клиент должен получить от сервера актуальные данные и обновить их у себя.

Получившаяся в итоге программа должна удовлетворять следующим требованиям:

1. Обмен данными между клиентом и сервером должен осуществляться по протоколу UDP.
2. При этом сервер должен использовать сетевой канал а клиент - датаграммы.
3. Для соединения с базой данных использовать `java.sql.DriverManager`.
4. Имя пользователя и пароль для соединения с базой задавать с помощью класса `Properties`.
5. Для получения результатов запроса использовать `javax.sql.rowset.CachedRowSet`.
6. Групповые операции удаления и вставки данных должны быть реализованы с использованием транзакций.
7. Одиночные операции модификации данных должны быть реализованы с использованием метода `PreparedStatement.execute()`.

Диаграмма классов программы



Исходный код:

```
public class Sender {
    private Socket socket;
    private ObjectOutputStream out;
    private ObjectInputStream in;
    private static Sender instance;
    private Sender(String host, int port) throws IOException {
        Thread t = new Thread(()->{
            while (true) {
                try {
                    retryConect();
                    System.out.println("Коннект восстановлен.");
                    sendCommand(new GetAllBears());
                    return;
                }catch (IOException exc){
                        Thread.currentThread().sleep(880);
                    }
            }
        });
        t.setDaemon(true);
        this.host = host;
        this.port = port;
        socket = new Socket();
        try {
            socket.connect(new InetSocketAddress(host, port));
        }catch (IOException e){
            t.start();
        }
        out = new ObjectOutputStream(socket.getOutputStream());
        in = new ObjectInputStream(socket.getInputStream());

    }
    public static Sender getInstance() throws IOException{
        if(instance == null)
            instance = new Sender("127.0.0.1",2552);
        return instance;
    }
    private void retryConect() throws  IOException{
        if(socket.isConnected())
            return;
        socket = new Socket(host,port);
        out = new ObjectOutputStream(socket.getOutputStream());
        in = new ObjectInputStream(socket.getInputStream());

    }
    /** Для общения с сервером используется паттерн "Команда" (ну, почти)
     *  Если Вы хотите отправить на сервер данные,
     *  должна существовать команда, принимающая в конструктор необходимые данные
     *  @param command <b> команда для отправки на сервер</b>
     */
    public void sendCommand(ServerCommand command) throws IOException{
        out.writeObject(command);
        out.flush();
    }
    public ServerAnswer nextCommand(){
            return (ServerAnswer)in.readObject();
}
public class InputCommandsHandler {

    private static Executor executor;
```

```
    private static NoticeSystem noticer;
    private static boolean isOn = false;
    public static void start(){
        if(isOn)
            return;
        isOn = true;
        Thread t = new Thread(()->{handleInputStream();});
        t.setDaemon(true);
        t.start();
    }
    public static void initNoticeSystem(NoticeSystem noticeSystem){ noticer = noticeSystem;    }
    private static void handleInputStream() {
        ServerAnswer command = null;
        while (true) {
            try {
                Thread.currentThread().sleep(1000);
            }catch (InterruptedException e){
            }
            command = Sender.getInstance().nextCommand();
            System.out.println("Получена команда" + command.toString());
            if (command instanceof ChangeBear)
                executor.changeElement(
                        ((ChangeBear) command).getIndex(), ((ChangeBear) command).getBear());
            if(command instanceof RemoveBear) {
                System.out.println("Собираюсь отправить на выполнение удаление элемента № " +((RemoveBear)
command).getIndex() );
                executor.removeElement(((RemoveBear) command).getIndex());
            }
            if(command instanceof  vt.smt.Commands.Message){
                if(noticer != null)
                    noticer.notice(((vt.smt.Commands.Message)command).getMessage());
                else
                    System.out.println( ((vt.smt.Commands.Message)command).getMessage());
            }
        }
    }
}
public interface ServerCommand extends Serializable {}
public class SaveAllBears implements ServerCommand, ServerAnswer{
    private LinkedList<Toy> data;
    public SaveAllBears(List<Toy> objects){
        data = new LinkedList<Toy>(objects);
```

```
    }
    public LinkedList<Toy> getData() {return data; }
}
public class BearsInteraction {
    private Properties user = new Properties();
    private CachedRowSet bearsSet;
    private LinkedList<Toy> bearsCashe;
    private BearsInteraction(){
            Class.forName("org.postgresql.Driver");
            user.setProperty("user", "bear");
            user.setProperty("password", "bear");
            connect = DriverManager.getConnection(
                    "jdbc:postgresql://localhost:5432/bears",
                    user.getProperty("user"),user.getProperty("password"));
            statement = connect.createStatement();
            bearsSet = new CachedRowSetImpl();
            bearsSet.populate(statement.executeQuery("select * from Bear"));
            bearsCashe = getBearsFromDB();
            System.out.println(bearsCashe);
            connect.close();
            bearsSet.close();
    }
    static BearsInteraction instance;
    public static BearsInteraction getInstance(){
       if(instance == null)
            instance = new BearsInteraction();
        return instance;
    }
    private LinkedList<Toy> getBearsFromDB(){
            LinkedList<Toy> ans = new LinkedList<>();
            while(bearsSet.next()){
               ans.add(new Toy( bearsSet.getString("name"),
                                    bearsSet.getFloat("weight"),
                                     bearsSet.getBoolean("isClean") ));
            return ans;
    }
    public LinkedList<Toy> getAllBears(){
       return bearsCashe;
    }
  public void commitChanges(){
    try {
       connect = DriverManager.getConnection(
```

```
            "jdbc:postgresql://localhost:5432/bears",
            user.getProperty("user"), user.getProperty("password"));
        statement = connect.createStatement();
        statement.execute("begin;");
        statement.execute("DELETE from bear");
        bearsCashe.forEach(e->{
            try {
                statement.execute(
                        "insert into bear(name, weight, isclean) VALUES('" +
                                e.getName() + "', " + e.getWeight() + ", " + e.isClean() + ")");
            }catch (SQLException ex){
                System.out.println("Беда при коммите мишек в БД");
                ex.printStackTrace();
            }
        });
        statement.execute("commit;");
    }catch (Exception e){
        try {
            statement.execute("ROLLBACK; ");
        }catch (SQLException oops){
            System.out.println("Приехали. С данными беда, сисадмин кричит, программист молятся");
            oops.printStackTrace();};
    }
}
    public void changeBear(int index, Toy newBear){
        if(index >= 0 && index <bearsCashe.size())
            bearsCashe.set(index,newBear);
        else
            System.out.println("Попытка изменить медведя с несуществующим индексом");
    }
}
```

Выводы:

Лаба классная, выполнять интересно. Я познакомился с основами сетевого взаимодействия приложений с помощью стандартных средств языка Java и вывел своих Медведиков на орбиту из глуши берлоги базы данных.