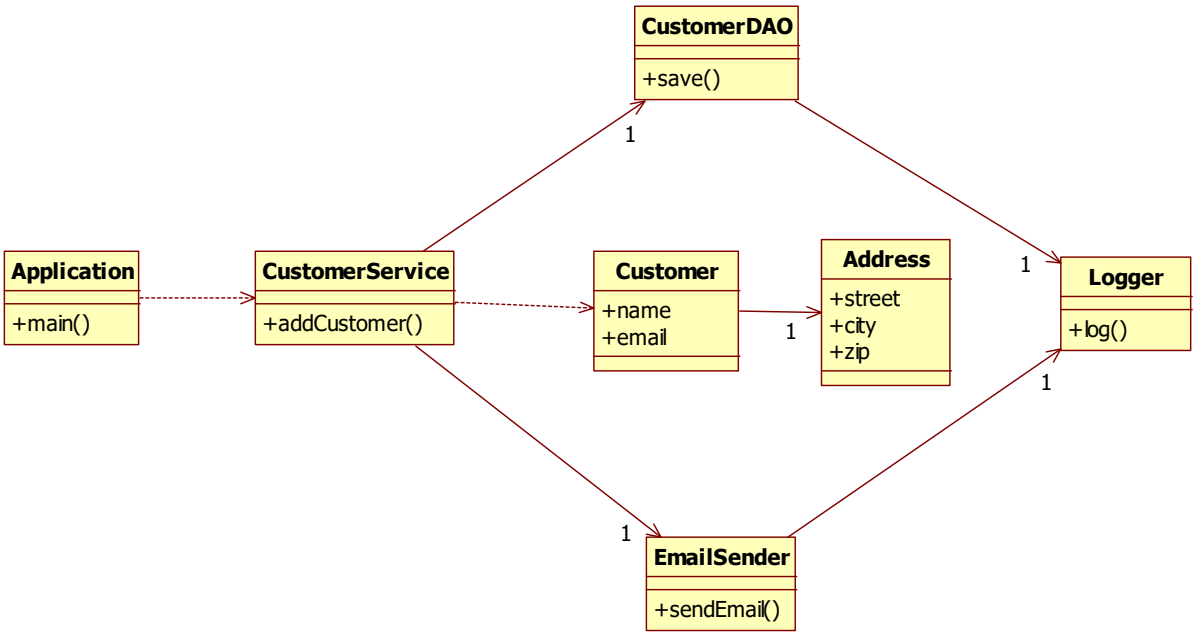
Lab 12

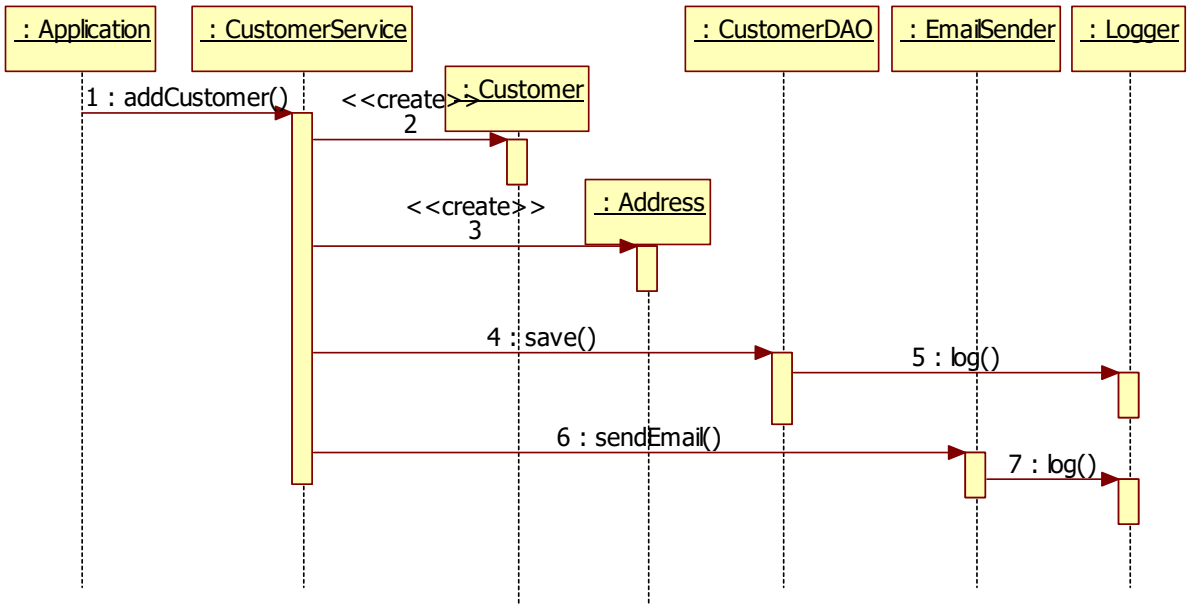
Part a.

Given is the project **SimpleSpring**.

The given application has the following classes:



And works as follows:



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

```
<?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:

```
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:

```
public class CustomerService implements ICustomerService {
    ICustomerDAO customerDAO = new CustomerDAO();
    IEmailSender emailSender = new EmailSender();
```

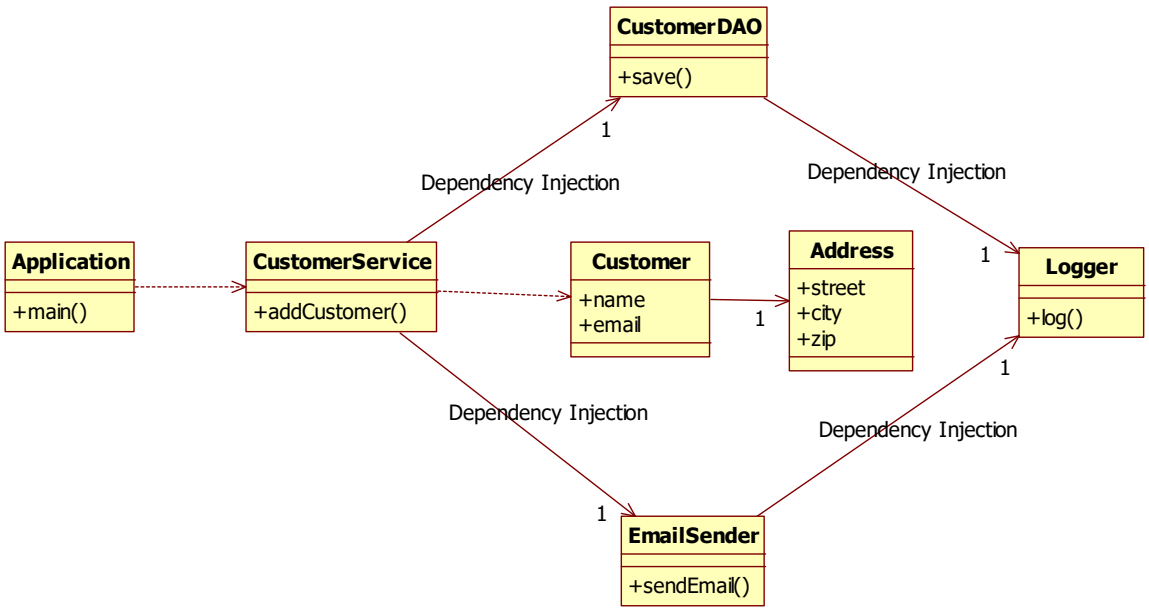
Also the CustomerDAO and the EmailSender create the Logger:

```
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();
```

Modify the application so that we inject the CustomerDAO and the EmailSender in the CustomerService using setter injection

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



b. Dependency injection with classpath scanning

Copy and paste the project of part a, and give the new project a different name.
Now modify this project such that it is configured using classpath scanning instead of xml configuration.

c. Dependency injection with Java Config

Copy and paste the project of part b, and give the new project a different name.
Now modify this project such that it is configured using Java configuration

d. Dependency injection with Java Config + classpath scanning + autowiring

Copy and paste the project of part b, and give the new project a different name.
Now modify this project such that it is configured using the most simple configuration with Java configuration + classpath scanning + autowiring

e. Spring boot
Given is the spring boot project SpringbootProject which shows just the basics of spring boot. Copy the classes of part d to this project,and modify it so that it works as a Spring boot application.

f. Profiles
Modify the solution of part e with the following additional requirement:
Write a CustomerDAOMock class that we want to use in our test environment. In our production environment we want to use the existing CustomerDAO class. Use profiles so that we can configure in the application.properties file which CustomerDAO we want to inject.