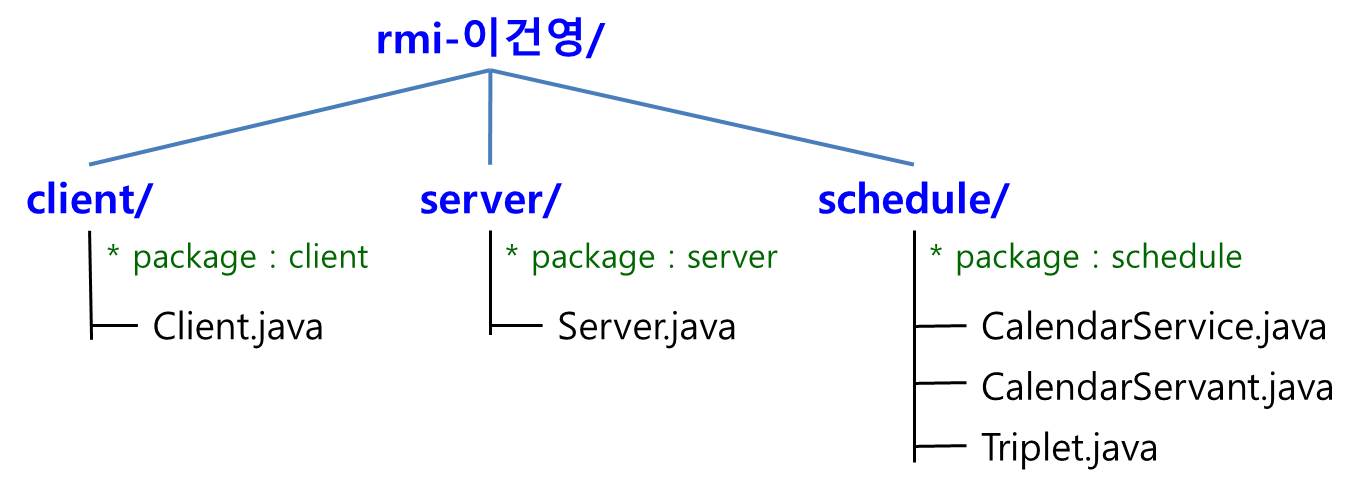
Distributed Computing - RMI Project

2013012115, 이건영

**Ⅰ**. 프로그램 설계 및 구조

• 프로그램의 디렉터리(패키지) 구조는 다음과 같다:   
schedule package에 CalendarService가 실제 RMI 인터페이스가 되며, CalendarServant는 CalendarService를 원격 클래스로써 실제 구현을 하며 Stub 클래스의 역할도 하게 된다. server package의 Server 클래스가 CalendarServant를 선언하여 RMI Registry에 바인딩을 하게 되고 client package의 Client는 룩업을 통해 Stub 클래스를 사용하여 원하는 서비스를 이용한다. schedule package의 Triplet 패키지는 일부 기능의 구현을 위해 만든 3쌍의 자료를 담는 구조이다.



• Schedule Package:  
 1) CalendarService.java - interface이며 Remote를 extends함으로써 실제 제공되는 원격 인터페이스이다. 내부엔 int addSchedule(), void deleteSchedule(), Vector<Triplet> retrieveSchedule() 함수가 포함되어 있으며 모두 RemoteException을 throw 한다.  
 2) CalendarServant.java - UnicastRemoteObject를 상속받으며 원격 객체, Stub 객체의 역할을 하고 동시에 CalendarService에 선언된 함수를 실질적으로 구현한다. 우선 내부적으로 calendar라고 하는 HashMap 구조의 자료구조를 가지고 있는데, 여기에 int형으로 id와 Triplet형으로 Schedule정보를 한 쌍으로 저장을 하게 된다. 이후 addSchedule이나, deleteSchedule, retrieveSchedule 모두 이 calendar라는 자료구조로부터 하나씩 원소를 꺼내어 확인하여 id를 찾고 기간이 겹치는 지 여부를 확인한다. 날짜의 비교는 GregorianCalendar 클래스가 지원하는 before(), after(), equals() method를 통해 원하는 날짜와 비교하며 중복여부를 통해 schedule을 calendar에 추가하거나 가져와서 반환한다.  
 3) Triplet.java – schedule의 정보를 담기 위한 class이며 내부엔 3개의 변수와 각 변수별로 set, get method가 정의되어 있다. 변수들은 각각 schedule의 시작 날짜와 끝 날짜를 나타내는 GregorianCalendar 자료형의 from, to가 있고, schedule 정보를 나타내는 String 자료형의 desc가 있다.

• Server Package: Server.java가 포함되어 있으며 하는 일은 간단하다. schedule package의 CalendarService, CalendarServant를 import하며 “CalendarService aCalendarService = new CalendarServant();”를 통해 Stub 객체를 만들고 1099번 포트에 RMI Registry를 사용하여 바인드 시킨다. 이는 java.rmi.registry 내부의 Registry와 LocateRegistry를 활용하여 적용시켰다. LocateRegistry의 createRegistry로 1099 포트를 Registry 자료형의 registry 변수에 할당하고 해당 변수의 rebind() method를 호출하여 “CalendarService”에 원격객체(aCalendarService)를 bind한다. Binding 후에는 서버가 준비되었다는 메시지를 호출하고 대기한다.

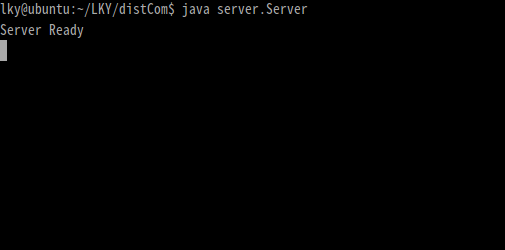
• Client Package: Client.java가 포함되어 있다. Server.java와 마찬가지로 schedule package의 CalendarService, CalendarServant, Triplet까지 import하여 CalendarService형의 aCalendarService 선언까지만 한다. 그 후 마찬가지로 RMI Registry를 사용하여 룩업을 진행하는데, LocateRegistry의 getRegistry()로 local host (host name을 인자로 줄 경우 원격 호스트도 가능)의 1099번 포트를 통해 설정 되어있는 registry를 담아온다. 그 후 해당 registry의 lookup() method를 통해 Server.java에서 bind 했던 “CalendarService”를 마찬가지로 찾아내며 aCalendarService에 할당한다. 이렇게 Client가 이제 원격객체를 사용할 수 있게 된다. aCalendarService를 통해 RMI 인터페이스를 사용하고 미리 구현해 놓은 addSchedule, removeSchedule, retrieveSchedule 세가지 method들을 사용할 수 있게 된다.

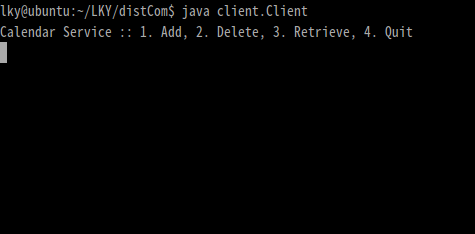
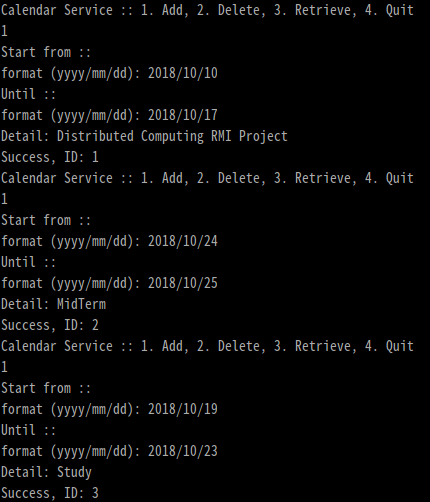
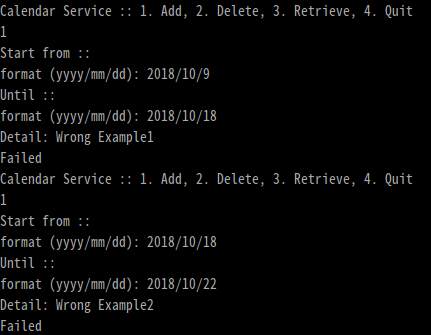
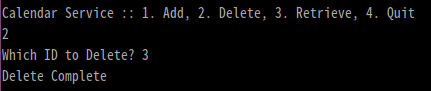
**Ⅱ**. 컴파일 및 실행방법

ㆍ 컴파일 방법은 리눅스 환경에서 JDK가 갖추어만 졌으면 간단하다, 우선 .jar 파일의 압축을 해제하는데 이 때 “jar -xvf rmi-이건영.jar” 명령어를 통해 압축을 해제한다, 압축을 해제한 후에 바로 javac 명령어로 컴파일하면 된다. 모든 디렉터리에 있는 모든 .java 파일을 컴파일해야 하기에 javac \*/\*.java 명령어를 입력해주면 schedule, client, server 모든 파일을 컴파일 할 수 있다.  
(예시) (1) 압축해제   
 (2) 컴파일 

ㆍ 실행 시엔 마찬가지로 리눅스 환경에서 Server 프로세스를 먼저 실행하고 Client를 실행하면 된다. Javac로 컴파일을 완료하면 각 디렉터리에 class 파일이 생기게 되고 Server를 실행 시 java server.Server로 실행하고, client를 실행 시 java client.Client로 실행하면 된다.  
(예시) (1) 서버   
 (2) 클라이언트 

**Ⅲ**. 결 과

ㆍ 서버 화면  
 

ㆍ 클라이언트 화면  
 (1) 메인 화면  
   
 (2) addSchedule()  
 \* Success  
   
 \* Fail  
   
 (3) removeSchedule()  
   
 (4) retrieveSchedule()  
 