**武汉大学国际软件学院**

**实验报告**

**课程名称 网络及分布式计算实验**

**专业年级 2014级软件工程**

**姓 名**

**学 号**

**实验学期 2016-2017　 学年 第1 学期**

**课堂时数 课外时数**

**填写时间 2017 年 5 月 12 日**

**实验任务三**

### 一、CORBA技术简介

简单地说，CORBA允许应用之间相互通信，而不管它们存在于哪里以及是谁设计的。CORBA1.1于1991年由OMG发布，其中定义了接口定义语言（IDL）以及在对象请求代理（ORB）中实现客户对象与服务器对象之间交互的应用编程接口（API）。CORBA2.0于1994年发布，规定了各个供应商之间的ORB的通信规则。

CORBA标准主要分为三个部分：接口定义语言（IDL）、对象请求代理（ORB）以及ORB之间的互操作协议IIOP。

ORB是对象之间建立Client/Server关系的中间件。使用ORB，客户可以透明地调用一个服务对象上的方法，这个服务对象可以在本地，也可以在通过网络连接的其他机器上。ORB截获这一调用同时负责查找实现服务的对象并向其传递参数、调用方法返回最终结果。客户并不知道服务对象位于什么地方，它的编程语言和操作系统是什么，也不知道不属于对象接口的其他系统部分。这样，ORB在异构分布环境下为不同机器上的应用提供了互操作性，并无缝地集成了多种对象系统。

在开发传统的Client/Server应用时，开发者使用他们自己设计的或一个公认的标准来定义用于设备之间通信的协议。协议的定义依赖于实现语言、网络传输和许多其他因素，而ORB的出现简化了这一过程。使用ORB时，协议是使用接口定义语言（IDL）定义的，而IDL是独立于语言的。并且ORB提供很强的灵活性，它使程序员选择最适合的操作系统、执行环境，甚至系统各个组件也可以采用不同的编程语言实现。更重要的是，它允许现有组件的集成。在一个基于ORB的解决方案中，开发者可以使用与创建新对象一样的IDL对遗留系统进行建模，他们创建“包装”代码以在标准化的软件总线与遗留系统接口之间传递信息。

使用CORBA，用户可以透明地访问信息，并不需要知道信息存在于什么软件中、使用什么硬件平台，以及位于企业网络的什么地方。作为面向对象系统的通信核心，CORBA为今天的计算环境带来了真正的互操作性。

### 二、CORBA与JAVA的相互关系

CORBA不只意味着对象请求代理（ORB），它还是非常全面的分布式对象平台。CORBA使JAVA应用可以跨越网络、语言以及操作系统，并为JAVA提供了一组分布服务，如分布式自我观察、动态发现、事务、关系、安全和命名等。

JAVA不仅是一种语言，它还是一个动态代码系统，它对运行对象来说是一个可移植的虚拟机（JVM）。JAVA为开发、管理、发布Client/Server应用提供了更简单的方式。人们可以通过将应用放在一个Web服务器上将这一应用发布给成千上万个用户，而不必关心它的安装和升级。JAVA还非常适合服务器的开发，它可以动态地将服务代码移向最需要它们的地方。

JAVA将会使CORBA对象能够运行在从主机、网络计算机到蜂窝电话等可编程的各种机器上，并简化了大型CORBA系统的代码发布。对客户和服务对象来说JAVA是很理想的编程语言，JAVA内置的多线程、垃圾收集和错误处理使编写健壮的网络对象变得很容易。

这两种对象模型可以很好地相互补充，CORBA处理网络的透明性，JAVA处理实现的透明性，CORBA为JAVA可移植应用环境提供了一个分布式的结构。

### 三、使用JAVA开发CORBA应用的步骤。

1.使用IDL创建接口 (\*.idl)

2.编译接口并生成CORBA支持文件

在idl所在文件夹运行命令行并输入：

idlj -fall Creator.idl

idlj -fall List.idl

3.实现服务器

（1）创建一个 ORB 实例

（2）创建一个服务对象实例（CORBA About 对象的实现）并通知 ORB

（3）获取一个命名上下文的CORBA对象引用，在该命名上下文中注册新的CORBA对象

（4）在命名上下文中将新对象注册在“About”名下

（5）等待对新对象的调用

4.实现客户机

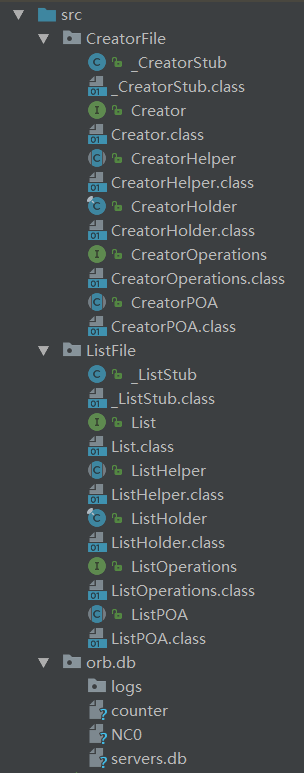
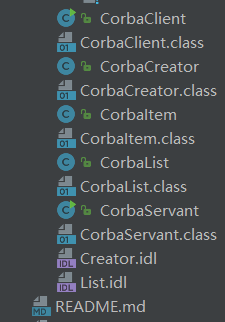
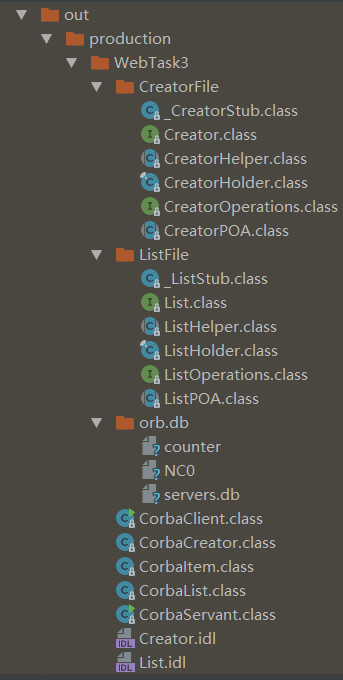
（1）创建一个ORB；

（2）获取一个指向命名上下文的引用；

（3）在命名上下文中查找“Show”并获得指向该 CORBA 对象的引用；

（4）调用对象的 ShowName() 操作并打印结果。   
5.构建和运行程序

在程序所在文件夹运行命令行，按照README.md里的步骤运行。

### 四、程序源代码如下：

Creator.idl

module CreatorFile {  
  
 interface Creator {  
  
 boolean register( in string username, in string password ); };  
};

List.idl

module ListFile {  
  
 interface List {  
  
 boolean add( in string username, in string password, in string start\_time, in string end\_time, in string description );  
 string query( in string username, in string password, in string start\_time, in string end\_time );  
 boolean delete( in string username, in string password, in long item\_id );  
 boolean clear( in string username, in string password );  
 };  
};

CorbaClient.java

import java.io.\*;

import org.omg.CORBA.\*;

import org.omg.CosNaming.\*;

import ListFile.\*;

import CreatorFile.\*;

public class CorbaClient {

static int argSum[] = { 0, 3, 5, 3, 2, 4, 0, 0 };

/\*\*

\* Connect with the servant and do operations which the user input.

\*/

public static void main(String[] args) {

try {

// connect with the servant

ORB orb = ORB.init(args, null);

NameComponent component;

org.omg.CORBA.Object remoteListRef, remoteCreatorRef;

Creator creator;

List list;

org.omg.CORBA.Object object = orb

.resolve\_initial\_references("NameService");

NamingContext namingContext = NamingContextHelper.narrow(object);

component = new NameComponent("Creator", "");

NameComponent[] creators = { component };

remoteCreatorRef = namingContext.resolve(creators);

creator = CreatorHelper.narrow(remoteCreatorRef);

BufferedReader input = new BufferedReader(new InputStreamReader(

System.in));

/\* initialize the variables \*/

String cmd = "";

int choice;

// get the register or login operation

while (true) {

displayMenu();

cmd = input.readLine();

try {

choice = Integer.parseInt(cmd);

if (choice < 0 || choice > 7)

choice = 7;

} catch (Exception e) {

choice = 7;

}

String[] arg = getArgs(argSum[choice], input);

if (choice == 0)

return;

if (choice == 6) {

displayHelp();

continue;

}

if (choice == 7) {

System.err.println("Input Error!");

continue;

}

if (choice == 1) {

if (!arg[0].equals("") && arg[1].equals(arg[2]))

if (creator.register(arg[0], arg[1])) {

System.out.println("Success!");

continue;

}

System.err.println("Fail!");

continue;

}

try {

component = new NameComponent(arg[0], "");

NameComponent[] lists = { component };

remoteListRef = namingContext.resolve(lists);

list = ListHelper.narrow(remoteListRef);

} catch (Exception e) {

System.err.println("Fail!");

continue;

}

if (choice == 2) {

if (list.add(arg[0], arg[1], arg[2], arg[3], arg[4]))

System.out.println("Success!");

else

System.err.println("Fail!");

}

if (choice == 3) {

if (list.delete(arg[0], arg[1], Integer.parseInt(arg[2])))

System.out.println("Success!");

else

System.err.println("Fail!");

}

if (choice == 4) {

if (list.clear(arg[0], arg[1]))

System.out.println("Success!");

else

System.err.println("Fail!");

}

if (choice == 5) {

String q = list.query(arg[0], arg[1], arg[2], arg[3]);

if (q != null)

System.out.println(q);

else

System.err.println("Fail!");

}

}

} catch (Exception e) {

System.err.println(e.toString());

}

}

/\*\*

\* Get the arguments which the operation needs.

\*/

public static String[] getArgs(int s, BufferedReader input)

throws Exception {

String[] a = new String[s];

for (int i = 0; i < s; i++) {

System.out.println("Please input the No." + (i + 1) + " argument");

do

a[i] = input.readLine();

while (a[i].equals(""));

}

return a;

}

/\*\*

\* Display the help documents to guide the user.

\*/

public static void displayHelp() {

System.out.println("Help:\n"

+ "\t 1. You need to input the number "

+ "that the operation you want.\n"

+ "\t 2. The program will do nothing "

+ "if you input a wrong number.\n"

+ "\t 3. The date should be \"YYYY\_M\_D\_h\_m\_s\"\n");

}

/\*\*

\* Display the menu to guide the user to do deeper operations.

\*/

public static void displayMenu() {

System.out.println("Corba Menu:");

System.out.println("\t 0. quit\n" + "\t\targuments: no args\n");

System.out.println("\t 1. register\n"

+ "\t\targuments: <username&gt; <password&gt; <password&gt;");

System.out.println("\t 2. add\n"

+ "\t\targuments: <username&gt; <password&gt; <start&gt; <end&gt; <label&gt;");

System.out.println("\t 3. delete\n"

+ "\t\targuments: <username&gt; <password&gt; <itemid&gt;");

System.out.println("\t 4. clear\n"

+ "\t\targuments: <username&gt; <password&gt;");

System.out.println("\t 5. query\n"

+ "\t\targuments: <username&gt; <password&gt; <start&gt; <end>");

System.out.println("\t 6. help\n" + "\t\targuments: no args");

}

}

CorbaCreator.java

import java.util.\*;

import org.omg.CORBA.\*;

import org.omg.CosNaming.\*;

import org.omg.PortableServer.POA;

import CreatorFile.\*;

public class CorbaCreator extends CreatorPOA {

ORB orb;

org.omg.CORBA.Object obj, remoteRef;

NamingContext namingContext;

NameComponent component;

Vector<String> users = new Vector<String>();

public CorbaCreator(String args[]) {

try {

orb = ORB.init(args, null);

} catch (Exception e) {

}

}

/\*\*

\* Do the register operation.

\*/

public boolean register(String username, String password) {

if (username.equals("") || password.equals(""))

return false;

if (users.contains(username))

return false;

try {

POA rootpoa = (POA) orb.resolve\_initial\_references("RootPOA");

rootpoa.the\_POAManager().activate();

CorbaList list = new CorbaList(username, password);

remoteRef = rootpoa.servant\_to\_reference(list);

obj = orb.resolve\_initial\_references("NameService");

namingContext = NamingContextHelper.narrow(obj);

component = new NameComponent(username, "");

NameComponent[] componentList = { component };

namingContext.rebind(componentList, remoteRef);

users.add(username);

return true;

} catch (Exception e) {

}

return false;

}

}

CorbaItem.java

import java.util.\*;

public class CorbaItem {

private String username;

private int id;

private Date start;

private Date end;

private String label;

/\*\*

\* The default constructor.

\*/

public CorbaItem() {

}

/\*\*

\* The constructor that has all attributes initial.

\*/

public CorbaItem(String u, int i, Date s, Date e, String l) {

username = u;

id = i;

start = s;

end = e;

label = l;

}

/\*\*

\* Return the username.

\*/

public String getUsername() {

return username;

}

/\*\*

\* Return the id.

\*/

public int getId() {

return id;

}

/\*\*

\* Return the start time.

\*/

public Date getStart() {

return start;

}

/\*\*

\* Return the end time.

\*/

public Date getEnd() {

return end;

}

/\*\*

\* Return the lable.

\*/

public String getLabel() {

return label;

}

/\*\*

\* Return the description of the item.

\*/

public String toString() {

return "\n No." + id + ": " + label + "\n\t Start: " + start.toString()

+ "\n\t End: " + end.toString();

}

}

CorbaList.java

import java.util.\*;

import ListFile.\*;

public class CorbaList extends ListPOA {

/\* a hash table to store the user informations \*/

private String username, password;

/\* a vector to store the items \*/

private Vector<CorbaItem> items;

/\* count number of items added as the id of items \*/

private static int counter = 1;

/\* count number of items deleted \*/

private static int deleted = 0;

/\*\*

\* The default construstor.

\*/

public CorbaList(String username, String password) {

this.username = username;

this.password = password;

items = new Vector<CorbaItem>();

}

/\*\*

\* Do the login operation.

\*/

public boolean login(String username, String password) {

if (username.equals(this.username) && password.equals(this.password))

return true;

return false;

}

/\*\*

\* Do the add operation.

\*/

public boolean add(String username, String password, String start,

String end, String label) {

Date s = toDate(start);

Date e = toDate(end);

if (!login(username, password))

return false;

if (s == null || e == null)

return false;

if (s.after(e))

return false;

CorbaItem item = new CorbaItem(username, counter, s, e, label);

items.add(item);

counter++;

return true;

}

/\*\*

\* Do the query operation.

\*/

public String query(String username, String password, String start,

String end) {

Date s = toDate(start);

Date e = toDate(end);

if (!login(username, password))

return "Fail!";

if (s == null || e == null)

return "Fail!";

if (s.after(e))

return "Fail!";

CorbaItem item;

Vector<CorbaItem> v = new Vector<CorbaItem>();

for (Iterator it = items.iterator(); it.hasNext();) {

item = (CorbaItem) it.next();

if (e.before(item.getStart()) || s.after(item.getEnd()))

continue;

v.add(item);

}

if (v.isEmpty())

return "[No items here.]";

return v.toString();

}

/\*\*

\* Do the delete operation.

\*/

public boolean delete(String username, String password, int id) {

if (!login(username, password))

return false;

CorbaItem item;

for (int i = 0; i < items.size(); i++) {

item = items.get(i);

if (id == item.getId()) {

items.remove(i);

i--;

deleted++;

return true;

}

}

return false;

}

/\*\*

\* Do the clear operation.

\*/

public boolean clear(String username, String password) {

if (!login(username, password))

return false;

items.clear();

return true;

}

/\*\*

\* Translate a string to date.

\*/

public static Date toDate(String date) {

StringTokenizer token = new StringTokenizer(date, "\_");

if (token.countTokens() != 6)

return null;

int year = Integer.parseInt(token.nextToken());

int month = Integer.parseInt(token.nextToken());

int day = Integer.parseInt(token.nextToken());

int hour = Integer.parseInt(token.nextToken());

int minute = Integer.parseInt(token.nextToken());

int second = Integer.parseInt(token.nextToken());

Date d = new Date(year, month, day, hour, minute, second);

return d;

}

}

CorbaServant.java

import org.omg.CORBA.\*;

import org.omg.CosNaming.\*;

import org.omg.PortableServer.\*;

public class CorbaServant {

/\*\*

\* Run the servant and waiting for call from clients.

\*/

public static void main(String[] args) {

try {

ORB orb = ORB.init(args, null);

POA rootpoa = (POA) orb.resolve\_initial\_references("RootPOA");

rootpoa.the\_POAManager().activate();

org.omg.CORBA.Object object = orb

.resolve\_initial\_references("NameService");

CorbaCreator creator = new CorbaCreator(args);

org.omg.CORBA.Object remoteRef = rootpoa

.servant\_to\_reference(creator);

NamingContext namingContext = NamingContextHelper.narrow(object);

NameComponent component = new NameComponent("Creator", "");

NameComponent[] componentList = { component };

namingContext.rebind(componentList, remoteRef);

orb.run();

} catch (Exception e) {

System.err.println(e.toString());

}

}

}

README.md

First to deal with the idl files and then compile the java file,

later we start the system.

Please follow these steps:

1.Open the command line at the project's src directory.

2.Type the following command one by one.

idlj -fall Creator.idl

idlj -fall List.idl

javac \*.java

javac CreatorFile/\*.java

javac ListFile/\*.java

start ORBD

java CorbaClient