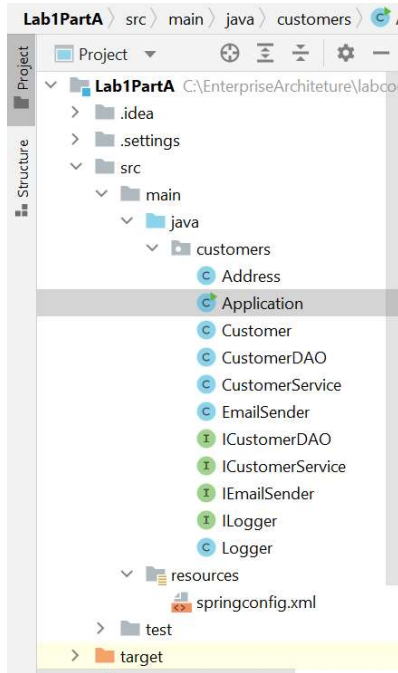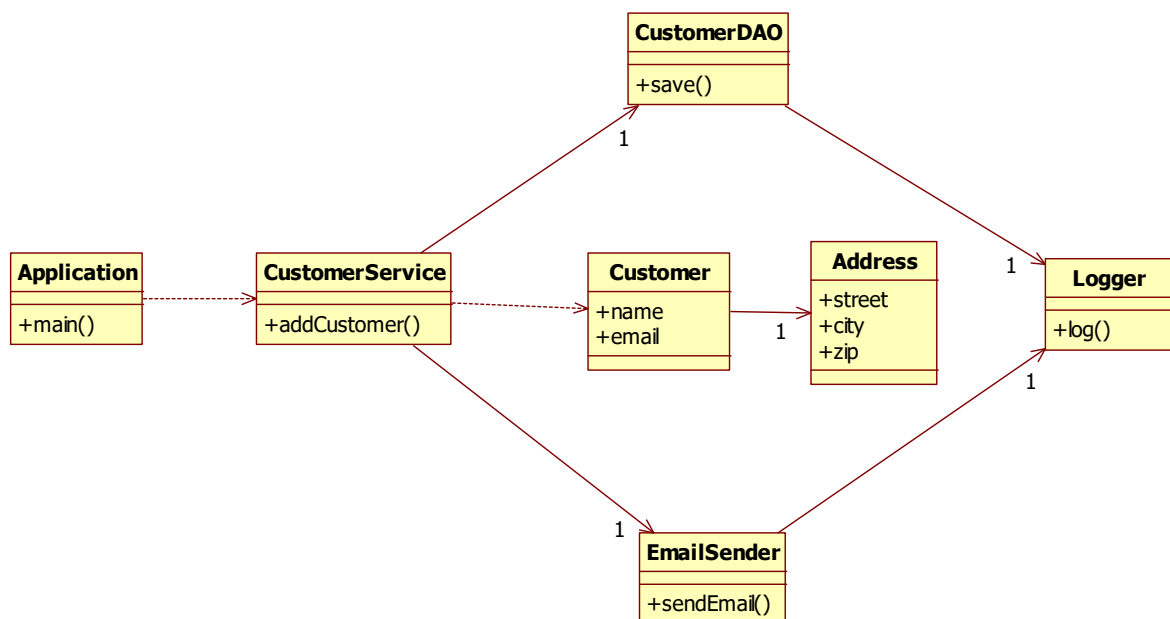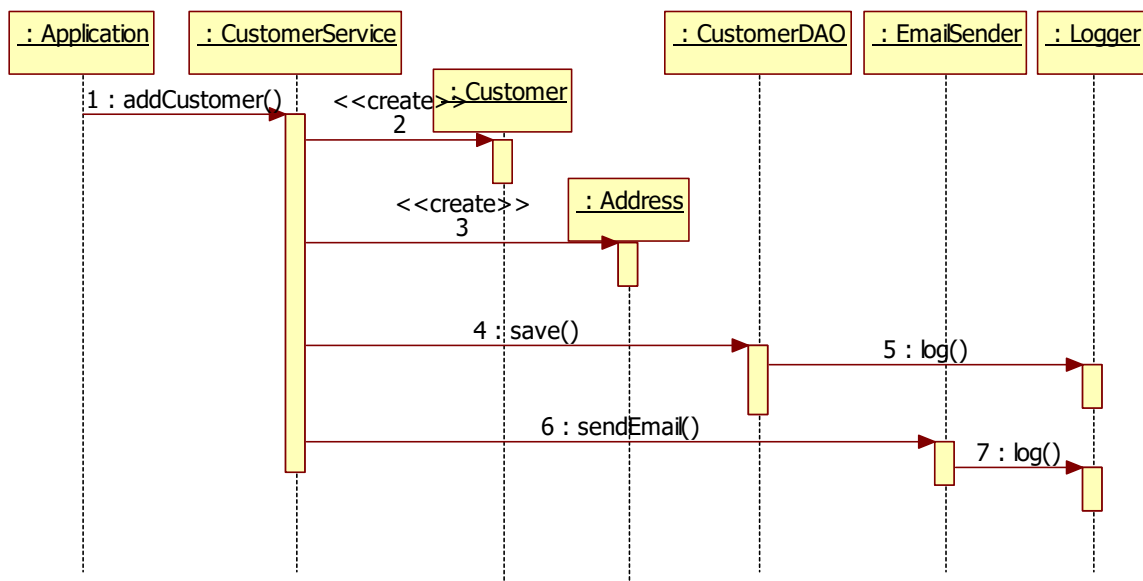# Lab 1

Part A

Given is the project **Lab1PartA**. Open the project in your IDE.

The given application has the following classes:

And works as follows:

In the folder **src/main/resources** you find the following configuration file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="customerService" class="customers.CustomerService"/>

</beans>
```

In Application.java you find the following code:

```java
public class Application {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(
                "springconfig.xml");
        ICustomerService customerService = context.getBean("customerService",
                ICustomerService.class);

        customerService.addCustomer("Frank Brown", "fbrown@acme.com",
                "mainstreet 5", "Chicago", "60613");
    }
}
```

The Application gets the CustomerService object out of the context and calls the method addCustomer().
Also note that the rest of the application is all hard wired, meaning that the CustomerService creates the CustomerDAO and the EmailSender:
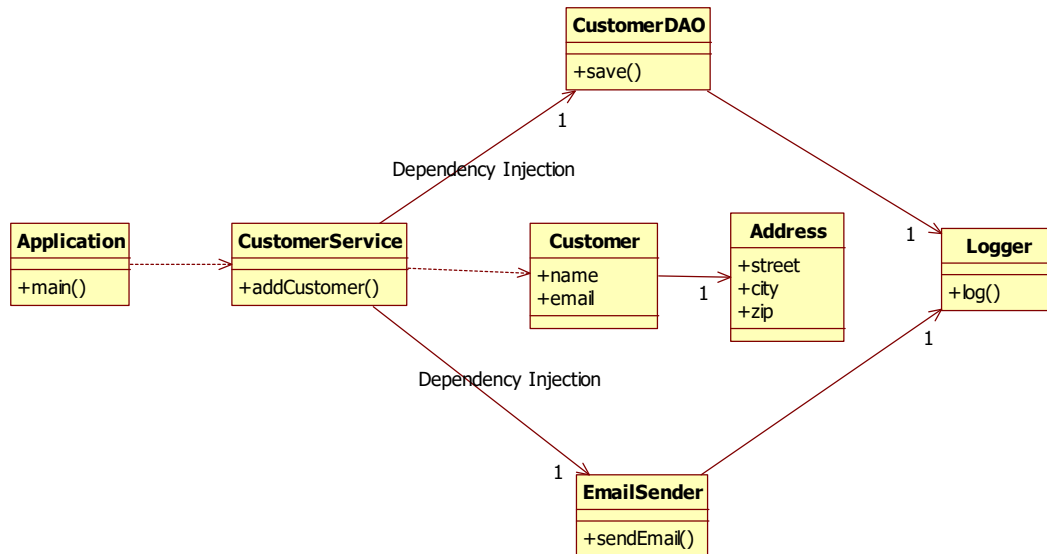
```java
public class CustomerService implements ICustomerService {
    ICustomerDAO customerDAO = new CustomerDAO();
    IEmailSender emailSender = new EmailSender();
```

Also the CustomerDAO and the EmailSender create the Logger:

```java
public class CustomerDAO implements ICustomerDAO{
    private ILogger logger = new Logger();

public class EmailSender implements IEmailSender {
    String outgoingMailServer = "smtp.acme.com";
    private ILogger logger = new Logger();
```
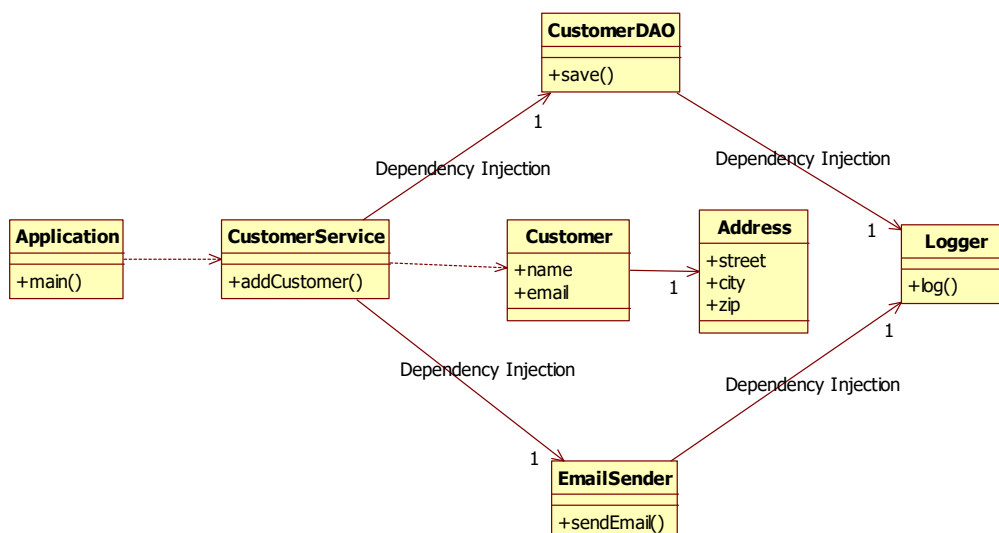
1. Modify the application so that we inject the CustomerDAO and the EmailSender in the CustomerService using setter injection

CustomerDAO

+save()

1

Dependency Injection

Application | **CustomerService** | **Customer** | **Address** | 1 | **Logger**
+main() | +addCustomer() | +name +email | +street +city +zip | | +log()

1

1

Dependency Injection

1

EmailSender

+sendEmail()

2. Modify the application so that we inject the Logger in the CustomerDAO and the EmailSender using constructor injection

CustomerDAO

+save()

1

Dependency Injection   Dependency Injection

Application | **CustomerService** | **Customer** | **Address** | 1 | **Logger**
+main() | +addCustomer() | +name +email | +street +city +zip | | +log()

1

Dependency Injection   1

Dependency Injection   Dependency Injection

1

EmailSender

+sendEmail()

Part B – Dependency injection with classpath scanning

In the project window in IntelliJ right-click **Lab1PartA** and select **Copy->Copy**
The right-click on a white area of the project window again and select **Paste**.

Give the new project the name **Lab1PartB** and click **OK**.
In IntelliJ open now **Lab1PartB**.
Now modify **Lab1PartB** such that it is configured using classpath scanning.

Part C – Dependency injection with Java Config

Copy and paste the **Lab1PartB** to **Lab1PartC**.
Now modify the **Lab1PartC** project such that it is configured using Java configuration

Part D – Dependency injection with Java Config + classpath scanning + autowiring

Copy and paste the **Lab1PartC** to **Lab1PartD**.
Now modify the **Lab1PartD** project such that it is configured using Java configuration + classpath scanning + autowiring

Part E

Copy and paste the **Lab1PartD** to **Lab1PartE**.

Modify the solution of **Lab1PartE** with the following additional requirement:
Add a ProductService class that allows you to create a new Product and save it to the database. Send an email every time you add a new product. Every time you go to the database or send an email, write this to the log file using the Logger class.

What to hand in:
1. A separate zip file with the solution of part A
2. A separate zip file with the solution of part B
3. A separate zip file with the solution of part C
4. A separate zip file with the solution of part D
5. A separate zip file with the solution of part E