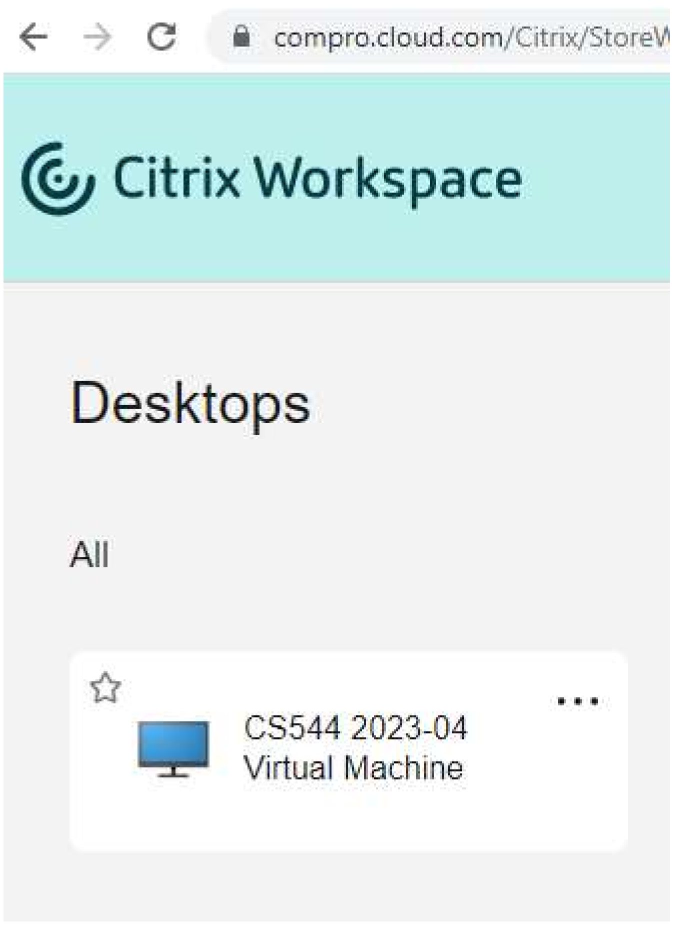
Lab 1

1. In your browser (preferably Chrome) create an **incognito window** and browse to: ***compro.cloud.com***

2. Login using your Windows Active Directory lab username and password. Enter your username in the following format: cs.mum.edu\123456 (where 123456 is your student ID)

3. Once you login, you should be able to see a large rectangular icon that bears the name of your VM. Click on it.



- If this is the first time that you are running Citrix, it will prompt you to "Detect Citrix Workspace" which will install a client application to run the remote session. You can always choose "Use Browser" to run your remote session in a new browser tab using HTML5 or you can install the client application. Client app takes longer to load and takes a bit of time to install but creates a more stable experience with your remote session.

- Please never shut down your VM. It is safe to restart, but never shut down. If you do shut down your machine, we will have to manually restart it for you, and you will lose precious time.

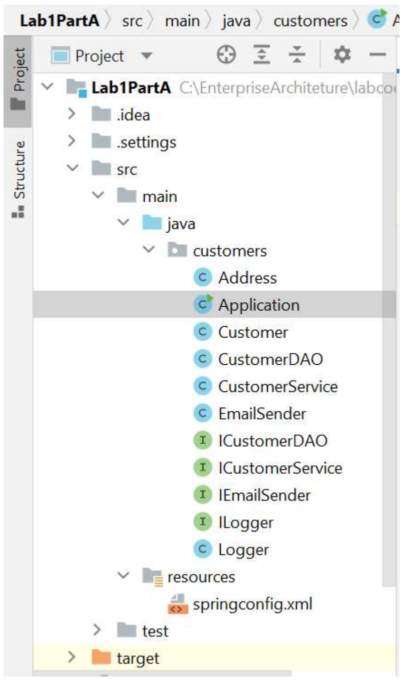
For any issues related to Citrix and Active directory usernames and passwords, please contact

- Utsav Pokhrel at **upokhrel@miu.edu**
- Trishakti Madhikarmi: tmadhikarmi@miu.edu

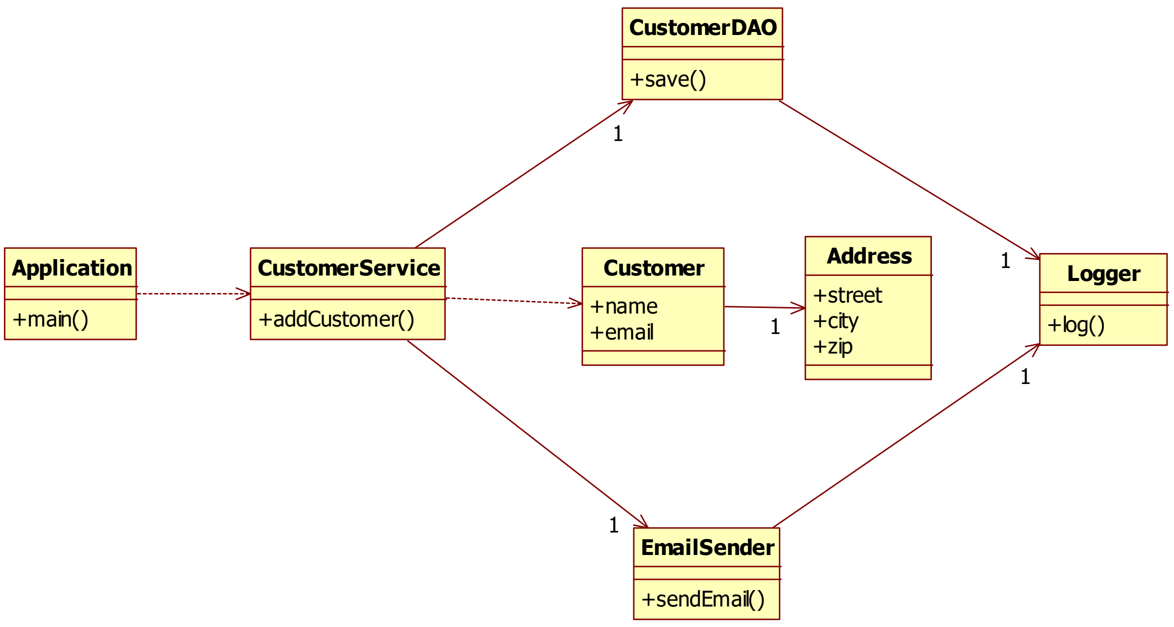
Contact them on Teams.

Part A

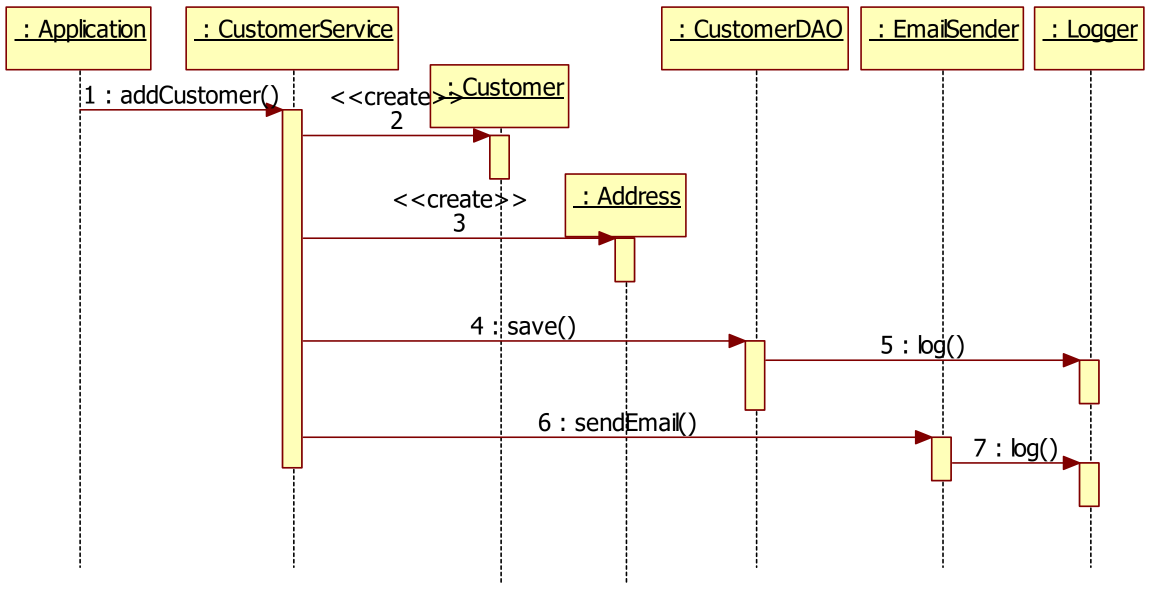
Given is the project **Lab1PartA**. Open the project in your IDE.



The given application has the following classes:



And works as follows:



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

```
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<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/bean
s 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:

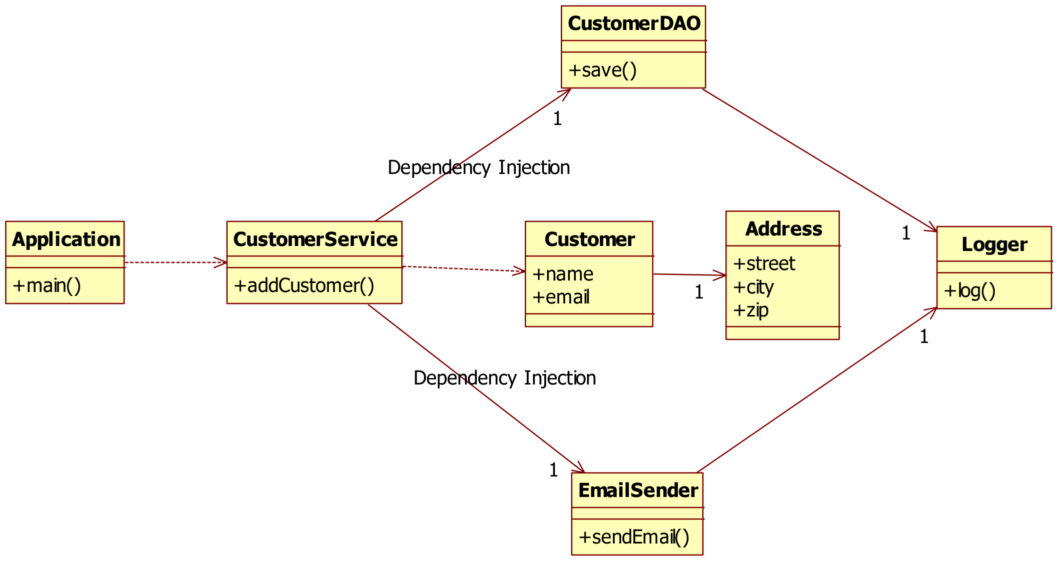
```java
public class CustomerService implements ICustomerService {
    ICustomerDAO customerDAO = new CustomerDAO();
    IEmailSender emailSender = new EmailSender();
```

Also the CustomerDAO and the EmailSender create the Logger:

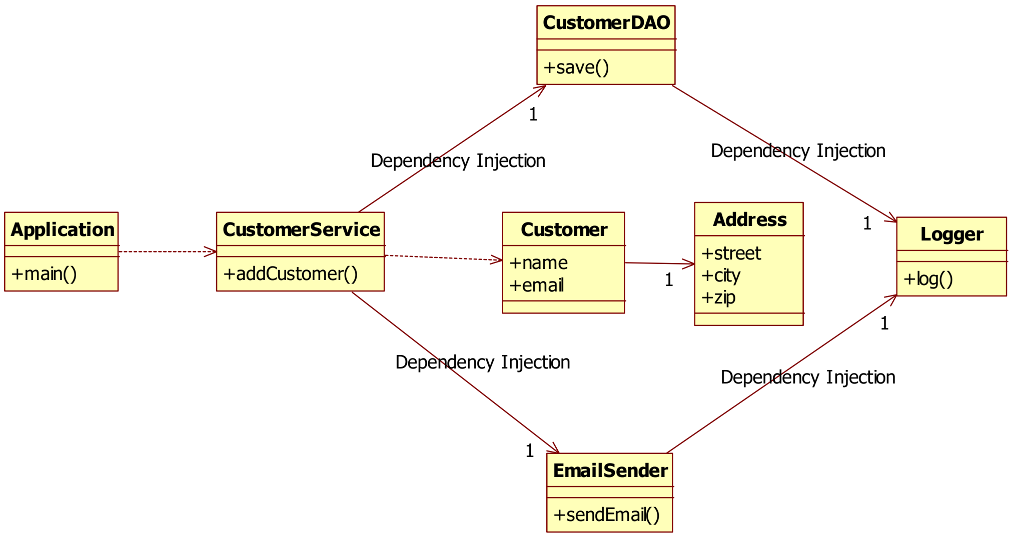
```java
public class CustomerDAO implements ICustomerDAO{
    private ILogger logger = new Logger();
```

```java
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



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



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