Тема:

SimpleNet

Проект 622

Направление: Приложни програми

2. Автор:

Име: Калин Петков Кръстев

Дата на раждане: 02.05.2000г.

Адрес: ж.к. Красно село кв. Борово 232 вх. Б ет. 5 ап. 35

Телефон: 0888744980

Имейл: kalin.p.krustev@gmail.com

Училище: Първа частна математическа гимназия, гр. София

Клас: 11

3. Ръководител:

Име: Диана Петрова Петрова

Телефон: 0882277384

Имейл: dianapetrova@abv.bg

Длъжност: Старши учител по информатика и ИТ

Месторабота: Първа частна математическа гимназия, гр. София

4. Резюме

4.1 Цели

Проектът е предназначен за разработчици, а не за потребители. Написан е на *Java* и може да бъде използван само от програми написани на същия език. Целта му е да улесни неопитни разработчици, които не са напълно запознати с *java.net* пакета. Приложението предоставя методи за: осъществяване на клиент-сървър връзка, изпращане на съобщения от клиента до сървъра и от сървъра до всеки един клиент както и за прекъсването на тази връзка. В приложението се съдържат класове, които могат да се използват като основа за двете страни на клиент-сървър мрежовата архитектура. Силната страна на проекта е, че той предоставя вградена възможност за криптиране на комуникацията между устройствата. Преносът на данни и криптирането са напълно интегрирани в приложението и не представляват пречка за неопитният разработчик. За него остава само да реши какво да изпраща и как да третира получените съобщения. Съществуващи решения на този проблем, но не предоставят опция за криптиране или ако предоставят не е написана в лесен за разбиране и използване вариант.

4.2. Основни етапи в реализирането на проекта

4.2.1 Реализиране на библиотеката

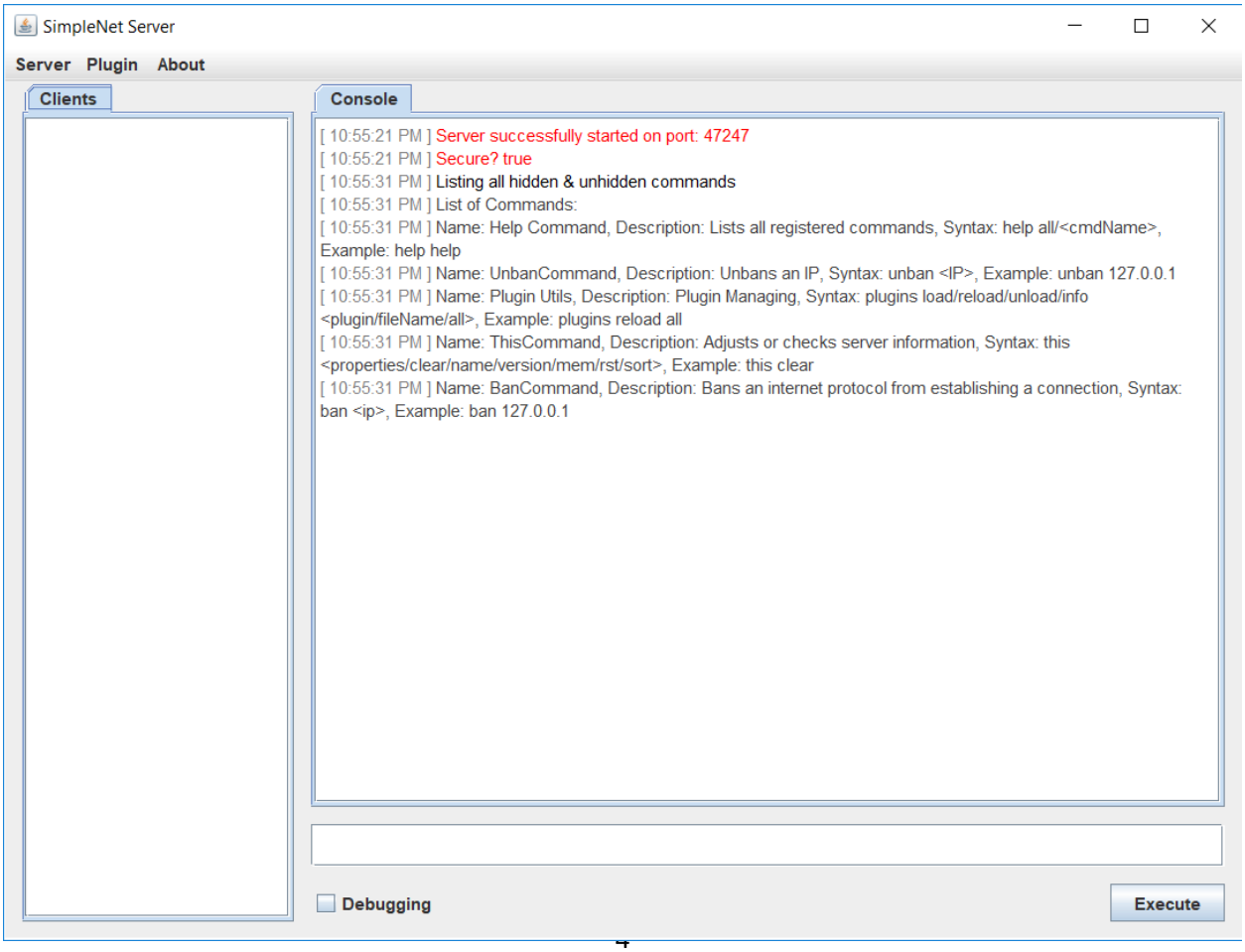
Първият етап от развитието бе създаването на основната функционалност – класовете *Client, Server* и *SerializableMessage*. Около тях са написани всички други класове в главния пакет. Първият гореспоменат клас представлява *клиентът* в клиент-сървър архитектурата, втория – *сървъра*, а третия представлява общият вид на съобщенията, които ще бъдат изпращани.

Вторият етап се състои в адаптирането на вече написаните класове да изпозлват интернет протоколи. Тази адаптация е отделена в подпакет на име *internet*, който се разделя на *udp* и *tcp*. В тези пакети се съдържат неабстрактни класове, които разширяват вече съществуващите основни класове в главния пакет – *Client* и *Server*. Това са класовете предназначени за използване от разработчиците, които искат да работят с интернет протоколи. Всеки друг протокол може да бъде написан, използвайки двата основни класа.

Следващият етап е подсигуряване на комуникацията чрез криптиране и така „се роди“ пакетът *encryption*. Единственият клас в него е *EncryptedMessage*, който разширява основния клас *SerializeableMessage*, като предоставя възможност информацията в него да бъде криптирана. Имплементираните видовете криптирания са два – един симетричен в пакет *aes* и един несиметричен в пакет *rsa*.

4.2.1 Реализиране на сървърното приложение

След написване на библиотеката следва създаване на приложение за комуникация с потребителите. Целта му е да улесни още повече разработчиците използващи библиотеката. Графичният интерфейс е отделен в друго приложение, наречено *сървърно приложение*. Първият етап в създаването му беше да се имплементира вече написаната библиотека и вграждането е отделено в пакета *networking*, а графичният дизайн в пакета *ui*. Елементите, които съдържа сървърното приложение са: две текстови полета – конзола и поле за въвеждане на данни, списък с всички свързани клиенти и меню, разположено в северната част на панела.



Следващата стъпка беше да се напише функционалността на полето за въвеждане на данни. Целта на това поле е да дава наставления на програмата и да я управлява докато работи. След въвеждането на текста, в полето за вход на данни се натиска клавиатурният клавиш *Enter* или графичният бутон *Execute* и командата се изпълнява. Управлението на командите е отделено в пакет *command*. Първият клас написан в този пакет е *Command* и той представлява абстрактен клас, който е основата на всяка команда. Следващият написан клас е *CommandMap*, който отговаря за управлението на всички команди. За да може една команда да се изпълни, тя трябва да бъде регистрирана в инстанция на гореспоменатия клас. Следващият етап е написването на всяка една команда, нужна за основното функциониране на сървърното приложение – налагане на забрана за осъществяване на връзка с определен интернет адрес, отмяна на тази забрана и настройки на самото приложение. Тези команди са отделени в подпакет *defaults*. След като функционалността на сървърното приложение беше написана остана само да се осъществи комуникацията на разработчика със сървърното приложение. Пътят за осъществяването на тази комуникацията беше имплементирането на система от приставки (plug-in, плъгин). По дефиниция от *Wikipedia.org* плъгин представлява "софтуерен компонент, който се инсталира в допълнение към съществуващо софтуерно приложение (компютърна програма), за да предостави на потребителя допълнителна функционалност, която не е част от базовото приложение." В конкретният случай тази приставка е кодът написан от разработчикът. За да може да бъде имплементиран в сървъра трябва да изпълнява няколко условия:

1. Да бъде изпълним файл с разширение jar.
2. Да съдържа клас, който да имплементира всички методи на написания в сървърното приложение файл *Plugin.java*, наречен накратко главен клас.
3. Да съдържа текстов файл, спазващ определен формат, който да навигира четеца на сървърното приложение къде се намира главният клас на приставката, какво е името ѝ, и каква е версията ѝ.

Всеки файл, който изпълнява тези условия се нарича плъгин за това приложение. Начинът на зареждане на този файл в сървърното приложение е следния: Със стартирането на сървъра се търси директория на име *plugins* и ако не е намерена, същата се създава. Тази

директория е предназначена да съхранява приставките. Ако при стартирането се открият плъгини, те се зареждат. В случай, когато файл бива добавен към вече стартирано приложение, неговото зареждане трябва да бъде зададено с команда от командното поле в графичния интерфейс. Имплементацията на тази функционалност е отделена в пакета *plugin*. Там се намира основата на всички плъгини – абстрактният клас *Plugin*. Това е класът, който прави връзката между разработчикът и приложението. В него има методи, с които уведомява разработчикът когато:

- Плъгинът бива активиран
- Плъгинът бива деактивиран
- Клиент се присъдини към сървъра
- Сървърът получи съобщение от клиент
- Клиент прекрати връзката си със сървъра

Останалите класове в пакета са написани да помогнат на плъгина да се интегрира.

4.3 Ниво на сложност на проекта 3 – основни проблеми при реализация

4.3.1 Поддържането на връзка с повече от един клиент

4.3.2 Приемане и изпращане на съобщения. Сериализация и десериализация

4.3.3 Подсигуряване на съобщенията. Криптиране и декриптиране

4.4 Логическо и функционално описание на решението

4.4.1 Поддържането на връзка с повече от един клиент

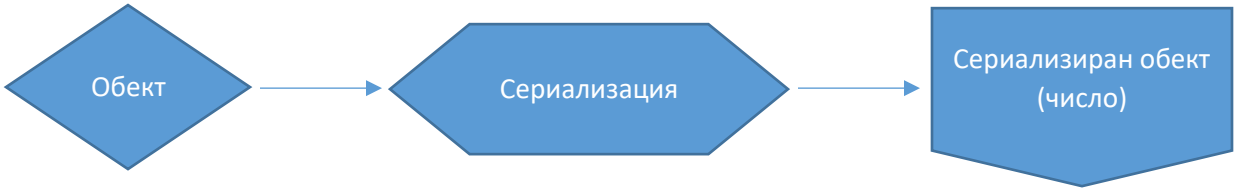
Когато сървърното приложение осъществи връзка с клиент, то постоянно чете дали има нови данни подадени от този клиент. Тази проверка ангажира цялото приложение и прекратява всякакви други операции докато работи. За да се отстрани този проблем приложението е разделено на отделни нишки (*thread*-ове). Идеята на тази нишка е да изпълнява командите, които ѝ се задават. По подразбиране всяко приложние има по една нишка, която изпълнява зададените инструкции. Нишките нямат заделена памет, а споделят ресурсите на процеса, който обитават. Когато сървърът приеме нов клиент, той създава нова

нишка, която е предназначена само за клиентът и която бива прекратена, когато клиентът прекъсне връзката. С поддържането на отделна нишка за всеки клиент, приложението работи без забавяния.

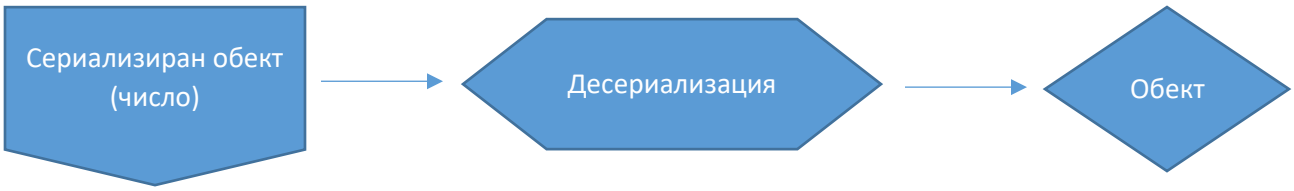
4.4.2 Приемане и изпращане на съобщения. Сериализация и десериализация

За да може едно съобщение да бъде изпратено, то трябва да бъде приведено в един общ вид. Този общ вид представлява резултата от сериализирането на това съобщение. По дефиниция от *Wikipedia.org* сериализацитя представлява „процес на преобразуване на структури от данни или обекти до поток от байтове запазвайки състоянието на техните полета и свойства.“. Използването на тази техника позволява да бъдат изпращани цели класове. Когато обектът бива сериализиран, той бива изпращан. Обратното действие на сериализацията е десериализацията и чрез прилагането на този процес се пресъздава идентично копие на оригиналното съобщение.

Сериализация:



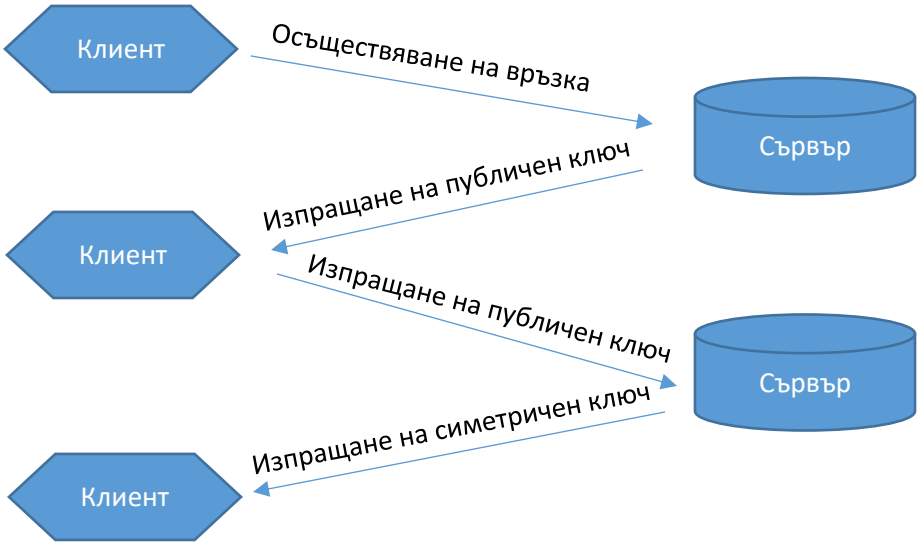
Десериализация:



4.4.3 Подсигуряване на съобщенията. Криптиране и декриптиране

За да може едно съобщение да бъде криптирано от библиотеката, то трябва да бъде под формата на число. Това условие се изпълнява от сериализацията на това съобщение, тъй

като винаги когато едно съобщение бива сериализирано, то бива преобразувано в число. Библиотеката поддържа два вида криптиране: симетрично под формата на AES (Advanced Encryption Standard) и асиметрично – чрез алгоритъма RSA. Подсигуряването на съобщенията става като и клинетът и сървърът разменят публичните си асиметрични ключове и след това сървърът изпрати симетричен ключ. Използването на този ключ позволява на двете страни да криптират съобщения бързо и ефикасно.



4.5 Реализация

За създаването на сървъра бяха необходими два ресурса: *Java Development Kit 9 и IntelliJ IDEA*. Първият ресурс е необходим, за да се използва езикът за програмиране *Java*, а вторият е софтуерът, който представлява средата за писане. За демонстрация на функционалност на приложението се използва софтуерът *WireShark*. За написването на документацията и изготвяне на презентацията се използва *MS Office 2013*.

4.6. Описание на приложението

Приложението се стартира като се отвори файлът *SimpleNet.jar*. За да може да функционира правилно приложението трябва да има правомощия да чете и да пише в директорията, в която се намира. Приложението няма опция да се инсталира или деинсталира. Поддръжка не е нужда. Дори някой от файловете да бъде изтрит, той ще бъде заменен от нов, генериран от приложението.

4.7. Заключение

Едно от приложенията е софтуер, който е написан за демонстрация като част от презентацията. Приложението още не е публикувано като завършен проект. Възможностите за развитие и усъвършенстване включват:

- Имплементация на интернет протоколът *UDP*, защото в момента разполага само с *TCP*
- Добавяне на още методи за криптиране, както и настройки на криптирането
- Добавяне на още функционалност на приставките (плъгините)
- Добавяне на режим на работа без графичен интерфейс