**Цель работы:** изучить технологию RMI, ее возможности, принцип реализации и способы применения, научиться работать с ней на примере прикладных задач.

**Задание:** реализовать сервер – радио. Клиенты посылают на сервер запрос, содержащий песню, дату, когда она должна прозвучать на радио, и поздравление, которое должно прозвучать перед песней. Также клиент может по запросу посмотреть все запросы на определенную дату.

**Общая теория:**

RMI – программный интерфейс вызова удаленных методов в языке Java. С помощью этой технологии можно вызывать удаленные методы, работающих на другой JVM. Основное достоинство RMI заключено в том, что вызов удаленных методов происходит так же, как и локальных, что избавляет от необходимости самостоятельно налаживать обмен данными между двумя (или более) JVM.

Понятие клиент и сервер для двух JVM, использующих RMI, условно, т.к. в разные моменты времени каждая из них может удаленно вызывать методы другой, меняясь ролями клиента и сервера.

Для того, чтобы удаленно вызвать метод, клиент должен знать его определение. Определяются методы удаленного доступа в интерфейсах, расширяющие класс Remote. Определение МУД (метод удаленного доступа) обычное, за исключением указание на выбрасываемое исключение типа RemoteException.

Клиент не знает реализации МУД. Зато они должны быть реализованы на сервере. Сервер реализует интерфейс определения МУД, т.е. реализует сами МУД.

В обмене данных между клиентом и сервером в технологии RMI участвуют классы Заглушка (Stub) – отправитель, и Скелет (Skeleton) – получатель. Их реализации скрыты от пользователя.

Stub является представителем серверного объекта, с его помощью клиент вызывает МУД. Stub осуществляет связь с сервером. Он создает блок байтов (идентификатор удаленного объекта, описание вызванного МУД, развернутые (сериализованные) параметры), и посылает этот блок серверу.

На сервере Skeleton принимает блок байтов, ищет вызванный метод, свертывает параметры, выполняет вызванный метод, разворачивает возвращаемое значение (если есть) и передает stub-у клиента. Stub клиента сворачивает возвращаемое значение (или исключение) и возвращает, как если бы МУД был реализован на клиенте и возвращал значение после выполнения.

Клиент и сервер находят друг друга при помощи репозитория RMI (rmiregistry). Он представляет собой список имен-идентификаторов(строк), при помощи которых клиент определяет, куда следует слать запросы со stub-а. Сначала rmiregister должен быть запущен на сервере. После чего на сервере создается объект удаленного доступа (stub), который затем связывается в rmiregistry со строковым идентификатором. После чего клиент может получить ссылку на объект registry с сервера и найти в нем ссылку на stub, использовав необходимый строковый идентификатор, чтобы использовать его для удаленного вызова.

**Ход работы:**

1. Нашел рекомендованный преподавателем пример использования RMI на сайте Oracle, прочитал теорию, поясняющую данный пример, при помощи гугл-переводчика.
2. Создал проекты клиента и сервера в Netbeans, скопировал в них код из примера, попытался скомпилировать как есть. К сожалению, ничего не вышло.
3. Более детально разобрал код примера, попробовал запустить rmiregistry через командную строку, но безуспешно.
4. Нашел пример использования RMI на Habrahabr.ru, проанализировал код приведенного там примера, узнал, как создать репозиторий RMI на сервере без использования командной строки.
5. Применил полученные знания, пошаманил с интерфейсом удаленного доступа, и, наконец, успешно выполнил сборку обоих проектов.
6. Изменил проекты под свое задание (Radio). Реализовал на сервере методы getRecord и printAllRecord, реализовал на клиенте контекстное меню и функционал для использования методов удаленного доступа, изменил определение методов в интерфейсе удаленного доступа.
7. Прочитал теорию по использованию технологии RMI, написал отчет по лабораторной работе.
8. Получил дополнительное задание: реализовать обмен между клиентом и сервером при помощи объектов, а также использовать HashMap на сервере в качестве хранилища. Реализовал это.

**Скриншоты:**

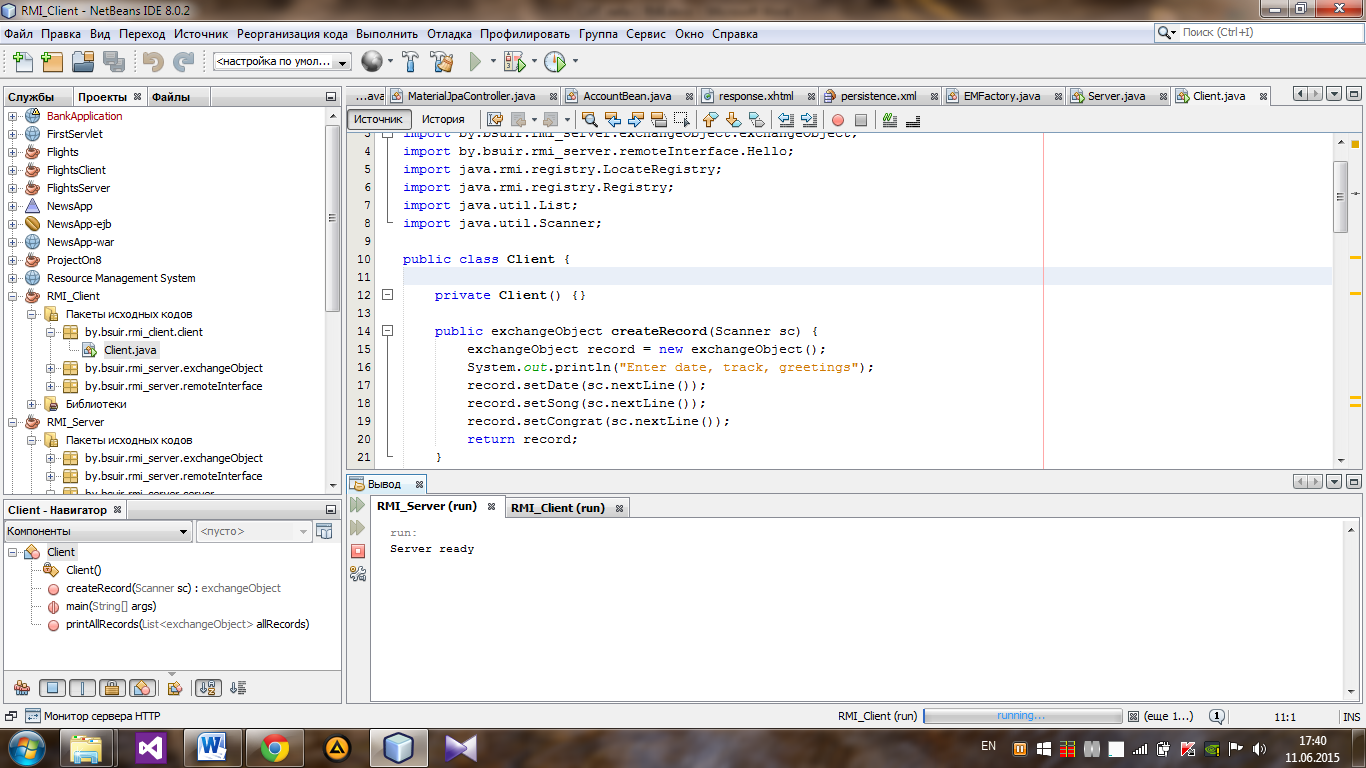


Рисунок 1 – Запускаем сервер

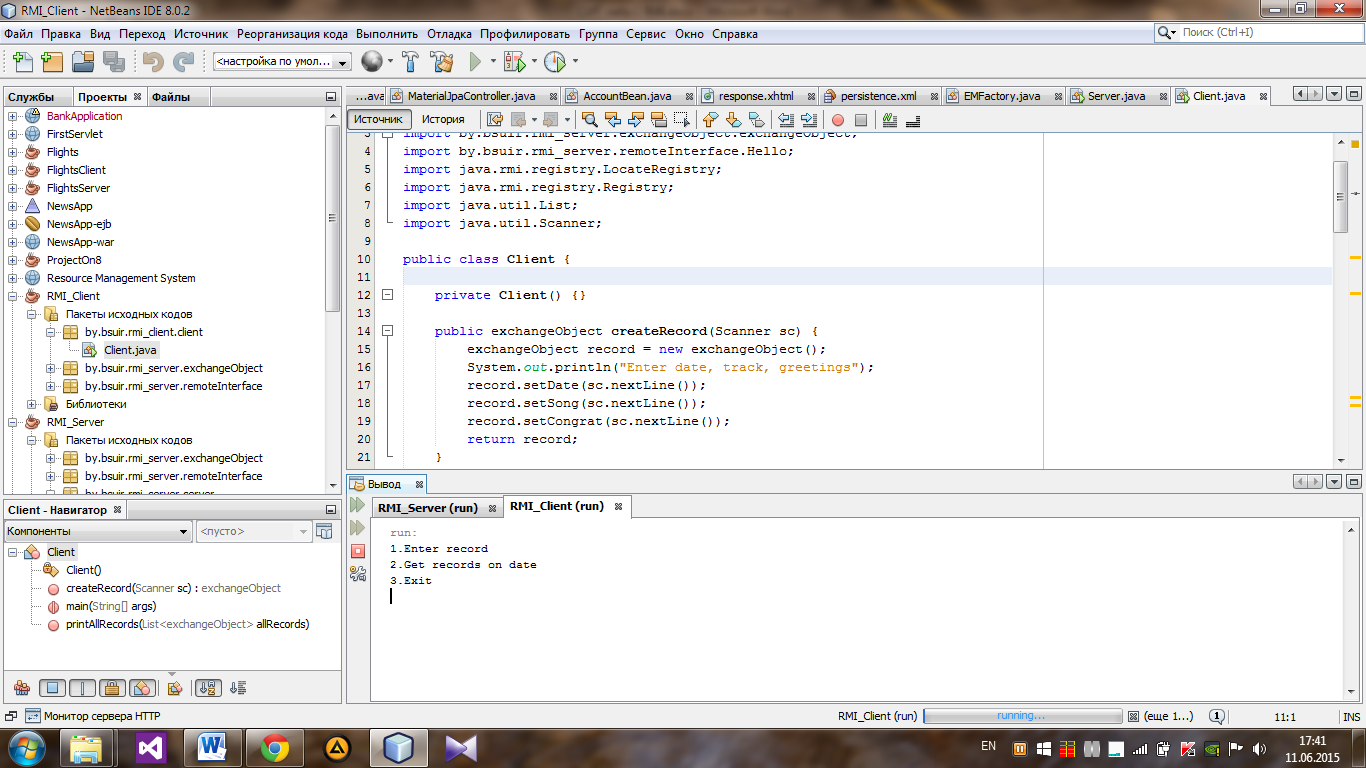


Рисунок 2 – Запускаем клиент

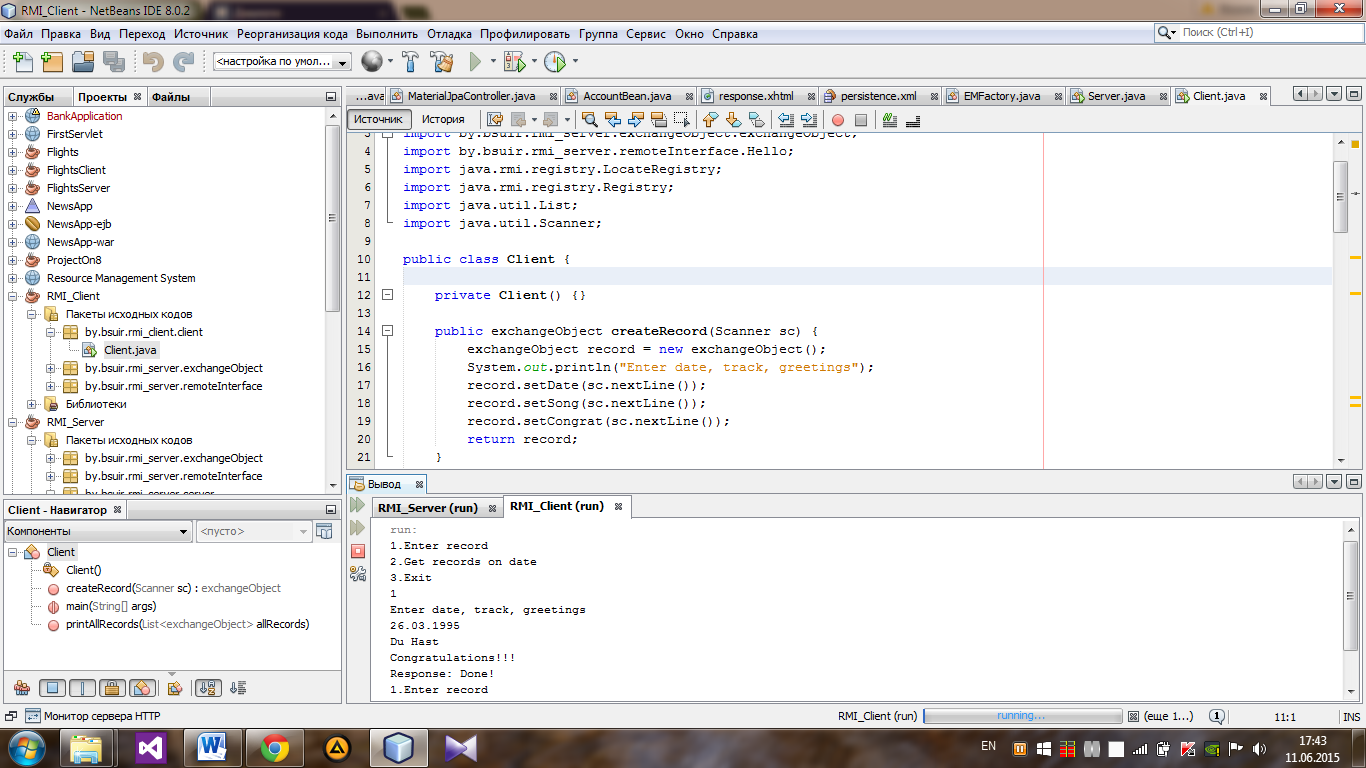


Рисунок 3 – Добавляем запись на сервер в хранилище

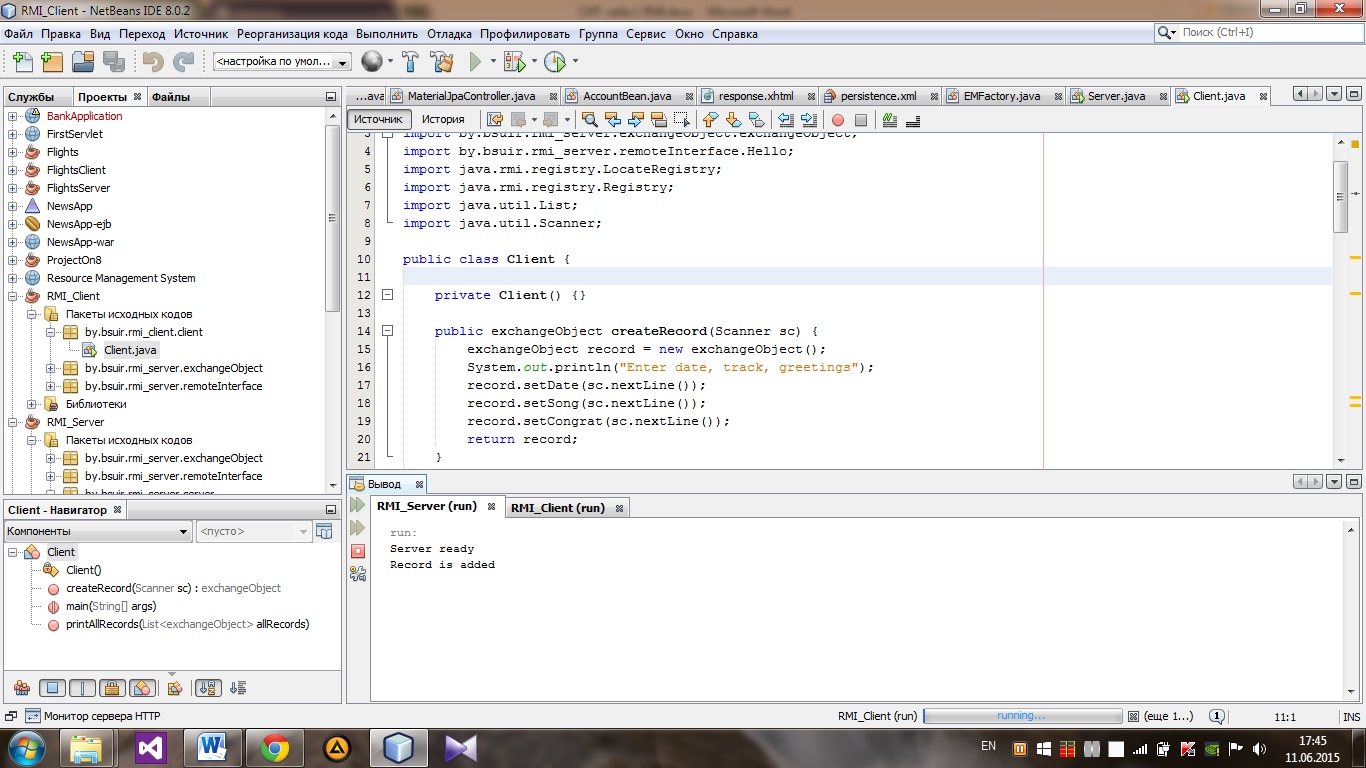


Рисунок 4 – Реакция сервера на добавление записи

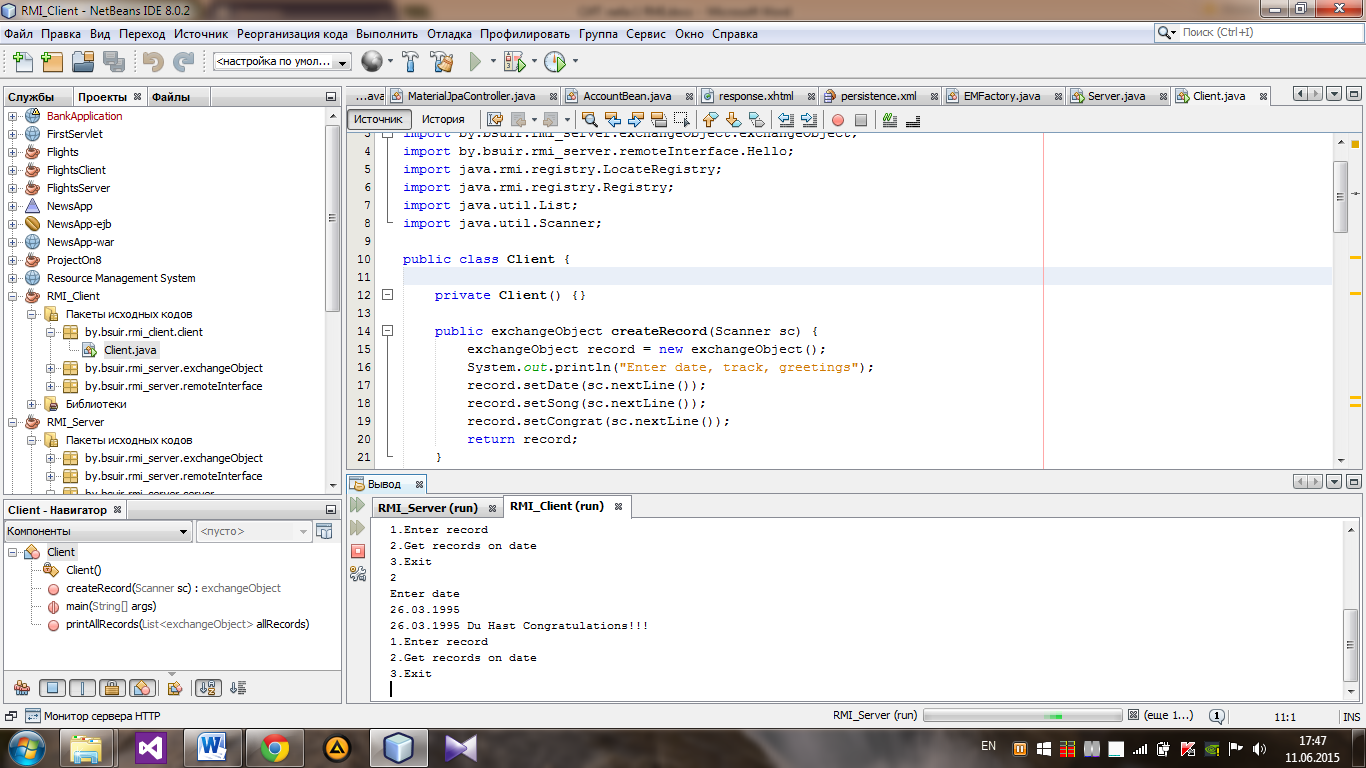


Рисунок 4 – Запрашиваем на клиент все записи на определенную дату

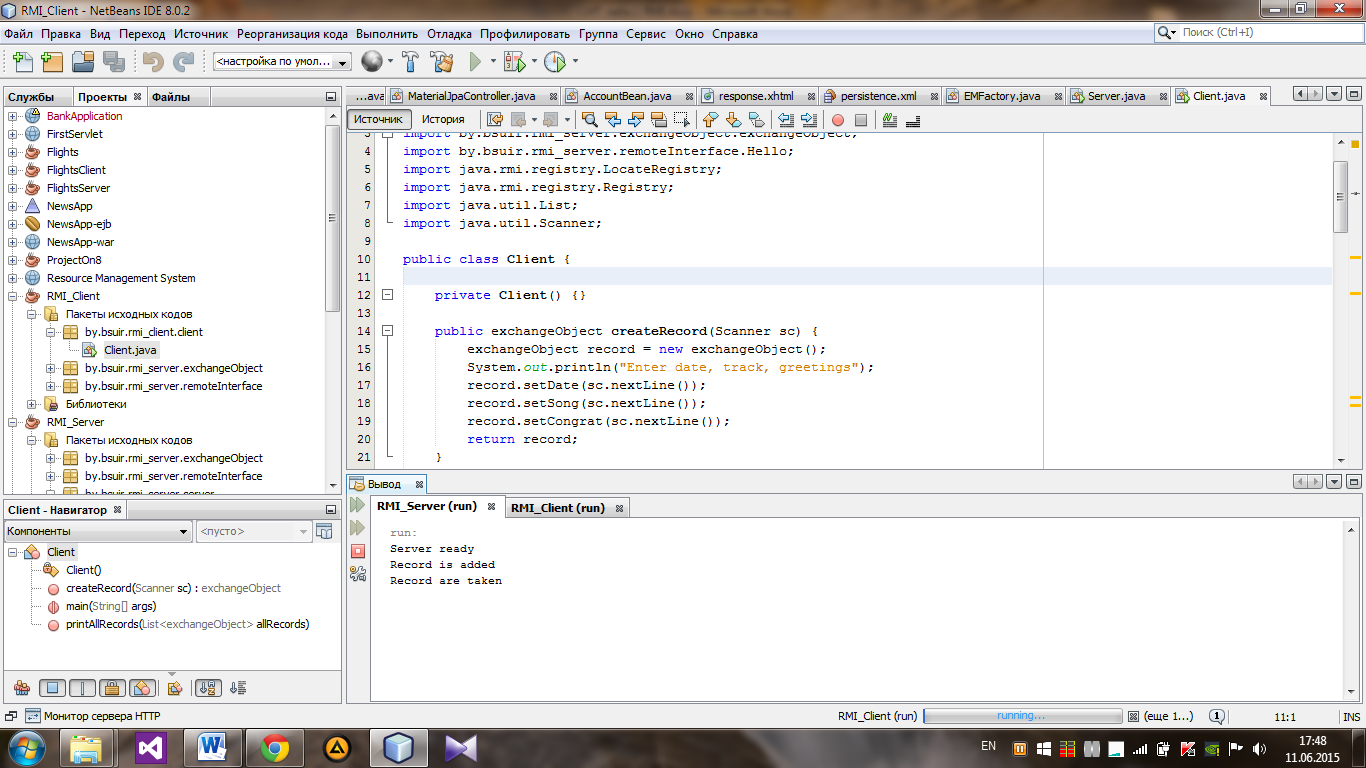


Рисунок 5 – Реакция сервера на запрос записей

**Выводы:** в ходе лабораторной работы изучил технологию RMI и принципы ее реализации. Технология оказалась весьма простой и удобной для использования, т.к. позволила не задумываться о самом процессе обмена данными между клиентом и сервером, благодаря чему освободившееся время можно потратить на расширение функционала распределенной системы и ее отладку.