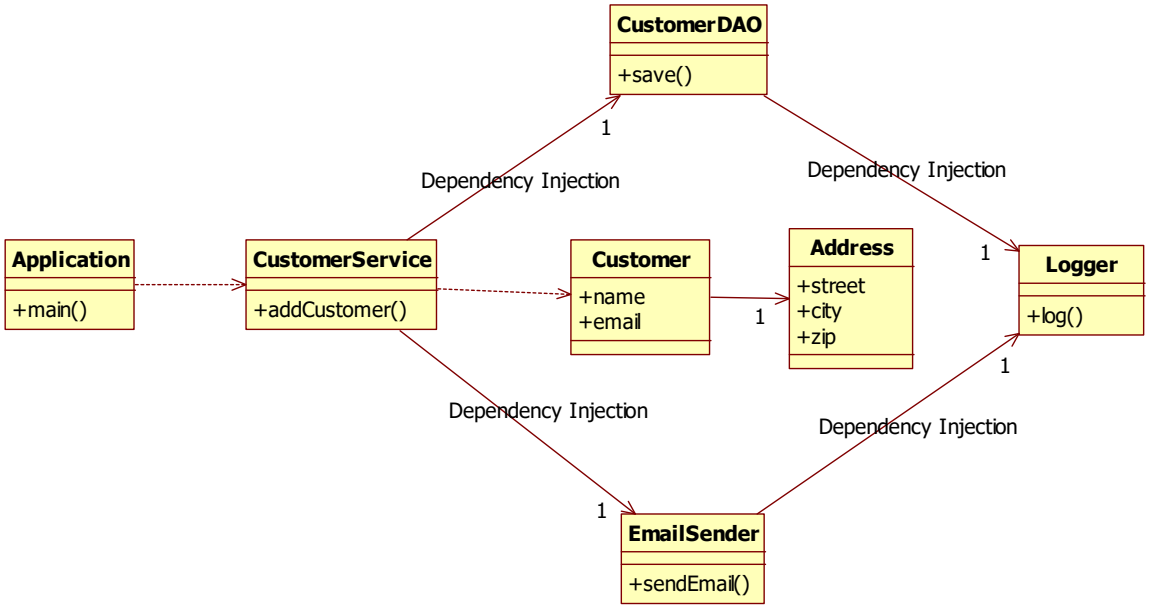
Lab 13

Part a: AOP

Given is the project **SpringBootLab13**.



If we run the application, we get the following output in the console:

```
CustomerDAO: saving customer Frank Brown
Logging 2018-03-20T11:23:44.588 Customer is saved in the DB: Frank
Brown
EmailSender: sending 'Welcome Frank Brown as a new customer' to
fbrown@acme.com
Logging 2018-03-20T11:23:44.588 Email is sent: message= Welcome Frank Brown
as a new customer , emailaddress =fbrown@acme.com
```

a. Modify the application so that whenever the sendEmail method on the EmailSender is called, a log message is created (using an after advice AOP annotation). This should produce the following output:

```
CustomerDAO: saving customer Frank Brown
Logging 2018-03-20T11:23:44.588 Customer is saved in the DB: Frank
Brown
EmailSender: sending 'Welcome Frank Brown as a new customer' to
fbrown@acme.com
Logging 2018-03-20T11:23:44.588 Email is sent: message= Welcome Frank Brown
as a new customer , emailaddress =fbrown@acme.com
2018-03-20T11:23:44.789 method=sentEmail
```

In order to use AOP in a Spring Boot project, we have to add the following dependency in the POM file:

```
<dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>1.8.12</version>
      <scope>compile</scope>
</dependency>
```

b. Now change the log advice in such a way that the email address and the message are logged as well. You should be able to retrieve the email address and the message through the arguments of the **sendEmail()** method. This should produce the following output:

```
CustomerDAO: saving customer Frank Brown
Logging 2018-03-20T11:23:44.588 Customer is saved in the DB: Frank
Brown
EmailSender: sending 'Welcome Frank Brown as a new customer' to
fbrown@acme.com
Logging 2018-03-20T11:23:44.588 Email is sent: message= Welcome Frank Brown
as a new customer , emailaddress =fbrown@acme.com
2018-03-20T11:23:44.789 method=sentEmail address=fbrown@acme.com
message= Welcome Frank Brown as a new customer
```

c. Change the log advice again, this time so that the outgoing mail server is logged as well. The **outgoingMailServer** is an attribute of the **EmailSender** object, which you can retrieve through the **joinpoint.getTarget()** method. This should produce the following output:

```
CustomerDAO: saving customer Frank Brown
Logging 2018-03-20T11:23:44.588 Customer is saved in the DB: Frank
Brown
EmailSender: sending 'Welcome Frank Brown as a new customer' to
fbrown@acme.com
Logging 2018-03-20T11:23:44.588 Email is sent: message= Welcome Frank Brown
as a new customer , emailaddress =fbrown@acme.com
2018-03-20T11:23:44.789 method=sentEmail address=fbrown@acme.com
message= Welcome Frank Brown as a new customer outgoing mail server
= smtp.mydomain.com
```

d. Write a new advice that calculates the duration of the method calls to the DAO object and outputs the result to the console. Spring provides a stopwatch utility that can be used for this by using the following code:

```
import org.springframework.util.StopWatch;

public Object invoke(ProceedingJoinPoint call ) throws Throwable {
   StopWatch sw = new StopWatch();
   sw.start(call.getSignature().getName());
   Object retVal = call.proceed();
   sw.stop();

   long totaltime = sw.getLastTaskTimeMillis();
   // print the time to the console

   return retVal
```

Part b: Events

In the application of part a, do the following 2 things using events:

1. Whenever we add a new Customer on the CustomerService, publish an NewCustomerEvent. Write a new class called AdvertisementService that subscribes to NewCustomerEvent's and writes the Customer information to the console

2. Write another class called CustomerRatingService that subscribes to NewCustomerEvent's and writes the Customer information to the console