**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО**”

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»**

Студент: Калинин Даниил Леонидович

Группа: P41141

Преподаватель: Дергачев Андрей Михайлович

\

Санкт-Петербург

2021

Задание:

Требуется разработать приложение, осуществляющее регистрацию сервиса в реестре jUDDI, а также поиск сервиса в реестре и обращение к нему. Рекомендуется реализовать консольное приложение, которое обрабатывает 2 команды. Итог работы первой команды – регистрация сервиса в реестре; вторая команда должна осуществлять поиск сервиса, а также обращение к нему.

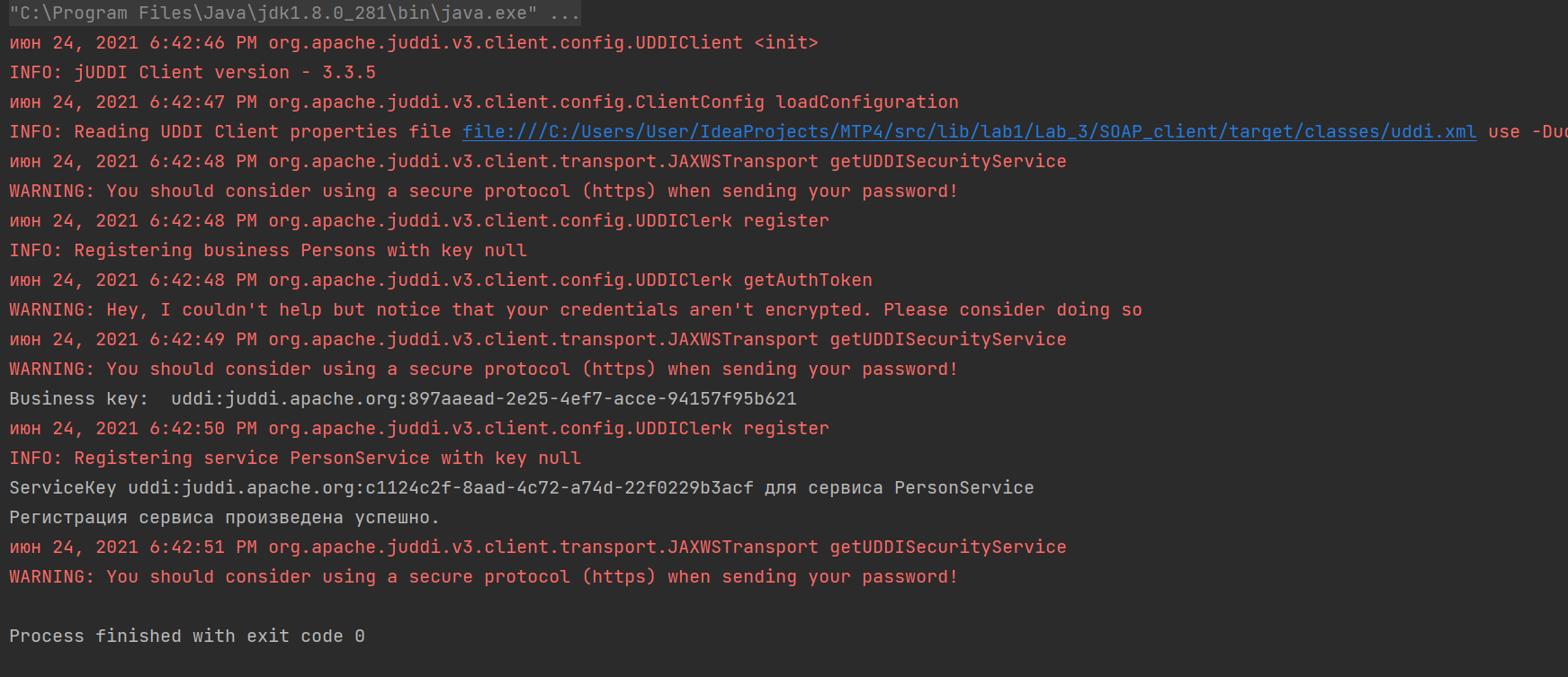
Этапы выполнения:

1. На первом шаге был развернут реестр jUDDI v. 3.2.0. Затем была реализована регистрация разработанного ранее web-сервиса в реестре jUDDI. Ниже приведен код классов регистрации сервиса

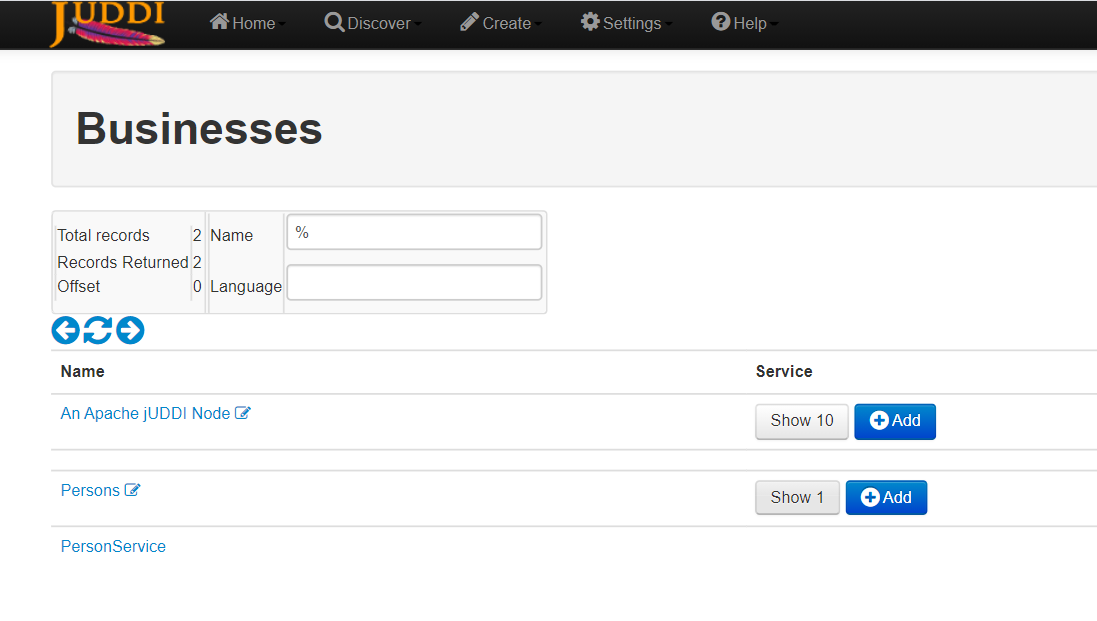
public class Client {  
  
 private final static Logger *log* = Logger.*getLogger*(Client.class.getName());  
  
 public static void main(String[] args) {  
 try {  
 UddiService uddiService =  
 new UddiService("uddi.xml");  
 String URL = "http://localhost:8081/PersonService?wsdl";  
 String serviceKey = uddiService.publish("Persons", "PersonService", URL);  
 System.*in*.read();  
 uddiService.removeBusiness(serviceKey);  
 uddiService.logOut();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

public class UddiService {  
 private UDDIClerk clerk;  
  
 private UDDISecurityPortType security;  
 private UDDIInquiryPortType inquiry;  
 private BusinessService businessService;  
  
 public UddiService(String pathToUddiSettings) {  
 try {  
 UDDIClient uddiClient = new UDDIClient(pathToUddiSettings);  
 this.clerk = uddiClient.getClerk("default");  
 security = uddiClient.getTransport("default").getUDDISecurityService();  
 inquiry = uddiClient.getTransport("default").getUDDIInquiryService();  
 if (clerk == null)  
 throw new Exception("the clerk wasn't found, check the config file!");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 private Name getName(String name,  
 String lang) {  
 Name busName = new Name();  
 busName.setValue(name);  
 busName.setLang(lang);  
 return busName;  
 }  
  
 private String ListToString(List<Name> name) {  
 StringBuilder sb = new StringBuilder();  
 for (int i = 0; i < name.size(); i++) {  
 sb.append(name.get(i).getValue()).append(" ");  
 }  
 return sb.toString();  
 }  
  
 private Description getDescription(String descr) {  
 Description description = new Description();  
 description.setValue(descr);  
 description.setLang("ru-ru");  
 return description;  
 }  
  
 private String ListToDescString(List<Description> name) {  
 StringBuilder sb = new StringBuilder();  
 for (Description description : name) {  
 sb.append(description.getValue()).append(" ");  
 }  
 return sb.toString();  
 }  
  
 private BusinessEntity registerBusiness(String businessName) {  
 BusinessEntity businessEntity = new BusinessEntity();  
  
 businessEntity.getName().add(getName(businessName, "ru-ru"));  
 businessEntity.getDescription().add(getDescription(businessName));  
 BusinessEntity register = clerk.register(businessEntity);  
 if (register == null) {  
 System.*out*.println("Ошибка регистрации...");  
 return null;  
 }  
 String businessKey = register.getBusinessKey();  
 System.*out*.println("Business key: " + businessKey);  
 return register;  
 }  
  
 private BusinessService registerService(String businessKey,  
 String servName,  
 String URL) {  
 BindingTemplate bindingTemplate = new BindingTemplate();  
 AccessPoint accessPoint = new AccessPoint();  
 accessPoint.setUseType(AccessPointType.*WSDL\_DEPLOYMENT*.toString());  
 accessPoint.setValue(URL);  
 bindingTemplate.setAccessPoint(accessPoint);  
  
 BindingTemplates bindingTemplates = new BindingTemplates();  
 bindingTemplates.getBindingTemplate().add(bindingTemplate);  
  
 BusinessService businessService = new BusinessService();  
 businessService.getName().add(getName(servName, "ru-ru"));  
 businessService.getDescription().add(getDescription(servName));  
 businessService.setBusinessKey(businessKey);  
 businessService.setBindingTemplates(bindingTemplates);  
 BusinessService svc = clerk.register(businessService);  
  
 return svc;  
 }  
  
 public String publish(String businessName,  
 String serviceName,  
 String URL) {  
 try {  
 BusinessEntity businessEntity = this.registerBusiness(businessName);  
 if (businessEntity == null) {  
 System.*out*.println("Ошибка регистрации бизнеса...");  
 *exit*(1);  
 }  
  
 BusinessService svc = registerService(businessEntity.getBusinessKey(),  
 serviceName,  
 URL);  
 if (svc == null) {  
 System.*out*.println("Непредвиденная ошибка регистрации веб-сервиса.");  
 *exit*(1);  
 }  
  
 businessService = svc;  
 String serviceKey = svc.getServiceKey();  
 System.*out*.println(String.*format*("ServiceKey %s для сервиса %s", serviceKey, serviceName));  
  
 System.*out*.println("Регистрация сервиса произведена успешно.");  
 return serviceKey;  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
  
  
 public void printServiceWSDLINFOToConsole(String WSDL\_url) throws Exception {  
 URL url = new URL(WSDL\_url);  
 URLConnection conn = url.openConnection();  
  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder = factory.newDocumentBuilder();  
  
 Document document = builder.parse(conn.getInputStream());  
  
 TransformerFactory factory1 = TransformerFactory.*newInstance*();  
 Transformer xform = factory1.newTransformer();  
  
 File file = new File("write.xml");  
 if (!file.exists()) {  
 file.createNewFile();  
 }  
  
 StreamResult streamResult = new StreamResult();  
 streamResult.setSystemId(file);  
  
 xform.transform(new DOMSource(document), streamResult);  
 xform.transform(new DOMSource(document), new StreamResult(System.*out*));  
 }  
  
 public void printServiceDetail(BusinessService businessService) throws Exception {  
 if (businessService == null) {  
 return;  
 }  
 System.*out*.println("Name: " + ListToString(businessService.getName()));  
 System.*out*.println("Desc: " + ListToDescString(businessService.getDescription()));  
 System.*out*.println("Key: " + (businessService.getServiceKey()));  
  
 JAXB.*marshal*(businessService, System.*out*);  
  
 printServiceWSDLINFOToConsole(businessService.getBindingTemplates().getBindingTemplate().get(0).getAccessPoint().getValue());  
  
 }  
  
 public void removeBusiness(String businessKey) {  
 clerk.unRegisterBusiness(businessKey);  
 }  
  
  
 public void logOut() {  
 clerk.discardAuthToken();  
 }  
  
 public BusinessService getBusinessService() {  
 return businessService;  
 }  
}

Результат работы программы



Сервис успешно зарегистрирован в реестре JUDDI



Вывод:

В ходе выполнения лабораторной работы было произведено развертывание реестра сервисов jUDDI, после чего в развернутый реестр был добавлен PersonWebSerivce, созданный в ходе выполнения первых трех лабораторных работ