Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

Распределенные информационные системы

Отчёт

по лабораторной работе №2

«Технология Java TM IDL. Архитектура CORBA»

Выполнила: студент гр. 714301

Макаренко А.Д

Проверил: Кунцевич А.А.

Минск 2020

*Цель:* научиться проектировать распределенную объектную систему на основе архитектуры CORBA.

Вариант 4. Разработать приложение в архитектуре клиент-сервер с тремя интерфейсами, один из которых является промежуточным, а два других выполняют кодирование и декодирование слов, которые посылает клиент.

Ход работы

1. IDL interfaces

*module HelloApp {*

*interface Hello {*

*oneway void shutdown();*

*};*

*interface CryptSys {*

*string crypt(in string str);*

*};*

*interface DecryptSys {*

*string decrypt(in string str);*

*};*

*interface BetweenSys {*

*string findMethod(in string user\_message);*

*};*

*};*

1. Реализация сервера

*public* *class* HelloServer {

*public* *static* *void* main(*String* *args*[]) {

try{

*// создание и инициализация ORB*

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

*// получим ссылку на rootpoa и активируем POAManager*

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

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

*// создадим сервант и зарегестрируем его в ORB*

*HelloImpl* helloImpl = new HelloImpl();

*BetweenSysImpl* betweenSysImpl = new BetweenSysImpl();

helloImpl.setORB(orb);

betweenSysImpl.generateKey();

*//получим объектную ссылку от серванта*

*org*.*omg*.*CORBA*.*Object* ref = rootpoa.servant\_to\_reference(betweenSysImpl);

*BetweenSys* between\_href = BetweenSysHelper.narrow(ref);

*// получим корневой элемент контекста имен*

*org*.*omg*.*CORBA*.*Object* objRef = orb.resolve\_initial\_references("NameService");

*// используем NamingContextExt, что является частью спецификации //Interoperable Naming Service*

*NamingContextExt* ncRef = NamingContextExtHelper.narrow(objRef);

*// привязка объектной ссылки в сервисе имен*

*NameComponent* path[] = ncRef.to\_name("BetweenSys");

ncRef.rebind(path, between\_href);

System.out.println("HelloServer ready and waiting ...");

*// ожидаем вызовов от клиентов*

orb.run();

} catch (*Exception* *e*) {

System.err.println("ERROR: " + e);

e.printStackTrace(System.out);

}

System.out.println("HelloServer Exiting ...");

}

}

1. Реализация клиента

*public* *class* HelloClient {

*static* *Hello* helloImpl;

*static* *BetweenSys* betweenSysImpl;

*public* *static* *void* main(*String* *args*[]) {

try{

*// создание и инициализация ORB*

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

*// получение корня контекста имен*

*org*.*omg*.*CORBA*.*Object* objRef = orb.resolve\_initial\_references("NameService");

*// опять используем INS*

*NamingContextExt* ncRef = NamingContextExtHelper.narrow(objRef);

*// найдем объектную ссылку по имени*

betweenSysImpl = BetweenSysHelper.narrow(ncRef.resolve\_str("BetweenSys"));

System.out.println("Obtained a handle on server object: " + helloImpl);

*int* process\_code = 0;

*Scanner* in = new Scanner(System.in);

System.out.println(

"Please write method name and text for crypt/decrypt:\n" +

"Example: crypt # Hellow word!\n" +

"For exit write: exit'"

);

while (process\_code == 0 && in.hasNext()) {

*String* string = in.nextLine();

if (string == "exit") {

process\_code = 1;

} else {

System.out.println(betweenSysImpl.findMethod(string));

}

}

in.close();

helloImpl.shutdown();

} catch (*Exception* *e*) {

System.out.println("ERROR : " + e) ;

e.printStackTrace(System.out);

}

}

}

1. Реализация crypt класса

*class CryptSysImpl extends CryptSysPOA {*

*private static Cipher ecipher;*

*private static SecretKey key;*

*public void setKey(SecretKey secretKey) {*

*key = secretKey;*

*}*

*public String crypt(String str) {*

*try {*

*ecipher = Cipher.getInstance("DES");*

*ecipher.init(Cipher.ENCRYPT\_MODE, key);*

*byte[] utf8 = str.getBytes("UTF8");*

*byte[] enc = ecipher.doFinal(utf8);*

*return new String(Base64.getEncoder().encodeToString(enc));*

*} catch (NoSuchAlgorithmException e) {*

*return "No Such Algorithm:" + e.getMessage();*

*} catch (NoSuchPaddingException e) {*

*return "No Such Padding:" + e.getMessage();*

*} catch (InvalidKeyException e) {*

*return "Invalid Key:" + e.getMessage();*

*} catch (Exception e) {*

*e.printStackTrace();*

*}*

*return "";*

*}*

*}*

1. Реализация decrypt класса

*class DecryptSysImpl extends DecryptSysPOA {*

*private static Cipher dcipher;*

*private static SecretKey key;*

*public void setKey(SecretKey secretKey) {*

*key = secretKey;*

*}*

*public String decrypt(String str) {*

*try {*

*// generate secret key using DES algorithm*

*dcipher = Cipher.getInstance("DES");*

*dcipher.init(Cipher.DECRYPT\_MODE, key);*

*// decode with base64 to get bytes*

*byte[] dec = Base64.getDecoder().decode(str.getBytes());*

*byte[] utf8 = dcipher.doFinal(dec);*

*return new String(utf8, "UTF8");*

*} catch (NoSuchAlgorithmException e) {*

*return "No Such Algorithm:" + e.getMessage();*

*} catch (NoSuchPaddingException e) {*

*return "No Such Padding:" + e.getMessage();*

*} catch (InvalidKeyException e) {*

*return "Invalid Key:" + e.getMessage();*

*} catch (Exception e) {*

*e.printStackTrace();*

*}*

*return "";*

*}*

*}*

1. Реализация промежуточного класса

*class BetweenSysImpl extends BetweenSysPOA {*

*private static SecretKey key;*

*public void generateKey() {*

*try {*

*key = KeyGenerator.getInstance("DES").generateKey();*

*} catch (NoSuchAlgorithmException e) {*

*System.out.println("No Such Algorithm:" + e.getMessage());*

*}*

*}*

*public String findMethod(String user\_message) {*

*CryptSysImpl cryptSysImpl = new CryptSysImpl();*

*DecryptSysImpl decryptSysImpl = new DecryptSysImpl();*

*cryptSysImpl.setKey(key);*

*decryptSysImpl.setKey(key);*

*String[] method\_string = user\_message.split("#");*

*switch(method\_string[0].trim()) {*

*case "crypt":*

*return cryptSysImpl.crypt(method\_string[1].trim());*

*case "decrypt":*

*return decryptSysImpl.decrypt(method\_string[1].trim());*

*default:*

*return "String invalid!";*

*}*

*}*

*}*

1. Реализация класса для инициализации ORB

*class HelloImpl extends HelloPOA {*

*private ORB orb;*

*public void setORB(ORB orb\_val) {*

*orb = orb\_val;*

*}*

*public String sayHello() {*

*return "\nHello world !!\n";*

*}*

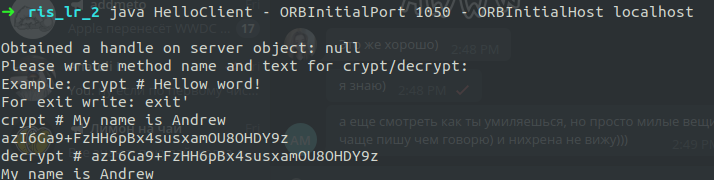
*public void shutdown() {*

*orb.shutdown(false);*

*}*

*}*

1. Компиляция и запуск; работа клиента



*Выводы:* в ходе данной работы научился проектировать распределенную объектную систему на основе архитектуры CORBA